Attention for Human Motion Captioning : Towards an Interpretable Semantic Motion Segmentation and Analysis

Karim Radouane

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR

DE L'INSTITUT MINES-TÉLÉCOM (IMT)

ÉCOLE NATIONALE SUPÉRIEURE DES MINES D'ALÈS (IMT MINES ALÈS)

En Informatique

École doctorale I2S - Information, Structures, Systèmes

Portée par l'Université de Montpellier

Unité de recherche EuroMov Digital Health in Motion

Attention for Human Motion Captioning: Towards an Interpretable Semantic Motion Segmentation and Analysis

Présentée par Karim Radouane

Le 26 Février 2024

Sous la direction de Sylvie Ranwez, et l'encadrement d'Andon Tchechmedjiev et de Julien Lagarde

Devant le jury composé de

Hazem Wannous, Professeur, IMT Nord Europe	Rapporteur
Karteek Alahari, Directeur de recherche, INRIA, Université Grenoble Alpes	Rapporteur, Président
Gül Varol, Chargée de recherche, LIGM, École des Ponts ParisTech	Examinatrice
Sylvie Ranwez, Professeure, EuroMov DHM, IMT Mines Alès	Directrice
Andon Tchechmedjiev, MCF, EuroMov DHM, IMT Mines Alès	Encadrant
Julien Lagarde, HDR, MCF, EuroMov DHM, Université de Montpellier	Encadrant

IMT Mines Alès
École Mines-Télécom

To my parents, Khadija Hanini and Lhoussine Radouane.

Acknowledgements

These three years of my thesis were full of challenges, and this work wouldn't have been possible without the various motivations I received.

First and foremost, I would like to express my deepest gratitude to God for making this dream a reality and giving me the strength to accomplish this dissertation.

I would like to thank my advisors for the multiple meetings we had, for revising my manuscript, and providing valuable feedback. I am especially thankful to my supervisor, Andon Tchechmedjiev, for his time spent to conduct long scientific discussions, answering countless work-related messages to help me achieve the best in my thesis.

My appreciation goes to Sylvie Ranwez for welcoming me into this thesis and giving me the opportunity to freely draw my own research path. I extend my thanks to Julien Lagarde for the various discussions we had about the human sciences.

I would like to express my thanks to the members of the jury for participating in the evaluation of this work and for the interest they have given to it.

I should not forget to recognize Pierre Richard for helping me solve many technical problems. My thanks also goes to Edith Teychene for helping me with administrative procedures and being available to answer my questions each time I visited her office.

It is worth acknowledging the Occitanie Region for funding my thesis during three years. Additionally, I want to express my appreciation to everyone who has contributed, either directly or indirectly, through valuable discussions, especially my lab friends and fellow researchers.

Finally, very special thanks to my family for their support throughout my pursuit for education. I am eternally grateful for my parents Khadija and Lhoussine who provided me with good educational environment and encouraging me to follow the doctoral studies. I also want to convey my gratitude to my brothers, Amine, Souhail and Kamal, who were my first public audience with whom I used to vulgarize my scientific work for a very long time.

Table of Contents

Chapter 1

Introduction

1.1 Context

Our current society is characterized by a strong dynamic of movement, whether as an essential component of human health or within a multitude of sports disciplines where each athlete aspires to excellence and performance. Movement is thus at the core of numerous scientific studies.

The work presented in this manuscript was conducted within the EuroMov Digital Health in Motion (EuroMov DHM) interdisciplinary research lab, whose research orientations pertain to the study of human sensorimotor plasticity. The lab comprises experts in movement sciences, physicians, and other healthcare professionals, as well as experts in computer science and artificial intelligence. The main objective of EuroMov DHM is to understand the sensorimotor signatures of physical and mental health in humans through innovative computational approaches in order to enhance the quality of life, health and well-being, as well as to help with the functional rehabilitation of patients after musculoskeletal disorders, stroke or neurodegenerative illnesses.

Motion-Language processing constitutes an emerging field within computer vision, with evident links to cognitive science, computational linguistics and knowledge engineering, robotics, social computing. The current prism of study of the interplay between motion and language is mainly focused on preliminary exploratory investigations pertaining to the general association between close descriptions of the movements and its parameters, mainly motion capture. The task of Motion to Language mapping or translation consists of mapping sensor-motion parameters (*e.g.*, MoCap) to natural language descriptions. Currently, few datasets include direct motion-to-description correspondences, enabling the use of supervised machine learning, mainly the KIT Motion-Language Dataset (KIT-ML) (Plappert et al., 2016) and HumanML3D (Guo et al., 2022a).

The literature provides an extensive research focused on generating motion from language, as detailed in the survey (Zhu et al., 2023b), but relatively few studies have been dedicated to the reverse direction: motion to language (Guo et al., 2022b; Plappert et al., 2018). Moreover, these tasks have traditionally been approached as a global mapping between motion and an overall description. However, the semantic analysis of human motion calls for a more fine-grained mapping that involves decomposing both motion and language. Solving this task enables the aligned transcription of human motion, with diverse applications, particularly in sign language research. Furthermore, the captioning of images or videos has introduced interpretable designs, identifying zones that contribute most, especially with the introduction of adaptive and guided attention approaches. However, interpretability analysis remains unexplored in the context of motion-language mapping.

The thesis defended in this context primarily focuses on text generation through interpretable architectures for the analysis and semantic segmentation of human motion and its applications. The goal is to enable the unsupervised resolution of related tasks such as synchronization between motion and text, spatio-temporal identification of elements describing a primitive of human motion.

1.2 Human motion understanding

There are different ways to interpret human motion, but they differ in the level of richness in information. In this context, the representation of movement can take

various forms and involve numerous parameters, and many research questions could be raised towards this goal of human movement understanding.

Why do we need to represent motion? Generally, the process of analyzing human motion involves a quantification step, which is strictly necessary for motion analysis using machine learning methods. These methods are implemented in a wide range of applications, such as detecting anomalies, monitoring the progress of patients in physical rehabilitation, providing recommendations for sports performance improvement, and ensuring athlete monitoring to prevent injuries (Song et al., 2023). They are also employed in virtual reality and robotics contexts to develop systems that mimic human movements. In this context, the use of machine learning leveraging 2D and/or 3D joint coordinates has been widely adopted for human motion analysis across various levels of granularity, ranging from simple action recognition (Qin et al., 2022) to detailed motion description using natural language (Plappert et al., 2018; Guo et al., 2022b).

What parameters allow the representation of movement? There are several methods for representing movement. The first method involves using sensors placed on the human body to collect kinematic data describing the movement. More precise technique rely on the use of Motion Capture system to obtain the global spatial position of human pose. However, motion capture systems are usually expensive and not practical in all scenarios. Less expensive methods relies on using video data for estimating movement parameters, like 2D pose estimation (Xu et al., 2022; Cao et al., 2019) and 3D pose estimation systems (Wang et al., 2021). Other works propose utilizing the human pose data to learn more robust and deep representations (Zhu et al., 2023a). Invariant measures, could also be used, either computed manually (Takano and Lee, 2020) or learned, such as in (Yang et al., 2022; Li et al., 2020, 2018). These motion representations, incorporating invariance properties, are useful for reducing variability between movements representing the same action, which can facilitate action recognition. Chapter 2 will present a detailed review on techniques of human motion quantification through pose estimation. Other representations are build on the top of estimated poses or Mocap data. The human pose in motion is also a natural graph evolving in time. This inherent characteristic allows the use of spatio-temporal encoding (Salih et al., 2016; Chi et al., 2022). In our context, some of these relevant representation will be discussed in detail with some applications in Chapter 3.

The semanticization of human motion is at the core of this thesis. The human motion interpretation could take different forms. Focusing on motion to language

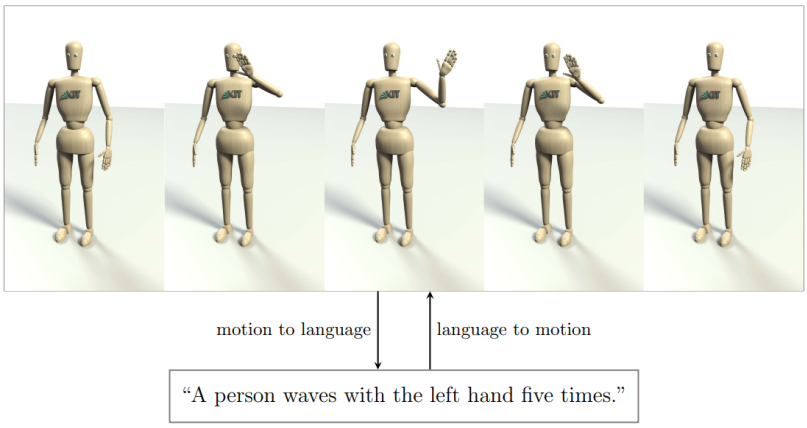
mapping, we will first provide a brief analysis of the works done in this context and highlight the limitations, some of which will be addressed in this manuscript.

Understanding human motion through motion-language mapping. For example, motion generation could be conditioned on a sequence of continuous actions (Lee et al., 2023) (Figure 1.1b), or conditioned on descriptive text of human action and the studied scene (Wang et al., 2022) (Figure 1.1c). In this manuscript, we take particular focus on the natural description of human motion, represented as a sequence of poses. From this perspective, the literature provides numerous works that aim to create a mapping between motion and language. In this field, earlier work was introduced by (Plappert et al., 2018), exploring the bidirectional generation of both motion and text (Figure 1.1a) based on bidirectional recurrent models. More recently, both modes of generation have been addressed by (Toyoda et al., 2022; Guo et al., 2022b). Focusing on motion generation, a variety of techniques have been applied. (Ghosh et al., 2021) propose motion synthesis (text-driven animation) using Gated Recurrent Units (GRUs). Advanced techniques leverage transformers (Vaswani et al., 2017). For instance, (Petrovich et al., 2022) encode motion and text through a transformer employing the learning concept of Variational AutoEncoder (VAE) (Kingma and Welling, 2022). The VAE was also used in other works (Guo et al., 2022a). More recently, (Zhang et al., 2023) proposed an architectural design incorporating a generative model based on VQ-VAE (Vector Quantized-Variational Autoencoder) (van den Oord et al., 2017). (Chen et al., 2023) uses the diffusion mechanism to generate motion conditioned on text or action. Consequently, the motion generation from text has seen diverse contributions.

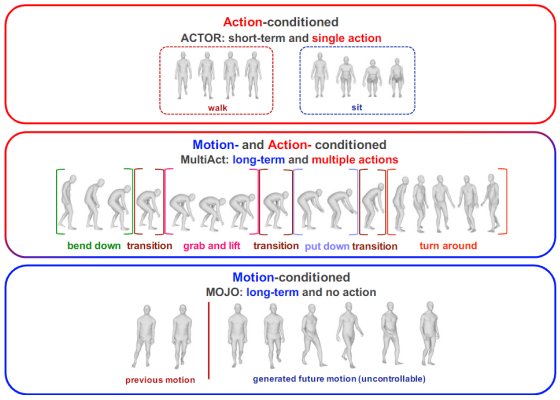
Limitations. However, in contrast to motion generation, very few investigations delve into the inverse process: *motion captioning*. Among the approaches proposed in the literature, a portion of the work by Takano stands out, employing methods that mostly revolve around the Hidden Markov Model (HMM) applied to a specific databases (DB) of similar natures (Takano et al., 2016; Takano and Lee, 2020; Takano et al., 2020).

Moreover, all prior works in the field of motion-language mapping approach this task holistically and do not offer semantic analysis and segmentation of the movement, nor discuss the interpretability and trustworthiness of model predictions. In contrast to the existing narrow focus on motion captioning, our objective is to *generate a semantic segmentation and interpretable description of human motion*. In

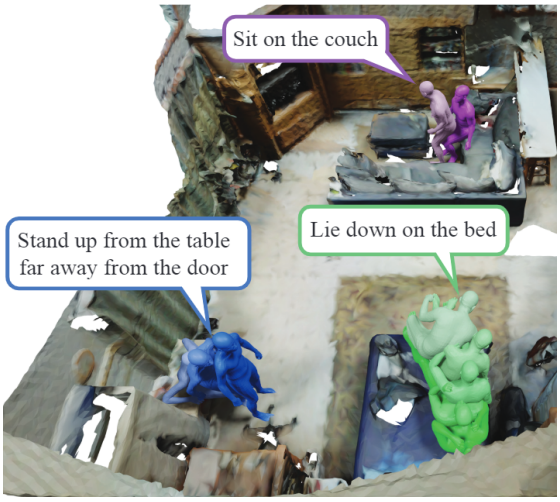
pursuit of this challenging goal, the manuscript tackles text generation through interpretable architectures, designed for the analysis and semantic segmentation of human motion.



(a) Bidirectional motion-language mapping (Plappert et al., 2018).



(b) Description using successive action names (Lee et al., 2023).



(c) Scene-dependent description (Wang et al., 2022).

FIGURE 1.1: Three different forms of motion semanticization.

1.3 Goals

Human movement processing applications require specific tools and methodologies. This involves extracting movement data from video/images or utilizing motion capture, preprocessing the data into standardized formats, and learning rich

movement representations. Before delving into our main contributions, it is essential to comprehensively cover these preliminary aspects and gain firsthand experience. Consequently, the manuscript is organized into two parts. The first one surveys state-of-the-art pose estimation and representation approaches, showcasing applications on real-world tasks. Resting on the rich literature laid out in part I, the part II addresses our main research problem from various complementary perspectives. More precisely:

Part I, reviews the parametric characterization of human motion through pose estimation. Subsequently, it analyzes the performance differences between the estimated data and Motion Capture (MoCap). In this context, we use human pose data for *protective behavior detection* (Olugbade et al., 2021) representing an application in the medical domain, with particular emphasis on model interpretability.

Part II, mainly focuses on solving the task of motion captioning from different perspectives. Starting with an incremental analysis leading to synchronized captioning, we then, propose unsupervised solutions for semantic motion segmentation with relevant metrics for quantitative evaluation. From a second perspective, we focus more on the interpretability on both temporal and spatial levels. As results, we formulate general methodologies to design an interpretable architecture for captioning. These proposed methodologies are demonstrated in the task of motion captioning through multiple specific experiments that aim to evaluate both the quality of the generated text and the interpretability of the model.

1.4 Challenges

Addressing the task of motion captioning, with semantic segmentation and interpretability goals, involves overcoming several challenges. From this perspective, various research questions naturally arise:

How to interpret motion? Is it possible to explain motion automatically as a human would?

Understanding and interpreting human motion pose significant challenges for researchers. Existing methods for interpreting human motion are limited, as they do not always allow for precise and comprehensive interpretation of motion. This limitation may arise from the complexity of human motion and the difficulty in translating the subjective experience of motion interpretation into an automated process. This interpretation can be graded across various levels of granularity, ranging from the overall description of the movement, such as action recognition

(e.g., jumping, walking), to the detailed description of each component involved in the movement: body parts engaged in the motion (*e.g.*, arms, legs), nature of the movement (*e.g.*, bending, placing, lifting), and the manner of execution (*e.g.*, speed). Figure 1.2 provides an example of graduated granularity definition across three levels: 0) recognition of the overall motion; 1) detection of parts involved in executing the motion; 2) identification of the relative movement of each part.

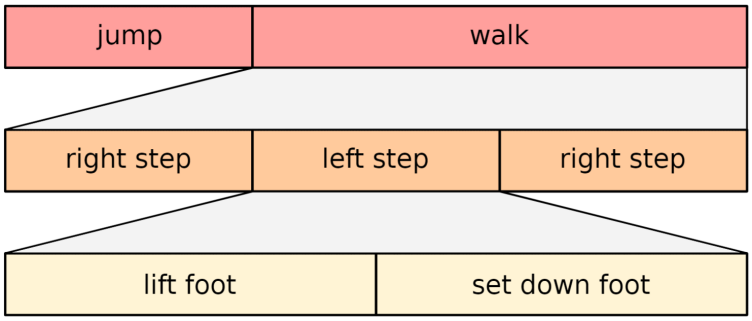


FIGURE 1.2: Example of motion level semanticization (Dreher et al., 2017).

How to understand and decompose motion through its temporal and spatial components ?

Understanding human motion requires an analysis in terms of its temporal and spatial components. Human movements can be decomposed into different parts, such as body segments, joints, linear or circular motions, etc. Temporal decomposition can also include the duration of the movement, speed, acceleration, etc. To understand and decompose human motion, it is necessary to first define what constitutes a decomposition in relation to the semantic analysis of motion. Both forms of decomposition can be defined in our context as follows:

Temporal Decomposition: Decomposition of motion along the temporal axis is akin to a segmentation process where each temporal phase identifies a basic movement (primitive). The set of primitives then composes the overall motion.

Spatial Decomposition: Concerning the spatial dimension, it involves identifying the body parts involved in forming each basic movement and their position in space.

What architecture is best suited to semanticize motion?

The definition of the architecture relies on the definition of motion semanticization. In this manuscript, motion semanticization is defined as the result of associating temporal and spatial decompositions applied to human motion. Thus, the objective of the architecture is to represent the different components of motion in

a way that makes them interpretable. It must also associate these different components with semantic concepts, such as actions and description in natural language.

1.5 Publications

In the following, we list all the research papers written during the thesis:

- Radouane, K., Tchechmedjiev, A., Xu, B., Harispe, S. Comparison of Deep Learning Approaches for Protective Behaviour Detection Under Class Imbalance from MoCap and EMG data. (Winner of AffectMove Challenge 2021). In 9^{th} International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW). (Radouane et al., 2021)

- Radouane, K., Tchechmedjiev, A., Lagarde, J., Ranwez, S. Motion2language, unsupervised learning of synchronized semantic motion segmentation. In Neural Computing and Applications, 2023. (Radouane et al., 2023a)

- Radouane, K., Lagarde, J.,Ranwez, S., Tchechmedjiev, A. Guided Attention for Interpretable Motion Captioning. *Under review*, 2024. (Radouane et al., 2023b)

  This manuscript does not include material from our following paper:

- Radouane, K., Lagarde, J.,Ranwez, S., Tchechmedjiev, A. Transformer with Controlled Attention for Synchronous Motion Captioning. *Under review*, 2024.

1.6 Outline

In the following, we provide details about the challenges that will be addressed in each part and the corresponding chapters:

Part I: Human pose estimation and applications. This part include two chapters:

Chapter 2, dedicated to pose estimation, details various techniques that can be used to quantify and represent human movement. The goal is to determine whether the estimated data can substitute for motion capture through the analysis of 2D pose estimation systems and 3D reconstruction.

Chapter 3 discuss relevant methods for action recognition, with a first contribution applying a deep learning architecture in the medical domain, specifically for protective behavior detection.

Part II: Human motion captioning and segmentation. This part includes three chapters:

Chapter 4 discusses the motion representation methods for motion-language mapping. In this context, we introduce the second contribution of this thesis, which aims to enable synchronized text generation with human action times. This process involves temporal segmentation of movement using attention mechanisms.

Chapter 5 constitutes the third contribution, proposing the quantitative and qualitative analysis of semantic segmentation of human movement inferred from attention weights. To evaluate the performance of this segmentation, several metrics are proposed and compared, followed by the determination of relevant methods for visualizing the synchronization between movement and language. Next, we describe our construction of the first part of the dataset called *Euromov Motion Language Dataset (EMLD)* as a preliminary step towards supervised segmentation.

Chapter 6 details the fourth major contribution, where we propose an interpretable architecture design for motion captioning based on guided spatio-temporal attention mechanisms. Then, we provide tools for interpretability analysis on quantitative and qualitative levels.

Conclusion and perspectives. Chapter 7 provides a final summary of the thesis contributions, potential applications of the proposed methodologies, as well as perspectives for improvement and impact analysis of the present work on other related tasks and domains.

Part I

Human pose estimation and applications

Chapter 2

Human Pose Estimation

2.1 Introduction

In the domain of computer vision and robotics, capturing and comprehending human motion is a fundamental challenge. A popular technique that has revolutionized the field is *human pose estimation*, which involves determining the spatial configuration of a person's body from visual data. By accurately estimating the positions of human articulations, or the positions and orientations of human body parts, pose estimation enables a myriad of applications such as in health care, anomaly detection (Markovitz et al., 2020), and sports analysis. For example, it can be used to track and analyze the movements of athletes in sports training or performance analysis (Badiola-Bengoa and Mendez-Zorrilla, 2021), or to enable gesture-based interfaces in human-computer interaction (Gu et al., 2019), and for human-robot interaction (Óscar G Hernández et al., 2021). Another interesting application domain is the healthcare, as in the work (Lu et al., 2020) authors uses a sequence of poses as input for a classification network to measure the severity of Parkinson disease.

This chapter focuses on a comprehensive introduction to pose estimation, shedding light on its significance, underlying methodologies, and the diverse domains it permeates. Firstly, we present some examples of datasets used for learning human pose estimation under the fundamental principles and challenges associated with pose estimation, including occlusion, varying viewpoints, and the complexities of human body articulation (Section 2.2). Then we present the classical architectures in comparison with more recent advanced approaches, mostly transformer-based (Section 2.3). Subsequently, we discuss the most widely used metrics to assess the quality of pose estimation systems (Section 2.4). Next, we compare the pose estimation performance of previous research (Section 2.5) and discuss the limitations of these previous works in the context of real scenarios (Section 2.6). Finally, we conducted experiments to measure the precision gap between MoCap (Motion Capture) and estimated pose data. This involved running action recognition tasks separately on both data sources and discussing the impact on recognition performance (Section 2.7).

2.2 Example of datasets

There are several publicly available databases that can be used for human pose estimation research and development. These databases contain annotated images

and videos of human poses, which can be used to train and evaluate pose estimation algorithms. Some of the most popular databases include:

COCO (Common Objects in Context) dataset (Lin et al., 2014): The COCO dataset is one of the most widely used datasets for human pose estimation. It contains over 200,000 images with more than 250,000 person instances annotated with keypoints. The dataset covers a diverse range of activities, poses, and occlusions, making it suitable for training robust pose estimation models. Figure 2.1 illustrates the content of this dataset with some images.

MPII Human Pose dataset (Andriluka et al., 2014): The MPII Human Pose dataset consists of around 25,000 images taken from YouTube videos. It provides annotated keypoints for over 40,000 poses in challenging real-world scenarios. The dataset includes a wide variety of activities and viewpoints, making it valuable for evaluating pose estimation algorithms under realistic conditions. The poses are annotated with 16 or 28 key points, depending on the version of the dataset.

Human3.6M Dataset (Ionescu et al., 2014a): The Human3.6M dataset is a large-scale benchmark for 3D human pose estimation. It contains around 3.6 million RGB images captured by several cameras, covering a range of indoor activities. The dataset provides highly accurate 3D annotations for 17 body joints, making it valuable for research on 3D pose estimation.

Leeds Sports Pose dataset (Johnson and Everingham, 2010): This dataset contains over 1,000 images of athletes performing various sports activities (badminton, baseball, gymnastics, parkour, soccer, tennis, and volleyball), with annotated 2D joint positions.

PoseTrack Dataset (Andriluka et al., 2018): The PoseTrack dataset is specifically designed for multi-person pose tracking, which involves estimating poses over consecutive frames. It consists of challenging video sequences with many people engaged in diverse activities. This dataset provides annotations for both pose keypoints and instance-level tracking, enabling research in temporal pose estimation.

These datasets provide a rich resource for developing and evaluating human pose estimation algorithms, and have contributed significantly to the recent progress in this field.

There are different skeletal formats depending on the dataset being used, as shown in Figure 2.2. The format differences make it difficult to transpose an architecture from one dataset to another. Additionally, every dataset comes with different environment challenges and settings (occlusion, multi-person, multi-view).

FIGURE 2.1: Example of COCO Keypoints (Lin et al., 2014).

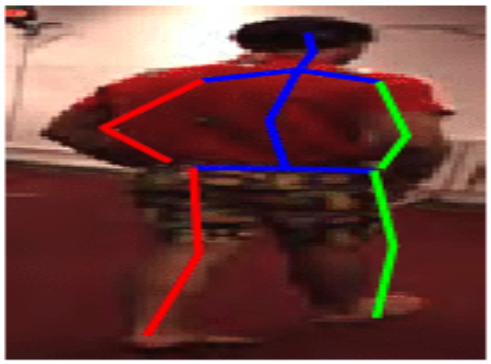
The Human36M (Figure 2.2a) has been widely used for 3D pose estimation, where the LSP (Figure 2.2b) and MPII (Figure 2.2c) are more specific to 2D keypoints detection. The Posetrack (Figure 2.2) is a more challenging real-world dataset that involves pose estimation and tracking of individuals in video scenes.

2.3 Pose estimation techniques

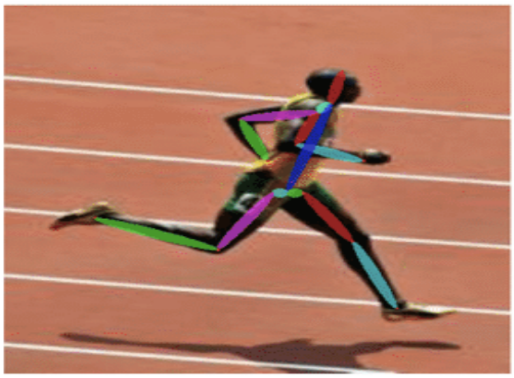
In this section, we will explore different combinations of systems employed for human pose estimation. The process of pose estimation involves addressing various factors such as environmental constraints, intended applications, challenges posed by the database, and the presence of occlusion phenomena. Similar to common categorization practices, we can classify the approaches into two distinct types:

Top-down approach. First the image is divided into regions of interest (ROI) corresponding to individual people, and then pose estimation is performed on each of these regions separately. To extract these regions there are multiple possible methods, one popular technique is to apply the object detection system to locate people in the image with a bounding box. Then, the pose estimator detect the keypoints corresponding to each person in the given region.

Bottom-up approach. Firstly, all the keypoints present in the input image are detected. This is usually done using a convolutional neural network (CNN) trained on a large dataset of annotated human images. Once the keypoints are detected, they are grouped into individual poses based on their spatial relationships (Geng et al., 2021).



(a) Human3.6M format (Ionescu et al., 2014b).



(b) LSP format (Johnson and Everingham, 2010).

(c) MPII format (Andriluka et al., 2014).

(d) PoseTrack format (Andriluka et al., 2018).

FIGURE 2.2: Different pose estimation dataset formats.

Bottom-up vs Top-down. In Figure 2.3 we synthesize the two approaches processes. In comparison, Bottom-up approaches have the advantage of being able to handle complex scenes with multiple people and occlusions, as they do not require prior knowledge about the number or location of people in the image. However, they may be less accurate than top-down approaches on images with a few well-defined individuals. Regarding the time of execution, it varies depending on the specific method used. In general, when there are few people in the scene, top-down approaches may be faster than bottom-up approaches, since they only need to perform pose estimation on a few regions of interest. However, as the number of people in the scene increases, the time required for top-down methods can grow quickly, since the pose estimation should be performed on each region separately. Bottom-up approach can be more efficient in this case for a larger number of individuals in the scene, as they do not rely on object detector, so the pose estimation is done at once for all individuals.

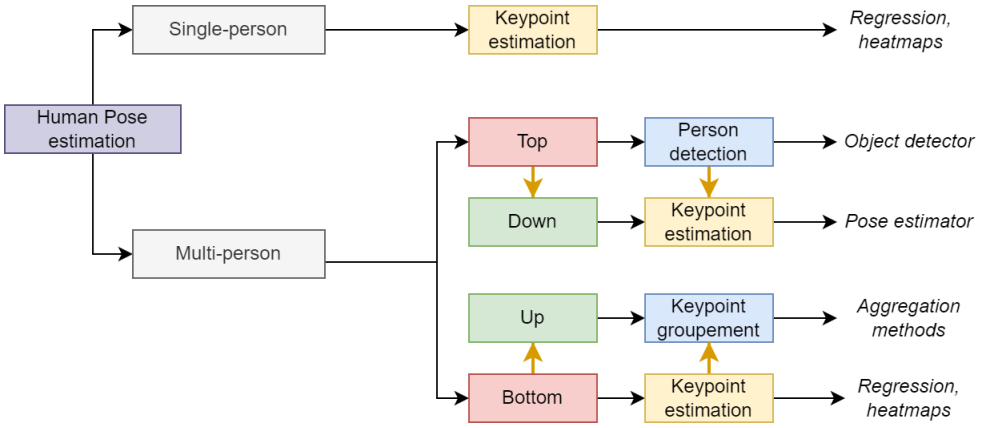


FIGURE 2.3: Overview: Top-down vs. bottom-up approach for HPE.

Single-person vs multi-person. In the context of human poses estimation, there are two possible scenarios. The first one is *Single Person pose estimation*, which represents the simplest case of dealing with the pose estimation problem where only one person is present in the input image. The second is the *Multi-Person pose estimation* where images contain several individuals, this is often the case in real-world scenarios. In human pose estimation for multi-person images, we start to differentiate between bottom-up and top-down approaches. With the top-down approach, the system developed for single-person estimation can be applied to each person present in the image, where the person is identified by the bounding box provided by the chosen object detector. However, multi-person pose estimation is a complex problem that presents challenges such as occlusions, varying body shapes and sizes, and interactions between individuals. Therefore, it requires careful consideration and advanced techniques to achieve accurate results.

In the following discussion, we will explore several architectures proposed for each of the defined approaches, namely top-down and bottom-up, focusing on both single-person and multi-person pose estimation scenarios.

2.3.1 2D Pose estimation

In the **top-down approaches** perspective, the architecture design relies on a combination of object detection and pose estimation components. The object detector is responsible for identifying and localizing relevant objects in an image by producing bounding boxes. These bounding boxes serve as input to the pose estimator, which then operates on each individual object to generate the corresponding keypoints. Regarding the performance influence of each system, a poor detection of

the person can introduce errors in the localization and identification of body parts, leading to inaccurate pose estimations. These errors can propagate throughout the system and affect subsequent stages or downstream tasks that rely on the pose information. To mitigate this, it is crucial to employ an object detector that can effectively and accurately detect people in various environmental conditions and poses. However, during training, the pose estimator can be implicitly trained to handle minor errors produced by the object detector, such as incorrect or imprecise bounding boxes. This is achieved through the use of training data that includes images with a range of different bounding boxes and other sources of variability. The pose estimator is then trained to produce accurate pose estimations despite these variations in the input data. A second solution is to simultaneously learn object detection and pose estimation, as proposed by (He et al., 2017).

In the case of the Faster-RCNN model (Ren et al., 2015), the architecture has two output branches, for each ROI (Region of Interest)[1], the first one is dedicated to class prediction, and the second is for bounding box offset estimation (Figure 2.4). The Mask-RCNN architecture represents the extension of the Faster-RCNN, by adding a third branch (Figure 2.5) for the prediction of segmentation masks. In the context of object segmentation, the mask defines the set of points that constitute the object. As shown in Figure 2.5, the third branch contains 80 [2] resolution masks of 28×28 size.

Drawing an analogy with the object segmentation task, the pose estimation process involves predicting a set of K *masks* specifically for the *person category*, where each mask corresponds to a specific keypoint. Notably, training a model simultaneously on the object detection, segmentation, and pose estimation tasks has demonstrated superior results (He et al., 2017), in terms of average precision (AP). However, when the segmentation task is excluded from the training, the performance tends to degrade. This observation leads to the conclusion that the proposed learning strategy proves to be highly effective in this particular context. Other methods rely on the prediction of heatmaps. In a study by (Chen et al., 2018), the authors proposed a cascaded pyramid network (CPN) architecture, illustrated in Figure 2.6a. This architecture employs two subnetworks: the first one, referred to as *GlobalNet*, focuses on extracting features in the form of heatmaps (as shown in Figure 2.6b). These heatmaps are then utilized for the detection of easily identifiable keypoints using a convolutional filter of dimensions 3×3. Subsequently, the

[1] Image texture based proposal of a rectangle that probably contains an object.
[2] Number of classes, one mask for each category.

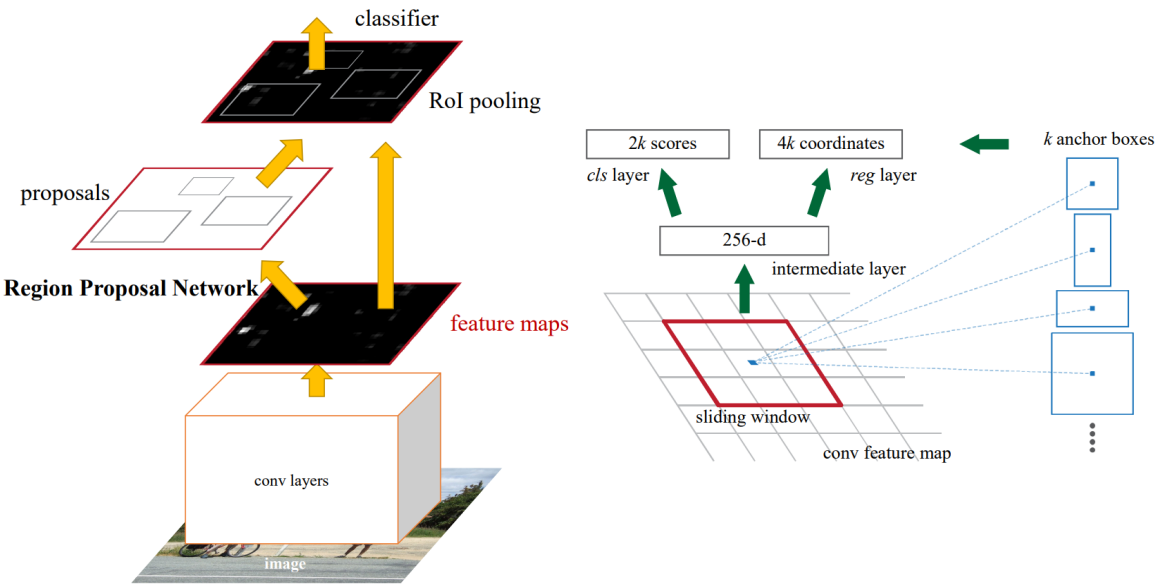


FIGURE 2.4: RPN Regional Proposal Network (RPN) (Ren et al., 2015).

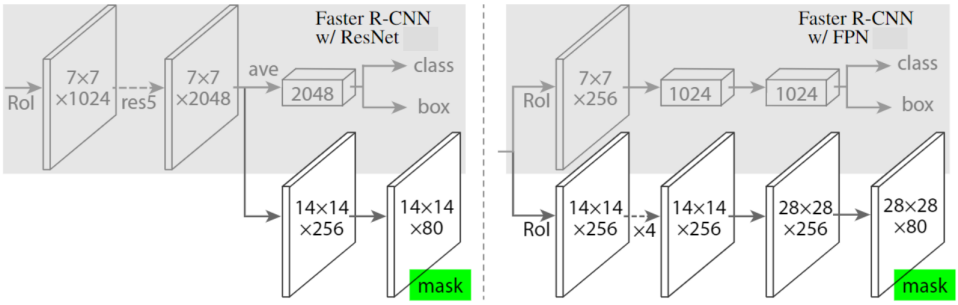
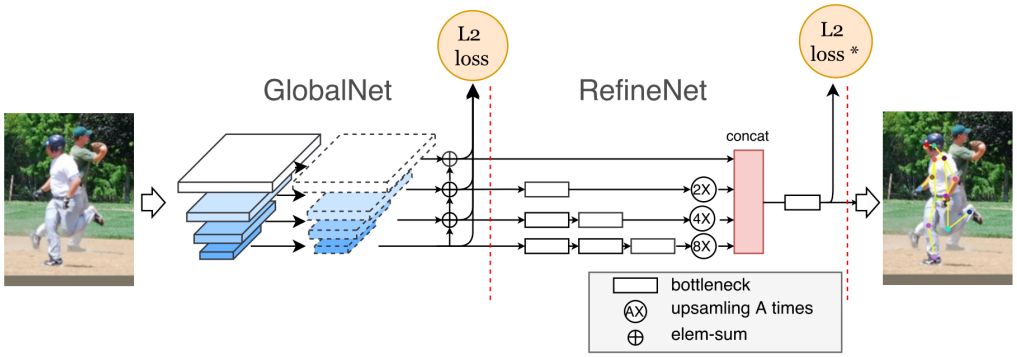


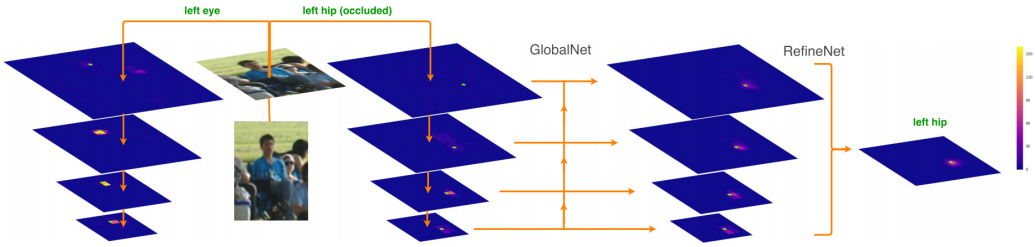
FIGURE 2.5: The head of the MASK-RCNN architecture (He et al., 2017).

hierarchical features generated by *GlobalNet* are passed to the second subnetwork, *RefineNet*, which aims to localize challenging keypoints that may be occluded or difficult to detect. Notably, *RefineNet* is trained specifically on a subset of hard keypoints, enabling to improve the accuracy of localization for these particular keypoints.

Some pose estimation methods incorporate multi-stage architectures to progressively enhance the accuracy of keypoint localization. One such example is illustrated in Figure 2.7. This architecture, convolutional pose machine (CPM) consists of a cascaded design with T stages. At each stage, a series of convolutional and pooling operations are applied to generate *heatmaps* that represent the keypoints. Notably, the predictions in each stage build upon the outputs of the



(a) Architecture of the CPN (*Cascaded Pyramid Network*).



(b) Green dots represent the *keypoints* and red dots are the associated estimations.

FIGURE 2.6: Multistage keypoints estimation.

previous stage, progressively refine the pose estimation results. This iterative refinement process enables the architecture to iteratively enhance the localization and depiction of the keypoints, leading to improved overall performance in pose estimation.

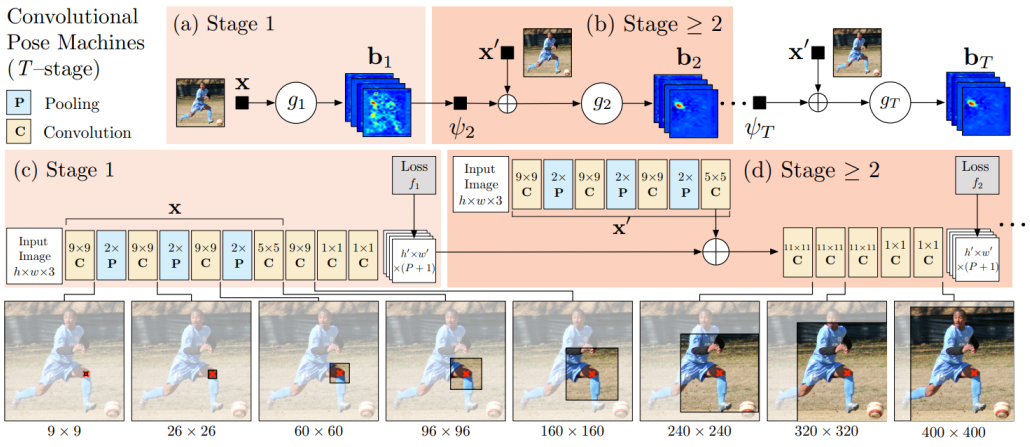


FIGURE 2.7: Convolutional pose Machines (Wei et al., 2016).

Since the introduction of the Transformer by (Vaswani et al., 2017), a wide range

of recent advanced architectures have adopted the transformer framework. Surprisingly, the integration of transformers has led to remarkable performance improvements in various tasks, including pose estimation. One notable example is the Vision Transformer (ViT) architecture proposed by (Xu et al., 2022). While the design of the encoder is not specific to any particular task, the authors present an architecture that can be trained for different vision tasks by adapting only the decoder. As illustrated in Figure 2.8, first the region of interest (ROI) are segmented into patches, each of which is then embedded in a new space along with positional encodings. The sequence of patch embeddings is then processed through a series of L transformer blocks. These operations result in a sequence of relevant encoded feature representations, which are subsequently utilized by the decoder for the specific task at hand. Specifically, they demonstrate high performance on HPE using the COCO dataset (Lin et al., 2014).

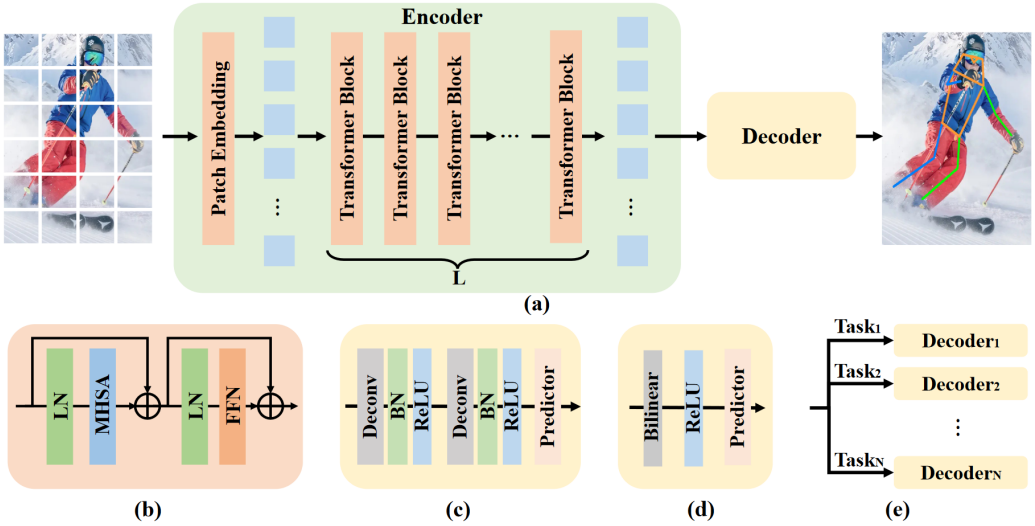


FIGURE 2.8: *ViTPose* framework, (a) decoder blocks, (b), (c) and (d) are possible examples of decoder, while (e) is a custom decoder task-based (Xu et al., 2022).

In the context of **bottom-up approaches**, among the various proposed techniques, `OpenPose`, introduced by (Cao et al., 2019), has emerged as a highly influential and widely used framework. The `OpenPose` model (cf. Figure 2.9) follows a two-step process for pose estimation. Initially, a neural network-based body part detector is employed to classify each pixel as either an element of body part or background. This step enables the identification of key body parts such as the head, shoulders, elbows, wrists, hips, knees, and ankles. In the subsequent step, a graph-based approach is utilized to group the detected body parts and establish

connections between them. The graph representation consists of nodes representing body parts and edges representing the likelihood of connections. By leveraging this graph structure, `OpenPose` determines the optimal set of connections, resulting in the final pose estimation. Additionally, `OpenPose` incorporates a refinement network that generates heatmaps indicating the likelihood of each pixel belonging to a body part or limb. These heatmaps are used to update the initial pose estimation, further enhancing the accuracy of the results.

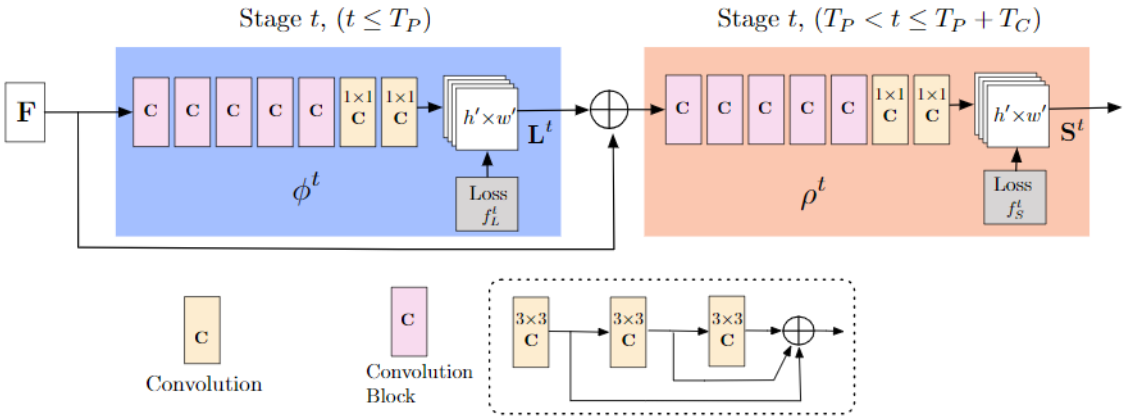


FIGURE 2.9: Open pose CNN-based operations.

The authors (Geng et al., 2021) propose a disentangled keypoint regression that involves learning representations for individual keypoints using adaptive convolutions from partitioned feature maps. As depicted in Figure 2.10, each branch focuses on one keypoint, receiving a specific partition of the feature maps and regressing the 2D offset of that keypoint using a separate 1×1 convolution. For the COCO pose estimation experiments, the feature maps are divided into 17 partitions, with 17 branches dedicated to regressing the 17 keypoints.

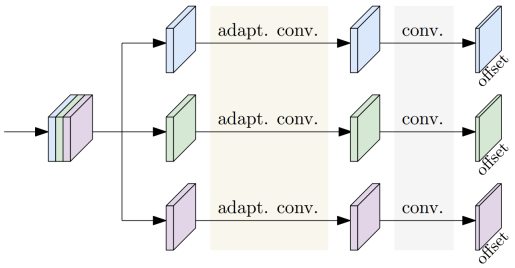


FIGURE 2.10: Disentangled keypoint regression (Geng et al., 2021).

2.3.2 3D pose estimation

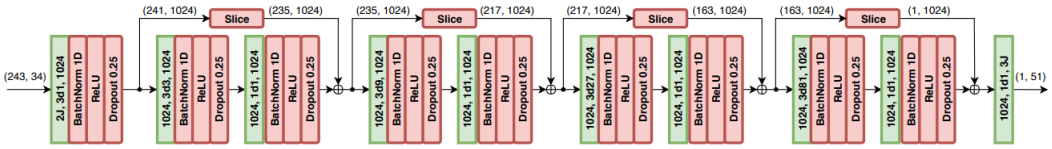
There are various methods used for 3D estimation that differ based on the type of input data, pose representation, and the number of individuals present in the image. In the field of 3D human pose estimation, there are several key approaches and considerations. *Monocular image-based methods* utilize a single image to estimate 3D pose, relying on prior knowledge of human anatomy and motion. *Multiple image-based methods* leverage multiple images or video sequences to estimate 3D pose (Pavllo et al., 2019), employing different algorithms for tracking and fusion of information. Pose representation varies, with some methods using joint angles or body part rotations, and others utilizing volumetric representations (Iskakov et al., 2019). Methods can be designed for single-person or multi-person pose estimation. 3D HPE approaches can be also categorized as top-down or bottom-up, based on whether they start with overall body pose estimation or individual keypoints identification. Another specific aspect in 3D HPE is the 2D-3D lifting, where the architectures are only based on 2D predicted or ground truth keypoints to generate the 3D pose.

2.3.3 Estimation from monocular images

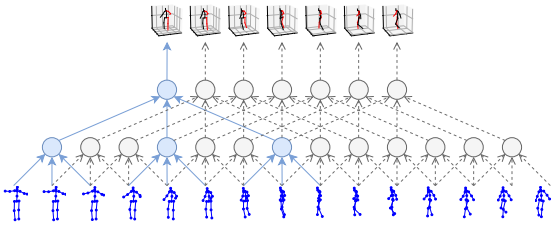
Monocular images are the most commonly acquired type of images in real-world environments, especially outdoors, where there could be only one available view of the scene. The design of 3D reconstruction architectures varies based on the input features used, temporal information, and the nature of the data and constraints. Thus, we can distinguish between methods that rely on 2D keypoints and those that use the input image directly. To estimate human pose from monocular images, deep learning architectures such as those proposed by (Chen et al., 2018), (Newell et al., 2016), and (Tompson et al., 2014) are designed to receive a single image as input at each time point to predict either two-dimensional or three-dimensional coordinates.

As an example, in the paper by (Pavllo et al., 2019), as depicted in Figure 2.11, the authors take advantage of dilated convolutions capability of modeling long-short dependencies with efficient computation. The dilated convolutions are specially useful in the context of estimating the pose in video, which allow the use of temporal information. In the proposed architecture (cf. Figure 2.11a), first the series of 2D keypoints are processed by dilated convolution (kernel size 3×1, dilated factor $d = 1$), followed by a series of operations (BatchNorm(1D)+Relu+Dropout(0.25)).

The rest of architecture design is composed by four blocks, each block is composed of two sub-blocks. The sub-blocks perform similar operations as the input block, with different kernel size and dilation factor. The four blocks are connected through block-outputs and sliced residual connections. Finally, the $4-th$ block outputs are processed by a convolution layer, producing 3D joints learned by regression (MSE loss). Note that each 3D joint (1,17 $\times$ 3) uses 243 key-frames (243,17 $\times$ 2) for optimal performance. Consequently, a drawback of this approach is that it is time-consuming and limited to a single person. If extended to multiple individuals, it would require tracking 243 frames for high precision. As a result, the execution time of each subtask accumulates, leading to a significant delay in the system runtime. This approach is unsuitable for real-time applications, given that non-causal convolutions rely on future and past pose information for 3D reconstruction, it further worsens this problem.



(a) Videopose3D architecture.



(b) Dilated convolutions for 2D-3D pose regression.

FIGURE 2.11: VideoPose3D: architecture based on dilated convolutions (Pavllo et al., 2019).

In contrast, alternative works such as the one proposed by (Cheng et al., 2020) aim to address the issue of articulation point occlusion. This problem arises when certain keypoints or body parts are not visible due to being obstructed by other elements in the image.

The previous described methods rely on 2D generated keypoints. Another set of methods exploits the image directly without an intermediate 2D system (Wang et al., 2014).

For more accurate 3D HPE several methods consider multi-view 2D keypoints

to efficiently construct the 3D pose. However, the multi-view information is typically not available in real-world scenes or can be expensive. Regarding this aspect, (Sun et al., 2020) propose an artificial generation of multiple skeleton views from a **single view**. The concept of this method is presented in Figure 2.12. Mainly, the skeleton input is sent to a `Multi-view Pose Generator` trained to generate different view-projections from the 3D rotated projection. Resulting 2D detections are fed to `GCNs`, which perform a 2D to 3D regression.

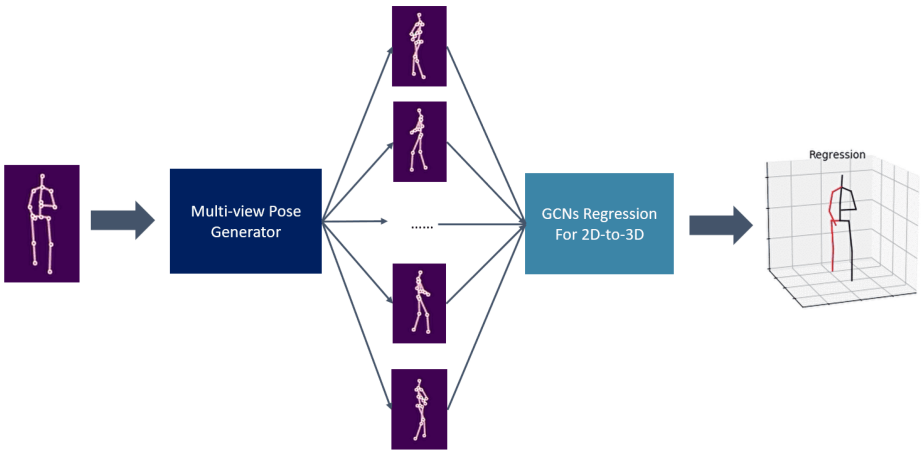


FIGURE 2.12: Multi-view generation architecture (MvPG) and 3D reconstruction (GCNs) (Sun et al., 2020).

Recent work has focused more on transformers, also in the context of 3D HPE. Figure 2.13 depicts an illustrative example. This `MixSTE` model design was proposed by the authors of (Zhang et al., 2022). This work is inspired by the popular idea of performing successive spatial and temporal operations, widely adopted previously with GCNs. In this context, the architecture blocks are aimed at *independent spatio-temporal modelling* of *joint-frame dependencies* based on self-attention mechanism (Vaswani et al., 2017). The HPE is considered as sequence to sequence learning. First, the 2D keypoints are linearly embedded along with a position embedding. Resulting outputs run in a loop of spatial and temporal transformer block for d_l times. Last output-loop is given to a regression head which finally regress the 3D joint positions. The overall architecture is trained end-to-end with temporal consistency-loss *T-Loss* (smooth poses), *WMPJPE-loss*, and *MPJVE-loss*.

In the work presented by (Shan et al., 2023), the authors draw inspiration from the diffusion mechanism and propose an architecture for performing 3D human pose estimation (HPE). The principle of training and inference is illustrated in Figure 2.14a. The proposed architecture aims to learn the reconstruction of 3D poses

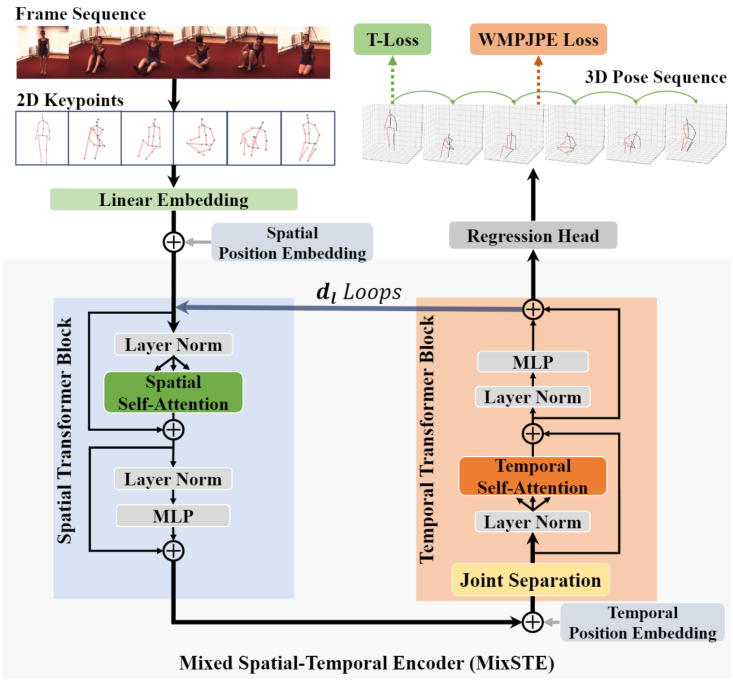
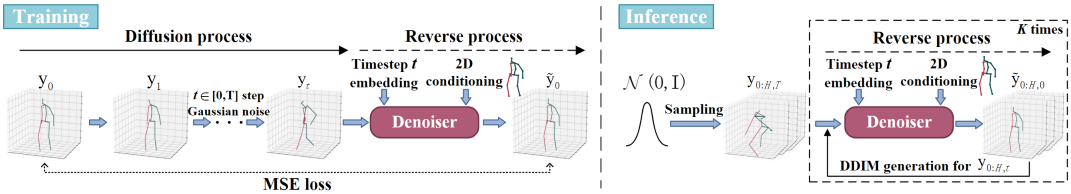


FIGURE 2.13: MixSTE architecture design (Zhang et al., 2022).

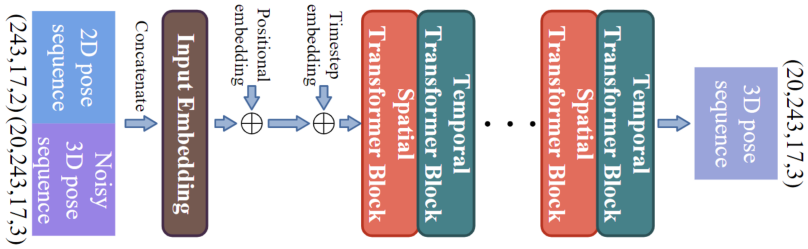
from contaminated 3D pose data, where the denoising process is guided by the utilization of 2D keypoints. Formally, a time step $t \sim \mathcal{U}(0, T)$ is uniformly sampled, where T is the maximum number of time steps. Then, the ground truth 3D pose y_0 is diffused to the corrupted pose y_t by adding t-step independent Gaussian noise $\varepsilon \sim \mathcal{N}(0, I)$. Subsequently, y_t is sent to a denoiser D conditioned on 2D keypoints x and timestep t to reconstruct the denoised 3D pose $\tilde{y}_0$ without noise: $\tilde{y}_0 = D(y_t, x, t)$. The entire framework is supervised by a simple mean squared error (MSE) loss: $L = ||y_0 - \tilde{y}_0||^2$. The denoiser D illustrated in Figure 2.14b is based on the previous presented model `MixSTE` (Zhang et al., 2022).

For inference, since the degraded data approximates a Gaussian distribution after the diffusion process, the same sampling is used to obtain an initial set of H poses $y_{0:H,T}$ by sampling noise from a unit Gaussian. Feasible 3D pose hypotheses $\tilde{y}_{0:H,0}$ are then predicted by passing $y_{0:H,T}$ through the denoiser D . Thereafter, $\tilde{y}_{0:H,0}$ are used to generate the noisy 3D poses $\tilde{y}_{0:H,t'}$ as inputs to the denoiser for the next timestep.

More recently, an advanced architecture has been proposed by (Zhu et al., 2023a), that represents one of the state-of-the-art approaches in the field. Their



(a) Overview of HPE based on diffusion process.



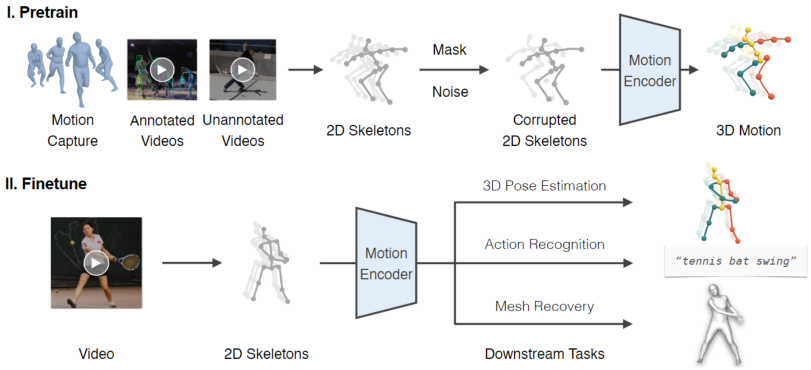
(b) Denoiser architecture where 20, 243, 17 are the number of hypotheses H, frames N , and human joints J in each frame, respectively.

FIGURE 2.14: Diffusion-Based 3D Pose Estimation (D3DP).

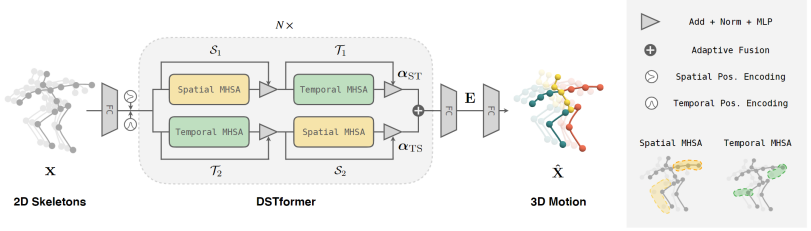
framework addresses the challenge of lifting 2D to 3D mapping through a two-step process, illustrated in Figure 2.15a. Initially, the model undergoes a pretraining phase on corrupted data, followed by fine-tuning on original 2D keypoints. To obtain uncorrupted 2D skeletons from the 3D pose, an orthographic projection is applied. The 2D keypoints data is intentionally corrupted with masking and noise, simulating real-world phenomena such as occlusions, detection failures, and errors commonly encountered in practical data. The resulting corrupted data is then used for pretraining the model, which is subsequently fine-tuned on three specific tasks: i) 3D pose estimation, ii) action recognition, and iii) mesh recovery, leveraging the uncorrupted 2D keypoints. The architecture, depicted in Figure 2.15b, is based on a transformer design with spatio-temporal modeling capabilities. It demonstrates good performance in various domains, including 3D pose estimation using the Human3.6M dataset (Ionescu et al., 2014b), action recognition on the NTU dataset, mesh recovery on Human3.6M and DPW.

2.3.4 Estimation from multiple views

A broad class of approaches for 3D reconstruction is based on multiple synchronized views of the scene. This configuration provides more information-rich data for a more robust and accurate prediction compared to the single view case. The



(a) Framework overview as two processes.



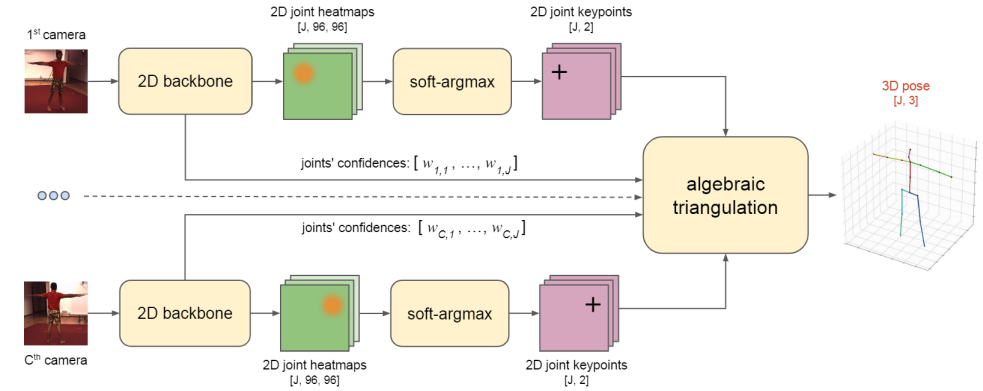
(b) Transformer-based model architecture.

FIGURE 2.15: MotionBERT : Overview of the framework and transformer-based model architecture (Zhu et al., 2023a).

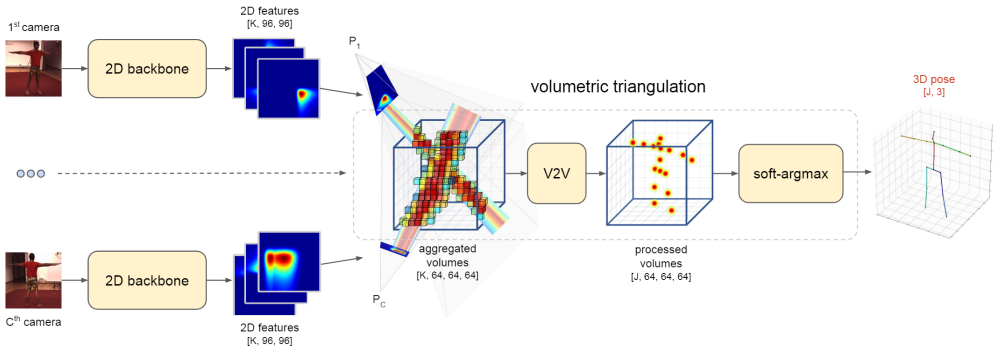
developed architectures exploit simultaneously the images from the different cameras observing the scene. The variations in design from one architecture to another lie in the way the data is processed and the type of neural network employed. An interesting idea based on triangulation was first explored by (Iskakov et al., 2019). The authors proposed two novel solutions (cf. Figure 2.16) for 3D pose estimation, respectively, based on *algebraic* and *volumetric* triangulation. Both methods are based on a *2D backbone* and can be categorized in the set of 2D to 3D lifting approaches.

2D backbone is a CNN-based network that produces 2D joint heatmaps, this block takes an image I_c from a camera c and predict an intermediate heatmaps M_c of size $(K, 96, 96)$. These heatmaps are used by another convolutional network that produce joint-heatmaps H_c of size $(J, 96, 96)$. The implementation of the backbone is based on `ResNet-152`.

First approach presented in Figure 2.16a. The *2D backbone* is trained to predict the joint-heatmaps. Then, *soft-argmax* operation is applied to obtain the 2D positions. The 2D keypoints along with joint-confidence scores are used to solve an over-determined system of linear equations. This algebraic triangulation process



(a) Learnable algebraic triangulation.



(b) Learnable volumetric triangulation.

FIGURE 2.16: Two triangulation-based methods for 3D multi-view pose estimation.

estimates the corresponding 3D coordinates of the keypoints, enabling the reconstruction of the human pose in 3D space.

The second approach depicted by Figure 2.16b, also uses the same *2D backbone*, but in this case the intermediate heatmaps M_c are directly used. The 3D reconstruction is based on volumetric triangulation. First, a $L \times L \times L$ volume grid is discretized by a volumetric cube $V^{coords} \in \mathbb{R}^{64,64,64,3}$ filled with the 3D global coordinates of the voxel centroid (note that $L = 2.5m$). $P_c \in \mathbb{R}^{2\times4}$ denote the projection parameters of camera c. The intermediate maps M_c of size $(K, 96, 96)$ are unprojected into 3D volumes of size $(K, 64, 64, 64)$(cf. Figure 2.16b). These volumes are filled by projecting 2D features maps M_c along projection rays from each camera view perspective. For aggregation of cameras projection, three methods are experimented : a simple i) ***raw summation***, ii) ***normalized weighted sum*** where the weight's contribution d_c of each camera are learned, and iii) ***relaxed version of maximum***. In this last solution, *softmax* is used to produce a distribution for each individual voxel across all cameras. Then, resulting probability distributions **per**

voxel are used as weights for camera-volumes aggregation. Resulting aggregated volumetric maps are then fed as input to 3D-CNN based on the `Voxel-to-Voxel` architecture (`V2V` (Moon et al., 2018)). The methods ii) and iii) give identical performance on the average, the first method i) shows obviously less accuracy.

These triangulation methods with their learnable formulations were adapted by (Chun et al., 2023), given the birth of an updated version presented as *Learnable human Mesh Triangulation*. The overall pipeline, which is very similar to the previous detailed work, is illustrated by Figure 2.17. This new method slightly ameliorates the performance of 3D reconstruction compared to the previous pose-based triangulation (Iskakov et al., 2019).

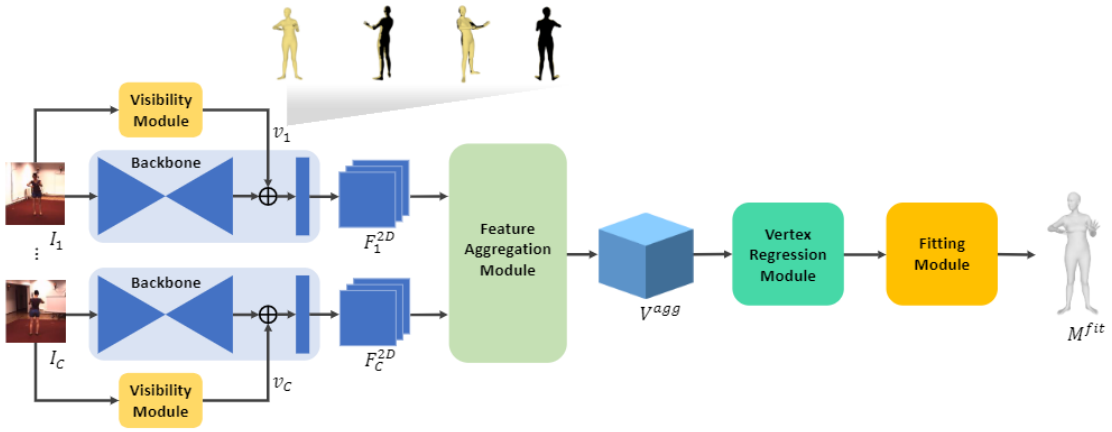
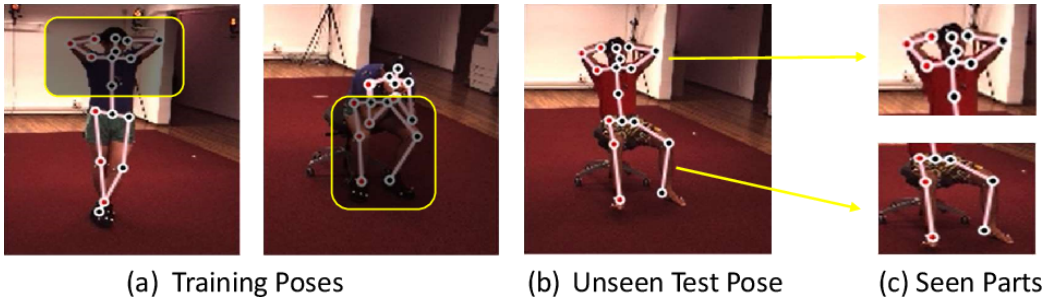


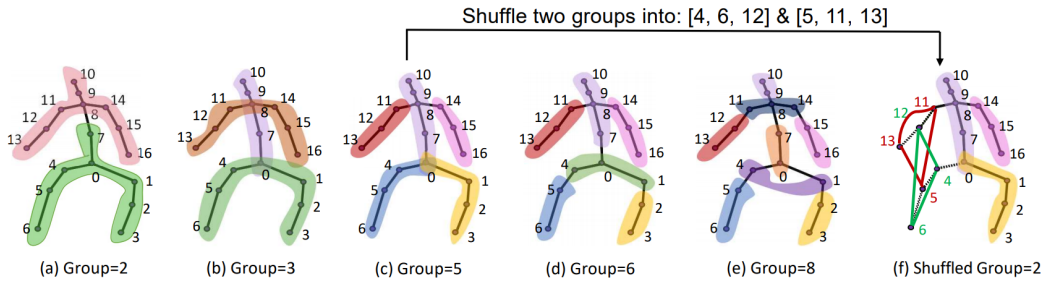
FIGURE 2.17: Overview : human-mesh based learnable triangulation (Chun et al., 2023).

Other sophisticated techniques explore the graph partitioning of skeleton. For instance, in *split-and-recombine approach*, the network proposed by (Zeng et al., 2020), splits the human posture to local groups of articulation points and recombines them at the level of lower dimensions. This technique allows a better generalization to previously unseen images. Figure 2.18a illustrates the idea of processing by local parts of the human body. These local parts can be found partially in the form of a group of points (cf. Figure 2.18a (c)) distributed over several training images. The assembly of these parts allows having new forms of skeleton (cf. Figure 2.18a (b)). The different forms of construction are illustrated by Figure 2.18b. This construction is based on the strength of geometric interdependence between the joints.

In the context of multi-person pose estimation and tracking, this task was efficiently tackled by (Reddy et al., 2021). This paper proposes to perform the 3D pose estimation and person-tracking in a multitask learning fashion. The framework



(a) Visualization of the partial groups of the pose (a) and (c). The assembly of these two groups constitutes a pose similar to the skeletons of an unseen image (b).



(b) Division of the skeleton into a local group set.

FIGURE 2.18: Split design and resulting groups (Zeng et al., 2020).

involves the person detection, 3D joint heatmaps prediction, and person tracking via `4D-CNN`. The overall framework is illustrated in Figure 2.19.

In order to enable a comprehensive assessment of the quantitative performance of both 2D and 3D human pose estimation (HPE) methods, we will introduce the respective metrics for each case in the upcoming section.

2.4 Measures and performance analysis of HPE techniques

As for any system, HPE has to be evaluated through the quality of its response to the problem given and the processing time. The overall performance of a pose estimation system depends on the performance of its components.

Several metrics have been proposed for the evaluation of the performance of human posture estimation, we distinguish two cases: two-dimensional (2D) and three-dimensional (3D) coordinates.

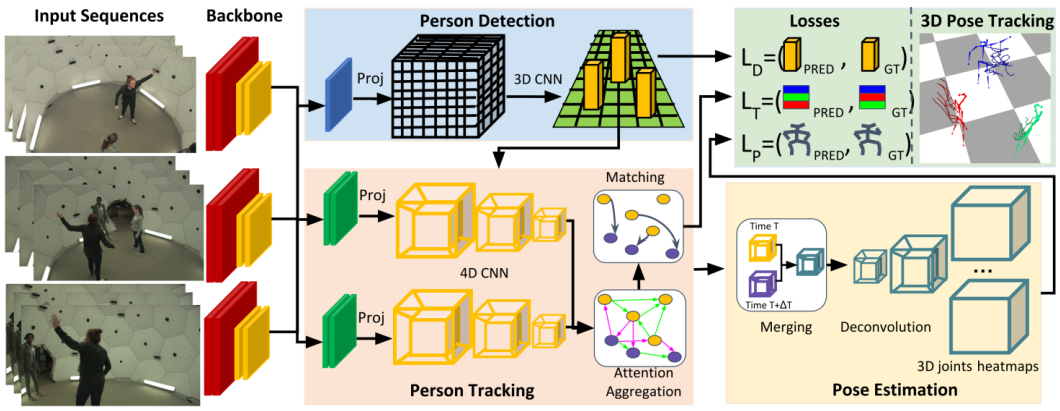


FIGURE 2.19: TESSE-TRACK : Multitask learning for multi-person pose estimation and tracking.

2.4.1 Metrics for 2D estimation

Average Precision & Recall. The evaluation of the detection of *keypoints* proposed by COCO[3] was inspired by the technique used in the case of object detection, which is based on the calculation of the *IoU (Intersection over Union)*. Similarly, the average precision and recall (*AP*,*AR*) are computed based on the *OKS (Object Keypoints Similarity)* :

$$OKS = \frac{\sum_i exp(\frac{-d_i^2}{2s^2k_i^2})\delta(v_i > 0)}{\sum_i \delta(v_i > 0)} \tag{2.1}$$

where d_i is the Euclidean distance between the target and the coordinate of the predicted keypoint. The parameters s, v_i represent respectively the scale of the object and the visibility index (*keypoint* not labeled $v_i = 0$, labeled and not visible $v_i = 1$, labeled and visible $v_i = 2$). The function $\delta(.)$ returns 1 if the condition is verified and 0 otherwise. Perfect predictions will have $OKS = 1$ and predictions for which the distances d_i are large in front of the values $s.k_i$ will have an $OKS \simeq 0$.

Figure 2.20 illustrates the correspondence between the constant k_i for each *keypoint* and the associated tolerance diameter (green disk) for each of these points. The diameters are tight for low k_i which means less error tolerance (e.g. eyes, nose), on the contrary for other k_i values the acceptable error margin is more important (hips, ankles).

Percentage of Detected Joints – PDJ. A joint is considered correctly detected if the distance between this predicted point $\hat{J}_i$ and the target point J_i is less than

[3] https://cocodataset.org/#keypoints-eval

Keypoint	k_i
hips	0.107
ankles	0.089
knees	0.087
shoulders	0.079
elbows	0.072
wrists	0.062
ears	0.035
nose	0.026
eyes	0.025

FIGURE 2.20: The k_i constants calculated for the skeleton format of the COCO dataset.

a fraction of the reference distance d_r. The distance d_r is the diameter of the *torso*, but can also be considered as the length of the diagonal of the person's bounding box. The PDJ metric is computed using Equation 2.2, where N_s is the number of *keypoints*.

$$PDJ = \frac{1}{N_s} \sum_{i=1}^{i=N_s} \delta(\|J_i - \hat{J}_i\| \leq \alpha d_r) \tag{2.2}$$

2.4.2 Metrics for 3D estimation

Several metrics have been proposed in the literature to evaluate the quality of 3D coordinate prediction. The metric often used in model comparison is MPJPE (*Mean Per Joint Position Error*) using the same notation as (Ionescu et al., 2014b):

$$\mathrm{E}(f,S) = \frac{1}{N_s} \sum_{i=1}^{N_s} \|m_{f,S}^{(fr)}(i) - m_{gt,S}^{(fr)}(i)\|_2 \tag{2.3}$$

where $m_{f,S}^{(fr)}(i)$ denotes the 3D coordinates of the i^{th} *join* of the S skeleton predicted by the f detector on the frame fr and N_s is the number of joins. The global error MPJPE represents the average of the $\mathrm{E}(f,S)$ over all images in the dataset.

The equation 2.3 for MPJPE measure can be rewritten in a more explicit way for the set of images by:

$$\mathrm{MPJPE} = \frac{1}{T}\frac{1}{N} \sum_{t=1}^{T} \sum_{i=1}^{N} \|(J_i^t - J_{root}^t) - (\hat{J}_i^t - \hat{J}_{root}^t)\|_2 \tag{2.4}$$

This equation 2.4 aims to measure the error in relative coordinates w.r.t the root joint between the targets and associated predictions.

PA-MPJPE. This metric computes the MPJPE after the alignment of the predicted skeleton with the target skeleton (*ground thruth*) using the *Procrustes analysis* (PA) method. This technique consists in applying a composition of transformations: *translation*, *rotation* and *scaling* to ensure the optimal correspondence between the predicted pose and the target pose.

Percentage of Correct Parts – PCP. This metric takes into account the length scale of the human body parts:

$$\frac{\|s_n - \hat{s_n}\| + \|e_n - \hat{e}_n\|}{2} \leq \alpha \|s_n - e_n\| \tag{2.5}$$

s_n and e_n define the points limiting the part n of the human skeleton, $\hat{s}_n$ and $\hat{e}_n$ are respectively the predicted points associated to these two points. A part n is considered correctly detected if it verifies the inequality 2.5 . Thus, the PCP is computed by the following expression:

$$PCP = \frac{Number\ of\ parts\ \boldsymbol{n}\ correctly\ detected}{Total\ number\ of\ parts\ \boldsymbol{n}} \tag{2.6}$$

Although it takes into account the scale aspect of the person, this error formulation penalizes members of small lengths that are difficult to detect.

Percentage of Correct Key-points (3D-PCK). A point is considered correctly detected if the Euclidean distance between the 3D predicted joint $\hat{J}_i$ and the target joint J_i is lower than a threshold θ. The definition of this threshold varies from one dataset to another. This metric is commonly employed to assess the accuracy of 3D pose estimation in the absolute coordinate system.

$$3DPCK = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{j=1}^{N}\delta\left(\|\hat{\mathbf{J}}_i^t - \mathbf{J}_i^t\|_2 \leq \theta\right) \tag{2.7}$$

2.5 Evaluation of performance in prior studies

After given details about recent techniques used for HPE in section 2.3 and defining metrics in section 2.4, we summarize the performances of previously presented 2D pose estimation systems in Table 2.1 and for 3D reconstruction in Table 2.2.

In the case of 2D estimation, we observe that `ViTPose`, which is transformer-based, outperforms the other systems in terms of AP^{kp} and AR^{kp} on the COCO validation set.

Model	Approach	AP ↑	AP5	AP75	APL	APM	AR
ViTPose (Xu et al., 2022)	Top-Down	81.1	**95.0**	88.2	86.0	77.8	85.6
CPN (Chen et al., 2018)	Top-Down	72.1	91.4	80.0	77.2	78.5	-
Mask-RCNN (He et al., 2017)	Top-Down	63.1	87.3	68.7	71.4	-	-
FasterR-CNN (Ren et al., 2015)	Top-Down	64.4	85.7	70.7	69.8	61.8	-
OpenPose (Cao et al., 2019)	Bottom-Up	64.2	86.2	70.1	68.8	61	-
Disentangled (Geng et al., 2021)	Bottom-Up	72.3	88.3	78.6	68.6	78.6	-

TABLE 2.1: Comparison of performance metrics for different 2D HPE methods.

When considering 3D reconstruction methods that rely on 2D ground truth, multi-view approaches generally outperform others in terms of MPJPE minimization. However, monocular-based approaches hold more practical appeal for real-world applications. Surprisingly, in Table 2.2 highlights the exceptional performance of `MotionBERT`, which achieves better results than other multi-view-based approaches using only monocular view.

Model	MPJPE (mm) ↓	Approach	2D Ground Truth
MotionBERT (Zhu et al., 2023a)	**16.9**	Monocular	Yes
MixSTE (Zhang et al., 2022)	21.6	Monocular	Yes
D3DP (Shan et al., 2023)	19.6	Monocular	Yes
SR-Net (Zeng et al., 2020)	32	Monocular	Yes
MvPG+GCNs (Sun et al., 2020)	35.8	Monocular	Yes
VideoPose3D (Pavllo et al., 2019)	46.8	Monocular	No
LT (Iskakov et al., 2019)	17.7	Multi-View	-
TesseTrack (Reddy et al., 2021)	18.7	Multi-View	No

TABLE 2.2: Performance of 3D Human Pose Estimation Models.

2.6 Addressing real-world challenges: usability of pose estimation frameworks in complex scenes

Transfer to new data. The issue of scaling in estimated pose data, put a significant challenge to the transferability of models. This challenge becomes particularly prominent when applying a 3D pose estimation system to images outside the dataset setting or in different environments, especially in cases of monocular acquisition. Additionally, variations in skeleton format, including the number of joints and spatial definitions, further hinder direct utilization of pre-existing models. For instance, when working with real-world tasks and motion capture data,

the formats often differ, necessitating a complex intermediate step of converting joint formats before effectively utilizing pretrained models for fine-tuning on limited data. One potential solution to address this issue partially is to employ the Skinned Multi-person Linear Model (SMPL) (Loper et al., 2015), which maps to a mesh and subsequently to a standardized joint representation. However, currently, this mapping is only defined for specific datasets, requiring custom mapping for other scenarios, which poses its own challenges. While this approach aids in dataset unification, it does not provide a complete solution when the MoCap joint set of real-world applications differs from the mapping established by SMPL.

Body length measurement. The model's predicted lengths of human body parts often differ from their actual measurements. While the model approximates the skeletal shape based on the image representation of a person, it lacks precise information about the person's real-world measurements beyond this representation. This discrepancy is particularly notable in monocular-based 3D reconstruction. Consequently, this obstacle in pose estimation systems poses a challenge to their implementation in applications that rely on accurate scale information, such as certain areas of robotics. Motion Capture data offer an advantage in addressing this issue, but they come with the drawbacks of being more expensive and time-consuming in the acquisition process.

Relative coordinate system. Previous studies (Zhu et al., 2023a; Iskakov et al., 2019; Zhu et al., 2023a), as detailed previously, have predominantly focused on optimizing the MPJPE which asses the estimation quality of 3D root-relative keypoints. However, real-world applications like video surveillance, anomaly detection, and tracking require the accurate generation of a skeleton in world space. In these applications, the absence of a global trajectory and the fixed center joint (Pelvis) in the generated poses, led to usage limitation. This drawback undermines the effectiveness of such approaches in real-world scenarios.

The gap between scientific literature and real-world requirement was highly detailed by (Kaid et al., 2022). In this paper, the authors propose a solution, that integrates top-down framework based on a modified `GAST-Net` (Liu et al., 2021) and `RootNet` (Moon et al., 2017) networks for multi-person 3D pose estimation from a monocular RGB video in a short execution time. This approach meets a wide requirement by the real-world application *Absolute position, real time, multi-person, monocular view*. The training and evaluation was done on the benchmark `MuPoTS-3D` (Mehta et al., 2018). `GAST-Net` was used for 3D root-relative keypoints estimation and `RootNet` for root trajectory estimation. The framework achieve

FIGURE 2.21: Pelvis alignment vs. Procrustes analysis based alignment Slide 9.

state-of-the-art result with **56.8** in terms of *3D-PCK*.

Absolute coordinate system. Describe more the real-world and give more realistic joint coordinates evolution, is still a very challenging problem for monocular based approach. A single view image lacks of depth information and make the prediction of precise global trajectory very difficult by nature. The benchmark dataset MuPoTS-3D (Mehta et al., 2018) presents an opportunity to enhance models for this challenging task as it closely resembles real-world scenarios.

Metrics evaluation limitation. The evaluation of 3D reconstructed poses is conducted using usually two protocols: Protocol #1, which measures the Mean Per Joint Position Error (MPJPE) relative to the pelvis joint, and Protocol #2, which utilizes the PA-MPJPE metric to evaluate pose independently of global orientation and position. It's important to note that both protocols do not operate in the world absolute coordinate system. The primary motivation behind these protocols is to focus solely on achieving accurate pose shapes, without considering global orientation and position. However, the validity of this criterion needs to be re-evaluated, especially in light of real-world scenarios, as mentioned earlier. Figure 2.21 showcases examples of pelvis alignment and Procrustes alignment, which illustrate the poses used for comparison in the MPJPE evaluation and the PA-MPJPE, respectively.

2.7 MoCap vs pose estimation

We conducted an experiment to compare motion-captured data and estimated keypoints. While motion capture data is known to be highly precise, we sought to determine the added value and cost-effectiveness of using this technology in comparison to estimated keypoints. Furthermore, we explored whether pose estimation could serve as a potential replacement for motion capture in certain applications. For this goal, we propose to evaluate the performance of each of the data using skeleton-based action recognition task. For fairness, we trained both MoCap and estimated data on the same custom model presented in Figure 2.24.

We used the labeled samples provided by (Ofli et al., 2013) in the berkeley Multimodal Human Action Database (MHAD).

2.7.1 Multimodal Human Action Database (MHAD)

The multitude of raw data not being directly exploitable, a multimodal representation is associated to each of the modalities (ex: words –> embeddings, environment –> video). In complex tasks, the use of several modalities is necessary to reach satisfactory performances. Figure 2.22 illustrates a typical example of multimodal data acquisition tools, used in for the construction of the MHAD (*Multimodal Human Action Database*) dataset (Ofli et al., 2013).

Multimodal acquisition system. The two Kinect systems (K_i) are installed diagonally to allow the acquisition of the depth of all elements present in the scene without interference. The acquisition of synchronized images is done through several cameras (C_i) installed on the four angles of view giving a description of the observed action equivalent to 360. The four microphones (M_i) allow the acquisition of the sound waves produced by the action of the person in movement. The data informing on the dynamics of the movement are obtained via 6 accelerometers positioned on the person. The eight optical motion capture systems Impulse (in blue) allow the capture of the 3D positions of the markers (active LED) attached to the person. Thus, the global system allows the acquisition of five modalities (3D coordinates, image, acceleration, audio, depth) giving a multimodal description of what is happening in the scene constituting a relevant and very rich representation of information.

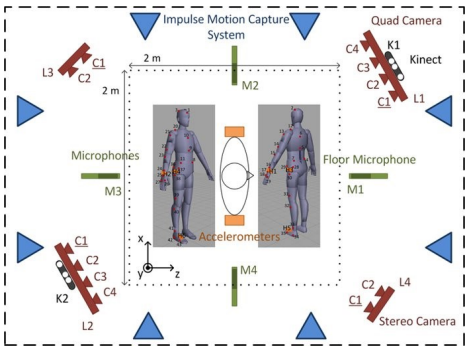


FIGURE 2.22: Multimodal acquisition system of the MHAD. (Ofli et al., 2013)

	Action	Number of Repetitions per Recording	Number of Recordings	Approximate Length per Recording
1	*Jumping in place*	5	5	5 sec
2	*Jumping jacks*	5	5	7 sec
3	*Bending - hands up all the way down*	5	5	12 sec
4	*Punching (boxing)*	5	5	10 sec
5	*Waving - two hands*	5	5	7 sec
6	*Waving - one hand (right)*	5	5	7 sec
7	*Clapping hands*	5	5	5 sec
8	*Throwing a ball*	1	5	3 sec
9	*Sit down then stand up*	5	5	15 sec
10	*Sit down*	1	5	2 sec
11	*Stand up*	1	5	2 sec
12	T-pose	1	1	5 sec

FIGURE 2.23: Action classes statistics. (Ofli et al., 2013)

2.7.2 Experiments

2D poses estimation. We applied the `OpenPose` model to the videos provided by the MHAD dataset, resulting in estimated 2D keypoints for each image. Each image contains 18 keypoints, along with corresponding confidence scores. The 2D keypoints are estimated from each camera view C_i. In order to obtain the ground truth 2D keypoints, we projected the 3D motion capture data onto the plane of each of the 12 cameras (denoted as C_i). To ensure a fair comparison using the same architecture, we reduced the number of 2D keypoints to 18 representative points. This reduction in keypoints does not result in any loss of information and enables us to have a fixed input size for further analysis and evaluation.

3D pose estimation. We use VideoPose (Pavllo et al., 2019) for 3D pose reconstruction. This model is based on 2D to 3D lifting approach, which takes into account a temporal receptive field of 243 frames. In this process, we use the previously estimated keypoints by `OpenPose` as input to generate the corresponding 3D keypoints as illustrated by Figure 2.24.

Model. Our architecture is CNN-based, presented by Figure 2.24 is a simple design comprising three layers of 2D convolutions with different strides and two linear layers. The outputs of the *convolutional* layers are concatenated and passed to the first dense layer. A dropout layer with a rate of 0.25 is applied after the

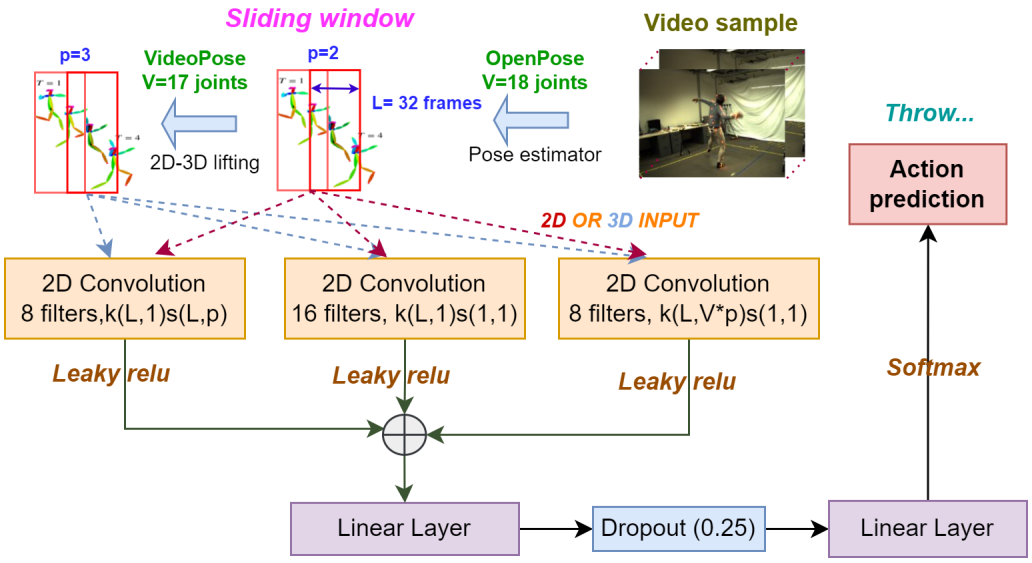


FIGURE 2.24: CNN-model for skeleton based action recognition on MHAD. s and k refer respectively to the stride and kernel size.

first dense layer, and the final dense layer utilizes a *softmax* activation to generate the probability distribution across 11 action classes. The *2D pose-based model* use 2D joints ($p = 2$) and *3D pose-based model* used 3D joints ($p = 3$). For the rest of the architecture, it's similar for both cases. The same architecture is used to train on estimated and MoCap data. This choice enable a fair comparison of based-performances.

Training settings. We use the same split recommended in (Ofli et al., 2013), first 7 subjects for training (S1-S7), validation and last 6 subjects for testing (S8-S12). This result in a cross-subject setting which allow a good assessment of the ability to generalize to a new subject. For training, video samples are splited into chunks of $L = 32$ frames, each chunk is associated with the corresponding action label. In inference the sliding window of length L is applied which allow a *continuous action recognition in real-time*.

Results discussion. As reported in Table 2.3, we see comparable performance using MoCap and 2D pose estimated keypoints only by the use of an average model for both 2D and 3D based on keypoints estimation and there 3D Reconstruction. When analyzing the accuracy on test set, we have 78.46% versus 89.04% on 2D projection of MoCap, and 89.04% versus 90.74% on 3D MoCap data. The model is lightweight, allowing a realtime action recognition. This can be enhanced using

well-designed architecture with spatial-temporal graph convolutions. However, the goal of this study is mainly to evaluate the consequences of difference in quality of the data using same architecture. The pose estimation technique offers a satisfactory approximation of performance compared to motion capture data, albeit with lower accuracy. However, this limitation can be addressed by utilizing a more advanced and precise pose estimator. In summary, human pose estimation serves as a valuable alternative when motion capture is impractical or constrained by factors such as task specifications, environmental conditions, and cost limitations.

# Model parameters	Input data	Subset	Accuracy %
78 123	2D Keypoints (OpenPose)	Test	78.46
		Validation	63.10
		Train	97.13
78 123	2D Projection (MoCap)	Test	***89.04***
		Validation	91.68
		Train	97.68
89 135	3D MoCap	Test	***90.74***
		Validation	84.90
		Train	99.89
100 397	3D Reconstruction (VideoPose)	Test	87.93
		Validation	85.78
		Train	99.53

TABLE 2.3: Recognition rate results with different CNN models on various inputs of MHAD.

2.8 Conclusion

In this chapter, we have reviewed classic and recent approaches of pose estimation, laying the groundwork for further exploration, and understanding of human motion quantification in upcoming chapters. In summary, significant progress has been made in the field of human pose estimation, particularly in the areas of 2D pose estimation and 3D root-relative pose estimation. However, challenges persist when it comes to 3D pose estimation in real-world scenarios. Future research should prioritize addressing these challenges, with a specific focus on settings that closely resemble real-world environments, such as outdoor/indoor detection, real-time estimation, absolute coordinate systems, and single-view scenarios. The

evolving nature of the domain necessitates ongoing efforts to advance the state-of-the-art in human pose estimation for practical applications. In the following chapter, we will explore the methods for encoding human pose data to perform analysis in the context of a real-world medical application.

Chapter 3

Graph representation for human movement classification

3.1 Introduction

This chapter starts with a brief definition of common features utilized in the encoding of human motion (Section 3.2). Numerous architectures have been employed to tackle the diverse array of tasks related to skeleton-based data. In our context, we place particular emphasis on the utilization of spatio-temporal graph modeling for human pose sequences. We specifically review the utilization of graph theory for human pose modeling (Yan et al., 2018). Next, in the second part, we present our first contribution (Radouane et al., 2021), which consists on the application of a graph convolutional network adapted from (Song et al., 2020) to the protective behavior detection task (Section 3.3). We detail the adaptations made to this architecture, that allows to effectively addressing the issue of class imbalance (Section 3.4). Consequently, our model emerged as the winner of the AffectMove 2021 challenge (Olugbade et al., 2021). Moreover, we discuss the architecture interpretability and visualize *implicit* and *explicit learned attention* for investigation of model-decision understanding (Section 3.5). Using these visualizations, we investigate the explainability of model predictions.

3.2 Methods of pose encoding in action recognition

The relevance of the input data is one of the key elements for achieving good performances, satisfactory enough to be applied to new situations brought by real scenes. In the context of action recognition, the state-of-the-art methods to deal with the action recognition task can be divided globally into two types of approaches. The first type is based on the natural exploitation of the sequence of images from the video as input to the architecture. The second type is based on using the predicted sequence of poses for a person tracked in the video.

CNN-based model. The convolutional architecture was mostly recognized as one of the most adapted operation design for tasks based on images as input (*e.g.*, image classification, video captioning...). However, a lot of attempts to use the convolution operations in other different tasks was studied in different field. Where the pose can be modelled as an image, the use of CNN is to become possible based on this new motion representation (Ali et al., 2023).

Graph-based model. This example of designs are more involved for the GCN motion representation, which open the road for a variety of methods to be applied, starting with the well-known model ST-GCN (Yan et al., 2018).

Next, we will present relevant methods used in skeleton-based action recognition in detail, starting with simple yet effective feature encoding (Section 3.2.1) and features learned through graph modeling (Section 3.2.2). We will thus have set the background knowledge to understanding the ResGCN architecture (Song et al., 2020). We employ this architecture for protective behavior detection task proposed in the AffectMove 2021 challenge (Olugbade et al., 2021) (Section 3.3).

3.2.1 Simple motion characteristics

Simple approaches are based on the direct utilization of raw pose data. In this straightforward method, the joint coordinates are used to construct time series and estimate the current action. However, this direct representation lacks discriminatory information and fails to leverage the specific geometry of the skeleton. Alternatively, more commonly employed, and effective features are computed manually, incorporating measures of invariance. In the study presented by (Yang et al., 2019), authors propose the utilization of JCD features (Joint Collection distances) along with measurements. The JCD matrix represents the set of Euclidean distances between pairwise skeleton joint points.

$$JCD = \begin{bmatrix} \|\overrightarrow{J_1 J_1}\| & & & \\ & \ddots & & \\ \|\overrightarrow{J_2 J_1}\| & & & \\ \vdots & \ddots & & \\ \|\overrightarrow{J_N J_1}\| & \dots & \|\overrightarrow{J_N J_{N-1}}\| & \|\overrightarrow{J_N J_N}\| \end{bmatrix} \tag{3.1}$$

Where N is the number of joints specified by the skeleton format. J_i is the 2D or 3D coordinate of the joint point i. As JCD is a symmetric matrix, just the lower part is used, without the diagonal since the distances J_{ii} are zeros. This conserved quantity, after flattening, provides a characteristic vector of size $C_N^2 = N(N-1)/2$. The Euclidean distance, being a metric independent of the coordinate system, provide an invariant information w.r.t. the viewing angle of the scene. This invariant property is very useful in many tasks. Particularly in the context of action recognition,

this property allows less variance in the input, which ameliorates the generalization across multiple data. On the other hand, the *JCD* does not contain information about the trajectory and dynamic of motion, which is necessary for some types of actions, that depend on the global motion. Thus, information-complementary features are used, such as velocity features defined by the Equation (3.2). Where $S^k = \{J_1^k, J_2^k, \ldots, J_N^k\}$ denotes the set of joint-coordinates of the *frame k*.

$$\begin{aligned} M_{Slow}^k &= S^{k+1} - S^k, k \in \{1, 2, \ldots, K-1\} \\ M_{Fast}^k &= S^{k+2} - S^k, k \in \{1, 3, \ldots, K-2\} \end{aligned} \tag{3.2}$$

This second velocity feature is position-invariant information about the skeleton and implicitly encodes trajectory information. For example, the `DDNet` model *(Double-Features Double-Motion Network)* developed by (Yang et al., 2019) uses as input the features computed by the Equations (3.1) and (3.2) to recognize the associated action. This type of features has also been used in other applications, such as for the estimation of the severity of Parkinson's disease, as illustrated by Figure 3.1. This architecture consists in tracking an object (patient) in the video, then generating a 3D skeleton. On the basis of this skeleton, the previous described characteristics *JCD* and *Velocity features* are computed.

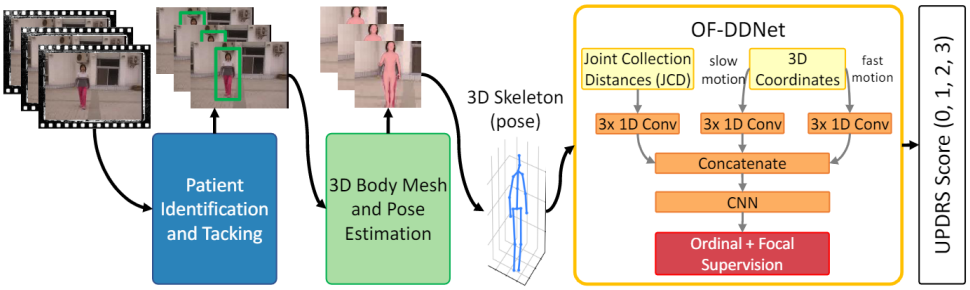


FIGURE 3.1: Architecture OF-DDNET

Figure 3.2 shows the degree of severity of Parkinson's disease and its correspondence in terms of pose sequences. The set of gait skeletons have been reduced to a central fixed point (*root joint or hip center*).

Additionally, Figure 3.3 describes additional features that are widely used, simple characteristics but very useful, such as bones length and joint angles.

A drawback of this kind of approaches is the need for pre-computation of features from the skeleton data for each frame. This additional step can introduce a considerable amount of computation time, particularly when utilizing features like *JCD*. In contrast, the direct exploitation of the raw skeleton data is more efficient

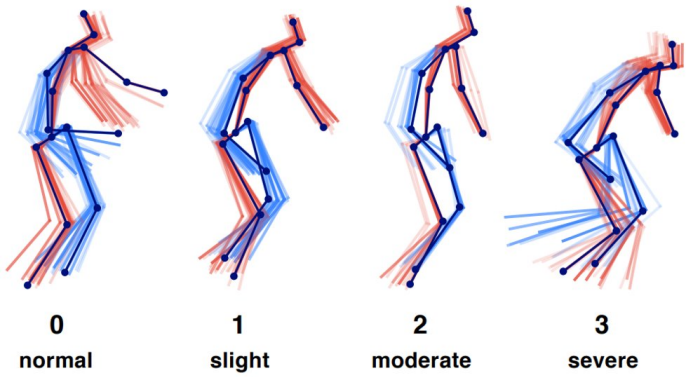


FIGURE 3.2: The four levels of severity of Parkinson's disease described by sequences of the skeleton (Lu et al., 2020).

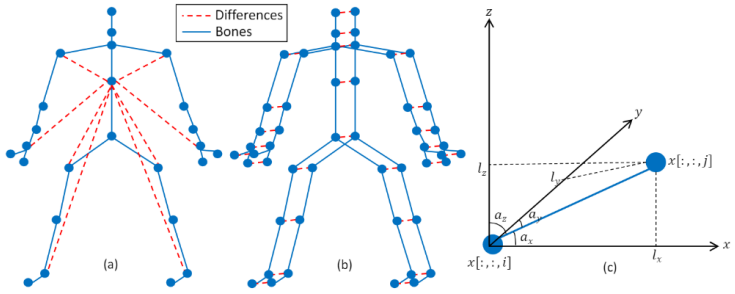


FIGURE 3.3: The demonstration of input data: (a) is the relative positions, (b) is the motion velocities, and (c) demonstrates the 3D lengths and the 3D angles of a bone. (Song et al., 2020)

but may yield less discriminative information in terms of capturing the nuanced parameters of the movement.

3.2.2 Graph modeling

The graph representation is another widely employed approach for modeling the human skeleton. To begin with, we will explore fundamental concepts and formal theories related to graphs. Next, we will explain how this theory can be applied to learn the graph representation of a sequence of human poses. Afterward, we will dive into the process of translating these theoretical concepts into practical implementation, enabling to effectively model the graph representation of the human body's poses over time.

Definition of a graph. A graph is given by a pair $G = (V, E)$ where $V = \{v_i / i \in \{1, \ldots, N\}\}$ is the set of nodes (or vertices) with N the number of nodes and E is a set of paired vertices; $E \subseteq V \times V$. Figure 3.4 illustrates a simple directed graph.

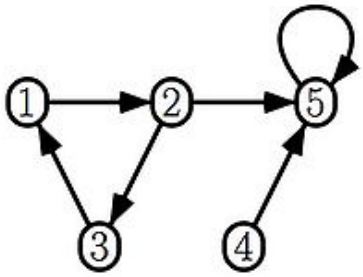


FIGURE 3.4: Example of a directed graph.

A graph can also be defined by a square matrix, named the adjacency matrix A, in which the coefficients are defined by :

$$A_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Using this notation, the graph of Figure 3.4 may be represented by:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Another important concept that should be included for normalizing features is the degree matrix D, which we will use subsequently. The definition of the coefficients of this matrix is given by:

$$d_{i,j} = \begin{cases} deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

Where $deg(v_i)$ is the number of edges connected to the nodes v_i (the loop is counted as a double value).

In order to extract relevant features while preserving the graph structure information, the convolution graph operation was introduced by (Kipf and Welling,

2016) as an extension of convolutional neural networks (CNNs) which are primarily designed for image data (cf. Figure 3.5). The Equation 3.5 gives the convolutional operation on a graph given by the adjacent and degree matrix, denoted respectively as A and D. This extension required redefining several key concepts used in convolutional computations, primarily due to the inherent differences between the topology of a general graph and the pixel grid representation of an image in 2D Euclidean space. This distinction is clearly illustrated in Figure 3.5.

$$H^{(l+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{3.5}$$

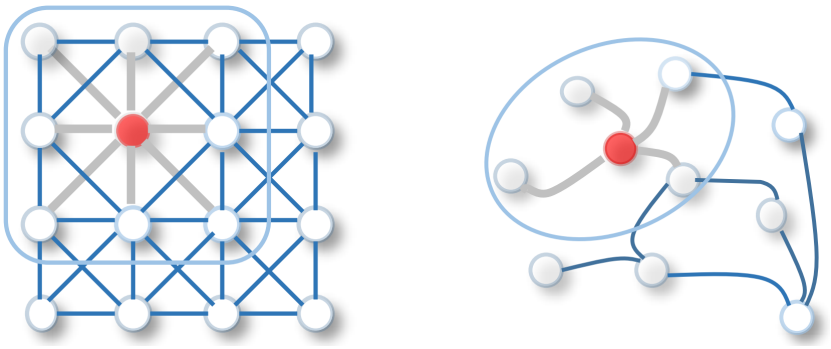


FIGURE 3.5: Comparison of convolution on an image and convolution on a graph (Wu et al., 2019).

Neighborhood node definition. In the case of images, the neighborhood N_i is naturally defined as a rectangular grid centered on pixel i. The dimensions of this grid are fixed and equal to the dimensions of the kernel w. However, in the context of graphs, the nodes do not constitute a naturally ordered set like pixels. Therefore, when applying convolution operations to graphs, it becomes necessary to redefine the concept of a node's neighborhood. The convolution operations applied in a CNN network is done through the Equation 3.6 :

$$h_i^{(l+1)} = \sum_{j \in N_i} < w_j^l, h_j^l > \tag{3.6}$$

Where $i, j \in [\![1, n]\!]^2$ and N_i is the set of points j that are in the neighborhood of point i. We use the symbol l to denote the layer index in the CNN architecture. For an image, h_i^0 represents the intensity values that define the color of a pixel at a position i. In the case of RGB images, $h_i^0 \in \mathbb{R}^3$, and for grayscale images, $h_i^0 \in \mathbb{R}$. In the context of a graph, this representation will be replaced by a quantitative

characterization of each node. Particularly, $l = 0$ refers to the input information level. The other outputs at each layer are computed by the recursive formula 3.6.

Spatial-temporal graph. Previously, we defined the fundamental concepts of graph representation without considering the temporal dimension, which is essential in the task of action recognition. The human skeleton poses a natural graph representation based on human joint articulation. Thus, when considering the temporal evolution of actions, the sequence of human skeleton can be modeled as a temporal graph. The specific formulation of this modeling, applied to the skeleton, was introduced by (Yan et al., 2018). Figure 3.6 illustrates the temporal graph operations.

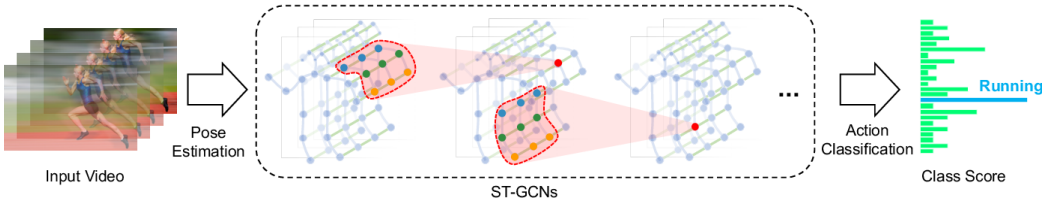


FIGURE 3.6: ST-GCN (Yan et al., 2018)

This modeling is more suited to the nature of skeleton data, allowing for the simultaneous exploitation of node information (*e.g. keypoints*) and natural skeleton structure. In addition to employing other techniques for grouping joints over time to enrich the temporal information, this technique enables a more semantic *embedding* of coordinate vectors, facilitating the recognition of actions.

Noting the importance of data modeling in the form of graphs, we propose to detail the mathematical modeling associated with this domain.

Mathematical modeling. A more general formulation of the Equation 3.6 is given by the following Equation:

$$f_{out}(x) = \sum_{h=1}^{K} \sum_{w=1}^{K} f_{in}(p(x,h,w)) \cdot \mathbf{w}(h,w) \tag{3.7}$$

$p : Z^2 \times Z^2 \rightarrow Z^2$ is the function that enumerates the neighborhoods of the point x, while Z is chosen as an ordered set. In the simple case of an image [1], we have the set $\{p(x,h,w)/(h,w) \in Z^2\}$ defined by a grid centered around the pixel x. The function f_{in} associates each node with its feature vector. The weight function

[1]The image represents a particular case of graph as defined by the natural positions of the pixels.

$\mathbf{w} : Z^2 \rightarrow \mathbb{R}^c$ is used, where c is the dimension of the vector characterizing a node of the graph.

Weight function. The Kernel has a fixed shape and dimensions, and it is not directly applicable to a graph. A solution has been proposed to counter this problem in (Niepert et al., 2016). It consists in defining a function l for labeling the graph, $l_{t_i} : B(v_{t_i}) \rightarrow \{0, \ldots, K-1\}$. The function l is used to partition the neighborhood $B(v_{t_i})$ of the node v_{t_i} into K fixed subsets. In this context, $B(v_{t_i})$ is defined as the set of nodes v_{t_j} that satisfy the condition $d(v_{t_j}, v_{t_i}) \leq D$. Here, the distance measure $d(x, y)$ represents the number of minimal edges connecting nodes x and y on the graph. The distance measure is calculated by finding the shortest path between the two nodes. The graph is utilized to represent the relationship between the nodes, and the constant value D serves as a threshold for the distance between the nodes. The parameter D is chosen as $D = 1$ in (Yan et al., 2018), which means that the neighborhood is formed by the nodes separated by at most one edge from the central node.

$$\mathbf{w}(v_{t_i}, v_{t_j}) = \mathbf{w}'(l_{t_i}(v_{t_j})) \tag{3.8}$$

The Equation 3.8 defines the distribution of weights on the nodes. Nodes with the same label have identical weights, and the kernel weights are shared over the entire graph during the convolution operation.

Spatial Graph CNN. As demonstrated in (Yan et al., 2018), Equation 3.7 can be reformulated as follows:

$$f_{out}(v_{t_i}) = \sum_{v_{t_j} \in B(v_{t_i})} \frac{1}{Z_{t_i}(v_{t_j})} f_{in}(v_{t_j}) \cdot \mathbf{w}(l_{t_i}(v_{t_j})) \tag{3.9}$$

Spatio-Temporal Graph CNN. In this case, we take into account the temporal dimension t, we extend the definition of the set $B(v_{t_i})$ as follows:

$$B(v_{t_i}) = \{v_{q_j} / d(v_{t_j}, v_{t_i}) \leq K, |q - t| \leq [\Gamma/2]\} \tag{3.10}$$

Where $B(v_{t_i})$ represents the spatial and temporal neighborhood of the node v_{t_i} and the parameter Γ defines the temporal kernel size.

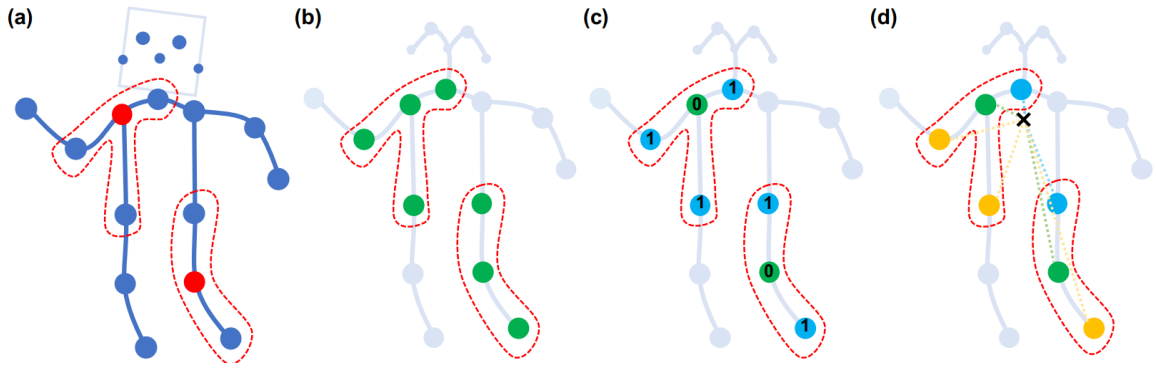


FIGURE 3.7: Human skeleton partitioning configurations Yan et al. (2018).

Partitioning strategies. In order to define the l_{t_i} function of labeling, several types of partitions have been proposed.

- ***Uni-labeling.*** The simplest strategy (cf. Figure 3.7 (b)) consists in assigning the same label to all the nodes of the same neighborhood, formally $K = 1$ and $l_{t_i}(v_{t_j}) = 0 \; \forall i, j \in [\![1, N]\!]$.

- ***Distance partitioning.*** Based on the selected distance threshold D. Figure 3.7 (c) provides an example for $D = 1$. In this case, the neighborhood of each node v_{t_i} is partitioned into two subsets: the singleton v_{t_i} labeled 0, and the subset $B(v_{t_i}) \setminus v_{t_i}$ of nodes labeled 1. This mapping is formalized by the function l_{t_i} as follows:

$$\forall i, j \in [\![1, N]\!] : l_{t_i}(v_{t_j}) = \begin{cases} 0 & \text{if } j = i \\ 1 & \text{otherwise} \end{cases}$$

- ***Spatial configuration partitioning.*** In this case, the partitioning criterion is based on the distance of the nodes from the center of gravity (black cross on Figure 3.7 (d)), and each neighborhood is partitioned into three subsets with the following labeling function, where r_i is the average across the training set of the distances from node v_i to the center of gravity.

$$\forall i, j \in [\![1, N]\!] : l_{t_i}(v_{t_j}) = \begin{cases} 0 & \text{if } r_j = r_i \\ 1 & \text{if } r_j < r_i \\ 2 & \text{otherwise} \end{cases} \tag{3.11}$$

Implementation of the graph convolution operation. The implementation of the concepts described above is done through equivalent matrix formulation. In the single frame, ST-GCN is implemented using a similar formula to (Kipf and Welling, 2016) that is valid for the *Uni-labeling* partitioning strategy:

$$f_{out} = \Lambda^{\frac{-1}{2}}(A+I)\Lambda^{\frac{-1}{2}} f_{in} W \tag{3.12}$$

The degree matrix Λ is used to normalize A. In the spatial-temporal case, firstly, a $1 \times \Gamma$ convolution is performed on the input. The resulting output is then multiplied by the normalized adjacency matrix. In general, to take into account the three strategies, the Equation 3.12 is rewritten as :

$$f_{out} = \sum_j \Lambda_j^{\frac{-1}{2}} A_j \Lambda_j^{\frac{-1}{2}} f_{in} W_j \tag{3.13}$$

Where A_j represents the matrix of adjacent nodes labeled with index j. For example, in the case of distance partitioning strategy $A_0 = I$ and $A_1 = 1$.

3.3 Application in the AffectMove 2021 challenge

The 2021 AffectMove challenge (Olugbade et al., 2021), organized in the context of the EnTimeMent H2020 project in the ACII 2021 conference, offered the opportunity of participating in three challenging tasks, that involve the detection of characteristics of human motion in multi-modal and uni-modal settings. We participated in task 1, entitled *Protective Behavior Detection based on Multi-modal Body Movement Data*.

Definition. A protective behavior is considered as the behavior resulting from a reaction to pain that occurs during the execution of a particular motion.

This first task provided participants with motion capture (MoCap) + Electromyography (EMG) data from the EMOPAIN dataset (Aung et al., 2016) for protective behavior detection across several subjects and tasks, with fixed three-second recordings.

Motivation. Our participation in Task 1 focused on detecting protective behavior in individuals with chronic musculoskeletal pain disorders. Evaluation and treatment of such disorders lack objective criteria, particularly in assessing patient

recovery and pain levels, which often have a significant psychosomatic component. Protective behavior, which exacerbates non-use and serves as an objective marker of pain levels, plays a crucial role in gauging the true extent of pain, beyond self-reported scales. Automatic detection of protective behavior from sensor data has the potential to revolutionize clinical protocols for treating chronic musculoskeletal disorders, offering a valuable tool for reforming current practices. We believe that the development of technology capable of assessing and protecting an individual's behavior could greatly enhance the delivery of personalized therapies.

3.3.1 Dataset description for task 1

- *Input data.* In the dataset of Task 1, each subject and experimental sequence pair represented one record and was materialized as a single file. The MoCap data was encoded in the first 51 columns of the data file, 3D spatial coordinates of 17 joints (cf. Figure 3.8a) and the EMG data (cf. Figure 3.8b) were encoded in the next 4 columns, followed by the action identifier. Each instance was composed of 180 frames (60fps) and the EMG signal was preprocessed (signal envelope), down-sampled and aligned with MoCap frames. The dataset was divided into training, validation, and test sets, with sizes of 5827, 1844, and 2744 samples, respectively. These samples were obtained from individuals with chronic musculoskeletal pain. The split setting is cross-subject, meaning that the subjects in the test set are different from those in the training and validation sets. Table 3.1 present the distribution over classes.

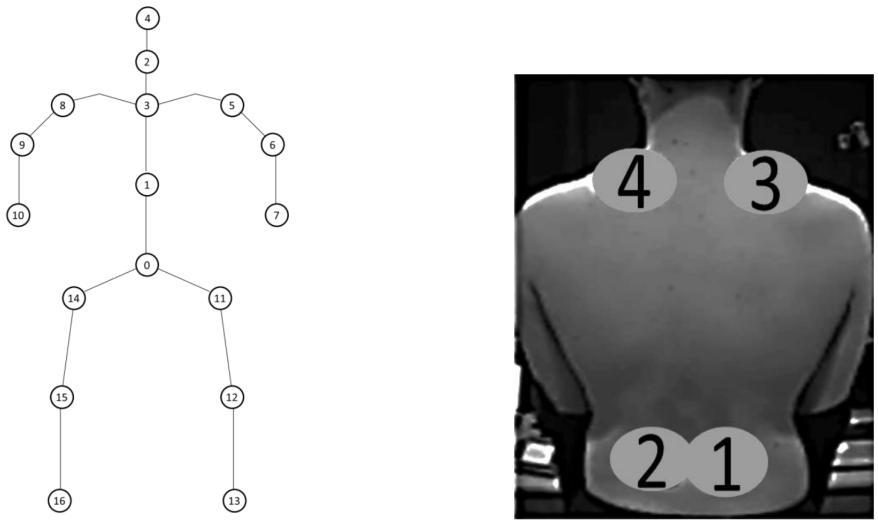
Exercise Type	Stand on one leg	Sit still	Reach forward	Sit-to-stand	Stand-to-sit	Stand still	Bend	Walk
Action label	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

TABLE 3.1: Exercise Types and action labels.

- *Labeling.* The protective behavior label was assigned based on ratings from four clinician raters, with a window being labeled as "present" **(class 1)** if 50% of the window showed at least one protective behavior according to two or more raters. The normal case represents the **class 0**. The windows were segmented based on continuous activity transitions, allowing for the inclusion of multiple activity types within a single window. Frame-level activity labels were provided for each window.

Training set			Validation set		
Protective	**Count**	**Percent**	**Protective**	**Count**	**Percent**
0	4522	77.60%	0	1580	85.68%
1	1305	22.40%	1	264	14.32%

TABLE 3.2: Characteristics and distribution of classes across the train and validation sets.



(a) Skeleton Graph (17 joints).

(b) EMG electrodes positions (1-4).

FIGURE 3.8: The multimodal data structure of the task 1 (MoCap+sEMG).

3.3.2 Architecture for protective behavior detection

In order to predict protective behavior, it is crucial to develop systems that can effectively leverage motion capture (MoCap) or multimodal approaches combining MoCap and electromyography (EMG) data. In our research, we have focused on utilizing deep learning techniques. Our team has introduced three distinct architectures: an *LSTM-based model*, a *Transformer-based model*, and a *GCN-based model*. Notably, the graph convolution-based architecture, which represents my specific contribution, has exhibited good performance on the test set and emerged as the winning solution for Task 1 in the AffectMove challenge. In the following, we will delve into the details of this graph convolution-based approach, training method and discussing the results. In addition, we present a comprehensive visual interpretation of the model's decision-making process by employing both *implicit* and

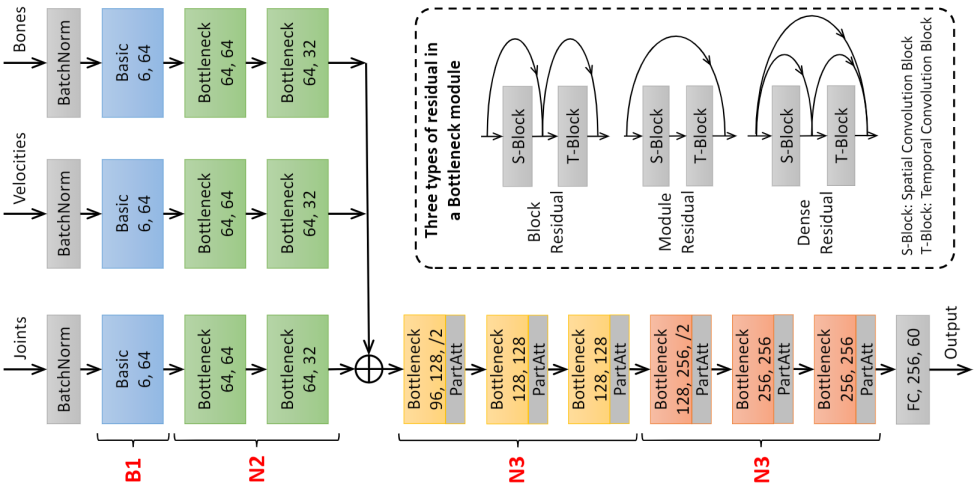


FIGURE 3.9: Part Attention Residual GCN architecture (Song et al., 2020).

explicit attention visualizations. This approach allows for a deeper understanding of how the model reaches its predictions, providing valuable insights into its reasoning and decision-making capabilities.

GCN-based model. For the extraction of spatial-temporal features, our experimentation focused on the use of ResGCN architecture (cf. Figure 3.9), as introduced in the work (Song et al., 2020). This particular architecture was chosen for its incorporation of part-wise attention, which can be advantageous for detecting protective behavior. We specifically consider the fact that certain body parts, such as the lower neck or lower back, are more prominently involved in exhibiting protective behavior, especially in response to pain or potential harm. The part-wise attention can help effectively focuses on these specific body regions, allowing for a more precise and targeted analysis of protective behavior patterns. Another aspect is that attention-based architectures enable visual interpretation.

Multi-Input Branch (MIB). The input features of ResGCN based architecture are derived from various aspects of joint movement. The motion is represented by relative and global joint positions, fast and slow velocities, bone vectors, and joint angles described before (Section 3.2.1). These features are concatenated together, resulting in an input size of **(6, 180, 17, 3)**. Each of the three branches processes one type of features.

ResGCN architecture. The model design is based on a series of spatial and temporal graph operations that are implemented differently for the compounds composing the architecture. Namely, the design construction is based on a *Basic block*, *Bottleneck block* and *Residual connections*. Additionally, in the PA-ResGCN variant, the model uses a *PartAtt block* incorporated in the main stream at each layer. We will give further details about each compound. First, let's define formally the spatial and temporal graph convolutional operations used in this context:

Spatial Graph Convolutional. The spatial GCN operation for each frame t in a skeleton sequence is formulated as:

$$f_{\text{out}} = \sum_{d=0}^{D} W_d f_{\text{in}} \left(\Lambda_d^{-\frac{1}{2}} A_d \Lambda_d^{-\frac{1}{2}} \otimes M_d \right) \tag{3.14}$$

In the given Equation, D represents a predefined maximum graph distance. In our context $D = 1$. The characteristics f_{in} and f_{out} correspond to the input and output feature maps, $\otimes$ denotes element-wise multiplication, A_d denotes the adjacency matrix for pairs of joints with a graph distance of d, and Λ_d is used for normalizing A_d. The learnable parameters are W_d and M_d for convolution operation and to adjust the significance of each edge, respectively.

Temporal Graph Convolutional. For temporal feature extraction, a convolutional layer with a size of $L \times 1$ is employed to capture the contextual information present in neighboring frames and nodes. The value of L is determined as a hyperparameter, indicating the length of the temporal window. In our context $L = 9$. This approach allows for the incorporation of both spatial and temporal aspects in the feature extraction process.

Basic Block. Composed of spatial and temporal blocks, this block performs the following operations:

$$x_{out_S} = Relu(BN(SGCN(x_{in})) + x_{in}) \tag{3.15}$$

$$x_{out_T} = Relu(BN(TGCN(x_{out_S})) + x_{out_S}) \tag{3.16}$$

Where *BN* stands for 2D Batch Normalization and *SGCN*, *TGCN* are respectively the spatial and temporal graph convolution operations defined before in the paragraph 3.2.2. This block is illustrated in Figure 3.9 by a blue rectangle, which also indicates the input and output channel dimensions.

Bottleneck Block. In order to reduce the model size and computational cost, a bottleneck structure was incorporated in the GCN model by the authors of (Song et al., 2020). This structure introduces a hyperparameter called the reduction rate r, which determines the number of channels in the middle layers ($C \rightarrow C/r$).

$$x_{out_1} = Relu(BN(Conv2D(x_{in}))) \tag{3.17}$$

$$x_{out_2} = Relu(BN(SGCN(x_{out_1}))) \tag{3.18}$$

$$x_{out} = Relu(BN(Conv2D(x_{out_2})) + x_{in}) \tag{3.19}$$

Global average pooling 2D. The final convolution output $(N, d_{out}, C/r, V)$ is averaged across the reduced channel with factor r and joint nodes V given an output (N, d_{out}), where N is the batch size and d_{out} the dimension of learned features.

Classification layer. In our context of protective behavior detection (PBD), we have one output followed by a *sigmoid* activation function instead of *softmax*. This sigmoid function outputs the probability p of the presence of PB, where $p \geq 0.5$ implies the positive class (class 1).

Part attention block. The computation of part-level attention, as depicted in Figure 3.10, follows a specific procedure. First, an average pooling operation is applied across all body parts, resulting in a set of pooled outputs. These outputs are then concatenated together. Next, a fully connected layer projects this concatenated output, incorporating a channel reduction factor r to reduce computational costs and the number of architecture weights. Subsequently, the output is re-projected by five layers, each dedicated to a specific skeleton part. This re-projected output is then used to generate the part-level attention distribution through a *softmax* operation. This process enables the model to allocate attention to different body parts and capture their respective contributions to the overall protective behavior assessment. The part-attention block is incorporated after each bottleneck block, this enables leaning important joint features through several layers.

This attention mechanism has been implemented in (Song et al., 2020). A preprocessing of the 3D MoCap data is performed by calculating the motion characteristics (slow motion, fast motion), the calculation methods are similar to those described previously in the subsection 3.2.1.

Figure 3.9 illustrates the `PA-ResGCN` architecture used. This architecture receives as input the features: coordinates of the keypoints, skeleton, velocities, lengths, and angles of the bones.

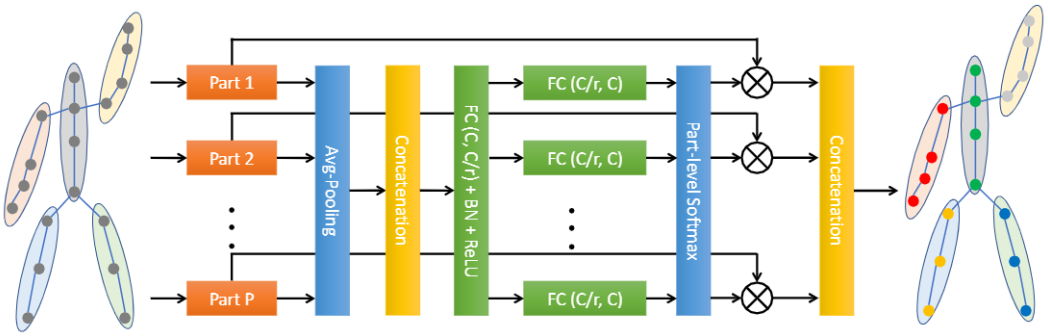


FIGURE 3.10: Attention weights computation (Song et al., 2020).

In the context of the action recognition task, the authors propose different level of attention weights computation, namely the *channelAtt*, *FrameAtt*, *JointAtt* and *PartAtt* blocks. We will detail the last block which led to the best results in their experiments on NTU dataset. As its name indicates, the block consists in applying the attention on each of the five parts of the skeleton (cf. Figure 3.10). Formally, the operation performed by *PartAtt* block is given by the Equation 3.20.

$$\begin{aligned} f_p &= f_{in}(p) \otimes \delta(\theta(pool(f_{in})W)W_p) \\ f_{out} &= Concat(\{f_p | p \in \{1, ..., P\}) \end{aligned} \tag{3.20}$$

where a weight matrix W_p is used for each part, $p \in \{1, \ldots, P\}$ with P is the number of partitions associated to the skeleton ($P = 5$ in this context). W is a shared learnable matrix by all skeleton-parts. $f_{in} \in \mathcal{R}^{N \times C \times T \times V}$ is the feature input map. $pool(.)$ is the function that performs *2D adaptive average pooling*, δ and θ are part-level Softmax and ReLU activation functions. Finally, $f_{out} \in \mathcal{R}^{N \times C \times T \times V}$ is the output feature maps, which represent the concatenation of all *attention enhanced feature maps* denoted by f_p .

3.4 Experiments

This section provides implementation details (Section 3.4.1) and evaluation (Section 3.4.2). Following that, we analyze and compare the performance of the models. Later on, we investigate the interpretability of model decisions (Section 3.5).

3.4.1 Implementation and training

Our implementation. We started from ResGCN (Song et al., 2020), which encodes three main features extracted from skeleton poses: joints position (*local and global*), velocities (*fast and slow*) and bones. We modified the official implementation of ResGCN by using another loss function to address the high imbalance between positive and negative classes. We defined the new graph corresponding to the skeleton format in AffectMove Task 1 (cf. Figure 3.8a), notably the formulation of the graph for the GCN. The proposed weighted reformulation of the binary cross-entropy loss accounts for the unbalancing issue and compensates the imbalance during training.

Multimodal Fusion – EMG and MoCap. Additionally, we proposed a simple multimodal extension that performs a late fusion between ResGCN(-N51) and a convolutional neural network for EMG classification. For the latter, the signals are first filtered with an exponential moving average ($\alpha = 0.2$). The CNN architecture consists of one layer normalization per batch to adjust scale of input EMG data with learnable parameters, followed by one 2D convolutional layer, batch pooling and two linear layers. PReLU activation is used after the convolutional layers and Tanh in between the two linear layers. This architecture serves to extract 128 EMG features, which are then concatenated to the 256 features extracted by ResGCN-N51. Figure 3.11 illustrates the full multimodal architecture.

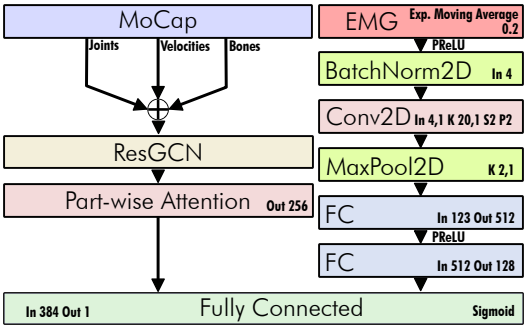


FIGURE 3.11: Architecture of the multimodal PA-ResGCN-N51 + EMG CNN system (Radouane et al., 2021).

Loss reformulation. We use binary cross-entropy with logits along with an added weighing scheme to penalize the majority class. The weights are computed per batch like in (Cui et al., 2019), where we consider $\beta \rightarrow 1$, so that following the

paper's notation, $E_n = n_{s,k}$. The loss l_k for the k^{th} batch is defined as:

$$l_k = \sum_{(s,y_s)\in B_k} w_s[y_s log\sigma(x_s) + (1 - y_s) \cdot log(1 - \sigma(x_s))] \tag{3.21}$$

$$w_s = \frac{1}{n_{s,k}} \qquad n_{s,k} = \begin{cases} n_{-,k} + \epsilon, & \text{if } y_s = 0 \\ n_{+,k} + \epsilon, & \text{otherwise} \end{cases} \tag{3.22}$$

where B_k is the set of training examples of the k^{th} batch, w_s is the weight per sample s where $n_{(-,k)}$ and $n_{(+,k)}$ are respectively the number of negative and positive samples present in the k^{th} batch, ϵ was set to 10^{-6} to avoid division by 0 errors.

Training and optimization schedule. First, we used only the 3D MoCap data to train the PA-ResGCN, particularly the ResGCN with it's variants (B19, N51, PA) where N51 means there are 51 convolutional or FC layers within the model and where PA stands for part Attention. We experimented with two types of optimization approaches: (i) Optimizer based on stochastic gradient descent (SGD) with a Nesterov momentum of 0.9, weight decay of 10^{-4} and with cosine scheduler using warm restarts, similarly to (Song et al., 2020), (ii) an Adam optimizer with an initial learning rate of 10^{-3}. The SGD optimizer with a cosine scheduler led to better results in comparison with the Adam optimizer (albeit slightly slower). Secondly, we jointly trained the CNN for EMG measures and ResGCN-N51 for 3D skeleton data for training the architecture to include information from EMG.

3.4.2 Model selection and evaluation

The model selection results are presented in Table 3.3. We first evaluated the ResGCN-N51 architecture at 0.77 million parameters in different Loss/optimizer settings. We then evaluated the best performing combination with the PA-ResGCN-B19 model with basic blocks (B19 stands for *19 basic blocks*, 3.61 million parameters) that achieved SOTA on NTU-60 and NTU-120. Finally, we evaluated the multimodal inclusion of EMG to ResGCN-N51 (best performing model). The ResGCN-N51 model configuration with a weighted loss and SGD gave the best performance with an F_1 score of 0.64 in comparison with the same model with SGD and an unweighted loss (0.54 F_1, +10%). This improvement demonstrates the benefits introduced by the weighted loss. One can see that unweighted loss generally gives a slightly better precision (compared to the PA-ResGCN-B19 model), with a recall

Model	Params.	EMG	Weighted. Loss	Optimizer	Class	F_1	P	R	Avg. F_1	Weighted Avg. F_1	MCC	Acc.	Balanced Acc.
N51	0.73m	✗	✗	SGD	1	0.54	0.55	0.52	0.73	0.87	0.46	0.87	0.73
					0	**0.93**	0.92	**0.93**					
N51	0.73m	✗	✓	SGD	1	**0.64**	**0.56**	**0.74**	**0.78**	**0.89**	**0.58**	**0.88**	**0.82**
					0	**0.93**	**0.95**	0.90					
N51	0.73m	✗	✓	Adam	1	0.50	0.42	0.63	0.70	0.83	0.41	0.82	0.74
					0	0.89	0.93	0.85					
PA-N51	1.11m	✗	✗	SGD	1	**0.56**	0.47	0.69	0.73	0.85	0.48	0.84	0.78
					0	0.90	0.94	0.87					
PA-N51	1.11m	✗	✓	SGD	1	**0.57**	0.44	0.79	0.73	0.85	0.5	0.83	0.81
					0	0.89	0.96	0.83					
PA-B19	3.61m	✗	✓	SGD	1	0.56	0.47	0.69	0.73	0.86	0.48	0.84	0.78
					0	0.91	0.94	0.87					
N51 + CNN	0.86m	✓	✓	SGD	1	0.60	**0.56**	0.64	0.75	0.88	0.53	**0.88**	0.78
					0	**0.93**	0.94	0.91					

TABLE 3.3: Model selection results for ResGCN variants (Radouane et al., 2021).

approximately similar to precision, while the weighted loss maximizes the recall and F_1 score. The Adam optimizer (on ResGCN-N51) shows less performance compared to SGD with the cosine scheduler. With the PA-ResGCN-B19 model for the best previous loss/optimizer combination, we observed lower overall performance. Contrarily to NTU-60 or NTU-120, which are very large datasets, our training data is comparatively smaller for Task 1, and we hypothesize that this model needs much more data to achieve better performance. Likewise, the introduction of the EMG signals to the ResGCN-N51 model degrades the results, as performance drops by -6%. The reason for this decrease could be either a low signal-to-noise ratio in the filtered EMG signal envelope, or the cause of an unsuitable fusion technique (single fully connected layer).

Result of submitted runs on the Test set. The Test results presented in Table 3.4 [2] show the performance of the three submitted models. Among the participating architectures, our model achieved the highest overall F1-score. Specifically, the utilization of weighted loss contributed to an improved F1-score, particularly for the minor class, yielding a value of 53.36. Additionally, our model based on the ResGCN architecture achieved the highest F1-average score of 0.712. The transformer-based architecture closely followed with an F1-score of 41.05 for the minor class. Subsequently, we will assess the performance of the additional models implemented after the challenge and proceed with optimal threshold selection analysis.

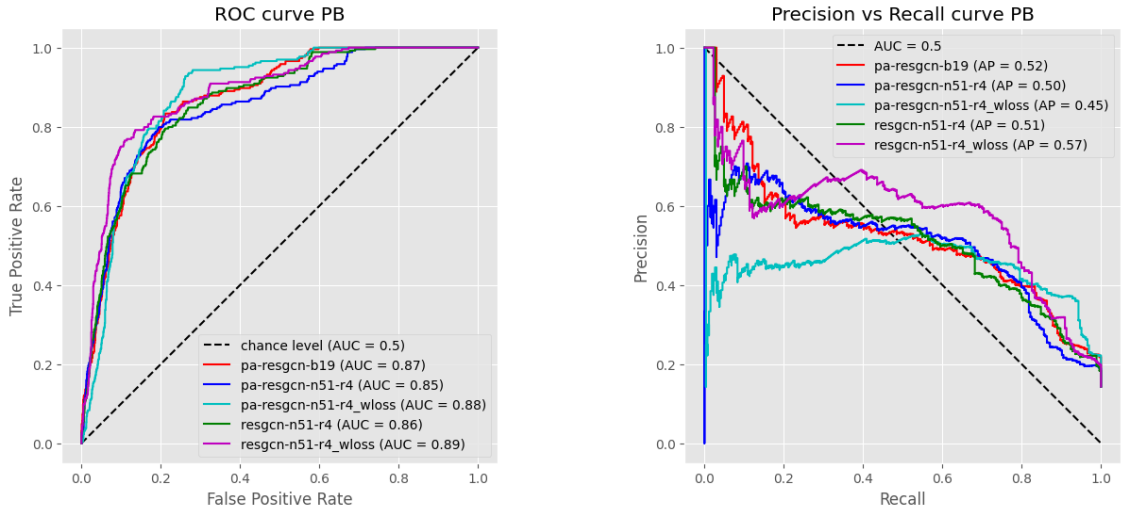
[2]Corrected model name from (Radouane et al., 2021).

System	F_1 C0	F_1 C1	MCC	Acc.
1. LSTM Baseline	85.83	31.16	0.17628	76.49
2. Transformer Baseline	**89.96**	41.05	0.35234	**82.84**
3. ResGCN-N51	89.07	**53.36**	**0.42471**	82.28

TABLE 3.4: Official results on Task 1 test set for the three submitted runs.

3.4.3 Performance analysis

It is usual to conduct performance analysis using ROC curves and precision/recall curves, particularly in the context of binary classification on imbalanced datasets. These analyses enable the evaluation of the overall model performance at various threshold levels. Optimal threshold selection is crucial as it can greatly enhance the classification results on different metrics, such as F1-score and balanced accuracy. To begin with, Figure 3.12 shows the curves corresponding to all trained models using MoCap modality. Notably, the model named `ResGCN-N51-r4_wloss`, which was trained using weighted loss, demonstrated the highest area under the curve (AUC) of 0.89 and an average precision (AP) of 0.57. These results highlight the superior performance of the `ResGCN-N51` model in comparison to the other models analyzed, as reported in Table 3.3. In the following, we explore the potential for improving these performance metrics through optimal threshold selection.



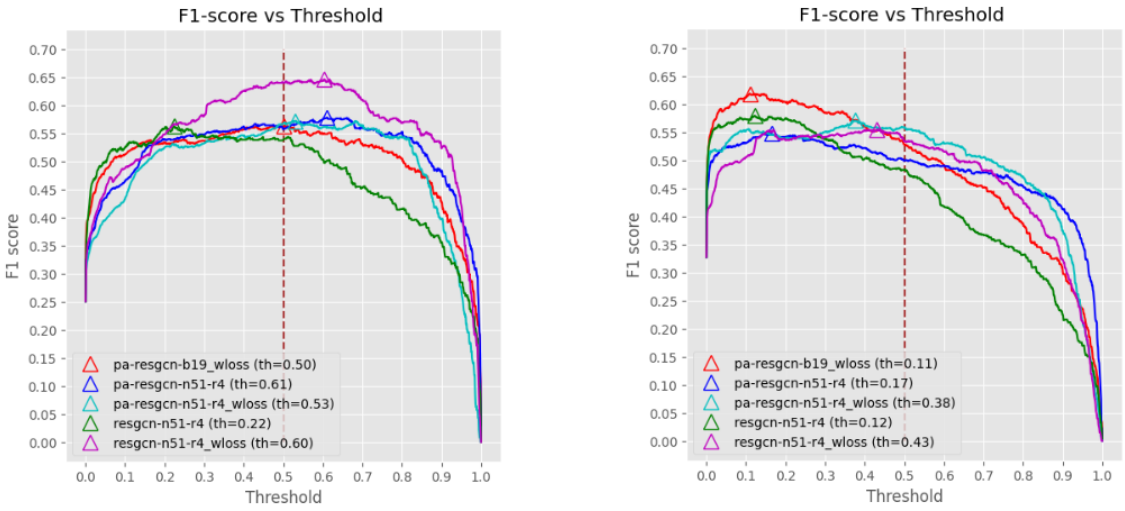
(a) ROC curves.

(b) Precision vs Recall curves.

FIGURE 3.12: Performance curves on validation set for different experimented models.

Optimal threshold selection. We propose to perform a threshold tuning by maximizing the F1-score for minor class (Positive class). The F1-score is displayed as a function of different thresholds on validation set. Then optimal threshold is depicted and used for test evaluation.

We see in Figure 3.13 that the optimal threshold is moving depending on the model used and the configuration. The `ResGCN-N51` with weighted loss perform better across a wide range of thresholds, demonstrating its robustness to threshold variation. This suggests that this model still has good generalization ability. We will further investigate this hypothesis with optimal threshold selection based on the validation set and evaluating on the test set.



(a) Validation set.

(b) Test set.

FIGURE 3.13: F1 score as function of threshold. Optimal thresholds are denoted as *th* in the legend.

Evaluation on test set with optimal threshold. In some cases, as reported in Table 3.5, the optimal threshold allows a better F1 score on the validation set. However, this behavior depends on the model and data subset. The performance of this method depends on the similarities between validation and test set, which may introduce biases in model selection. We compute the optimal threshold using only validation set and report the F1 score on the test set using the validation optimal thresholds in Table 3.5.

Results discussion. As shown in Table 3.5, the optimal thresholds are not significantly different from 0.5 when using weighted loss ($\Delta \leq 0.1$). Interestingly,

Model	F1@0.5				F1@Opt				Value	Δ	F1.Avg	
Subset	Val		Test		Val		Test				Test	
Class (C)	1	0	1	0	1	0	1	0	Opt_{th}		0.5	Opt
N51	0.54	0.93	0.48	0.89	0.56	0.91	**0.57**	0.89	0.22	0.28	0.685	**0.730**
N51+W	**0.64**	0.93	**0.53**	0.89	0.65	0.93	0.51	0.89	0.60	0.10	0.710	0.700
PA-N51	0.56	0.90	0.50	0.88	0.58	0.92	0.49	0.88	0.61	0.11	0.690	0.685
PA-N51+W	0.57	0.89	0.56	0.89	0.57	0.90	0.55	0.89	0.53	0.03	0.725	0.720
PA-B19+W	0.56	0.91	0.53	0.9	0.56	0.91	0.53	0.90	0.50	0.00	0.715	0.715

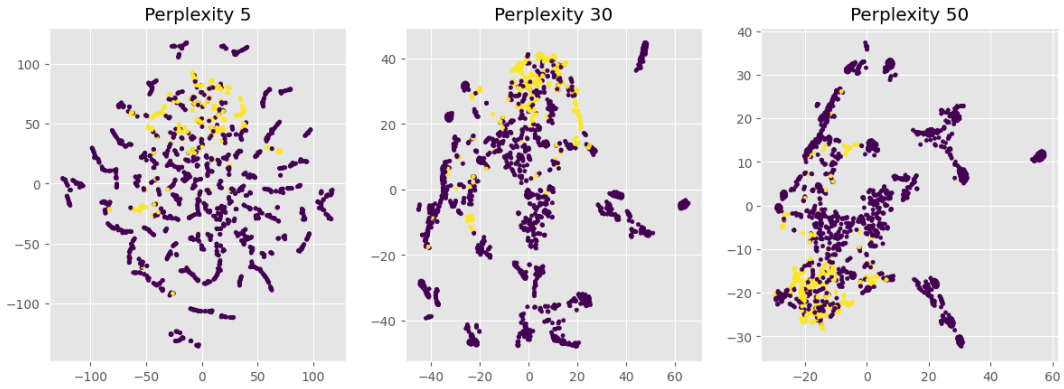
TABLE 3.5: F1 results on test and validation subsets. The optimal threshold value Opt_{th} is computed using only the validation set. Opt indicates F1 scores using Opt_{th} and $\Delta = |F1_{Opt} - F1_{0.5}|$.

even the last model, `PA-B19+W`, has the same optimal threshold 0.5 ($\Delta = 0$). In contrast, for unweighted loss, the optimal thresholds are much lower ($Opt_{th} \leq 0.12$, cf. Figure 3.13b). This observation is clearly depicted in Figure 3.13a, where we can see the distinct evolution of the F1 scores for different thresholds. The same phenomenon is evident in Figure 3.13b, which assesses the generalization ability of the models. Here, we observe that the optimal thresholds deviate significantly from the standard value of 0.5 specially for model trained with unweighted loss (See also Δ values, Table 3.5). While some of these models, like `ResGCN-N51` (unweighted), yield better F1 scores for very small thresholds. However, this selection is biased and can't be taken in consideration, as the model selection should not be impacted by the test set distribution during training and validation. This demonstrates empirically that weighted *BCE* loss allow optimizing for balanced threshold, making the real optimal values close to 0.5.

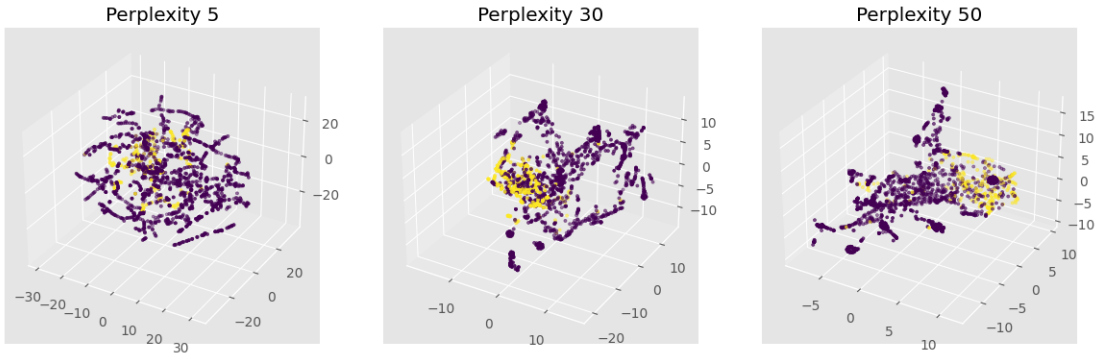
The F1 score on the test set is presented for various thresholds, serving only as a comparison with the optimal threshold (Opt_{th}) obtained on the validation set. An ideal model should maintain consistent and high F1 scores across different thresholds (Lipton et al., 2014), which measure the robustness of the model. This implies having a higher level of confidence in correctly predicting both negative and positive samples. Among the models analyzed, the `ResGCN-N51+wloss` model demonstrates the closest adherence to this condition. As depicted in Figure 3.13a, it exhibits a more extended and elevated plateau compared to the other models.

Learned representations. We utilize t-SNE (t-Distributed Stochastic Neighbor Embedding) to visualize the high-dimensional learned features in both 2D and 3D spaces, considering various perplexity values. In Figure 3.14, we observe a notable class imbalance between the positive class (represented by yellow points) and the

negative class (represented by violet points). In this lower-dimensional representation, the data points are not distinctly separated. However, the accuracy of this observation is related to t-SNE quality which depends on the extent to which the structure of the data is preserved. The 3D space representation (Figure 3.14b) with perplexity 50 could give a better visualization close to the real distribution and structure of the original high-dimensional space.



(a) 2D tsne-visualization.



(b) 3D tsne-visualization.

FIGURE 3.14: T-SNE Visualization of learned features by ResGCN-N51+W on the validation set.

3.5 Investigation of prediction explainability by implicit and explicit attention visualization

Despite the prevalence of machine learning models, their inner workings often remain opaque, seen as black boxes. However, comprehending the factors influencing predictions is crucial for establishing trust (Ribeiro et al., 2016), especially when decisions or actions are based on these predictions or when considering the

deployment of a new model. This understanding not only enhances trustworthiness but also offers valuable insights that can drive the transformation of an unreliable model or prediction into a reliable one. Therefore, we delve into a thorough examination of model interpretability, seeking to elucidate the underlying factors driving predictions. Additionally, we explore the challenges posed by the attention mechanism in achieving a comprehensive understanding of the model's decision-making process.

3.5.1 Explicit: Part level attention

The first method of visualization to try understanding model prediction is to visualize the attention score values for each human skeleton part (cf. Figure 3.10). We display attention map as *Part-level attention per each layer* for the sample *Stand on leg/Abnormal (class 1)* from the validation set in Figure 3.16. The attention distribution seems not selective. The phenomena are observed across multiple samples. To investigate this hypothesis, we draw the histogram of attention weights per each part and per each layer in Figure 3.15. We observe that most of the distributions have mean close to $m = 0.2$ and low standard deviation $\sigma \in [0.020, 0.080]$. The value $m = 0.2$ means that somehow attention weights are close to a uniform distribution in average across the five body parts ($5m = 1$). Which also indicates that the model pay equal attention on whole body parts. This observation could also explain why the part attention was not very effective for this binary classification. The model without part attention *ResGCN* has led to better results.

To provide an example, when examining the attention map for the *stand-on-one-leg* action in the abnormal class (cf. Figure 3.16), it was expected that the leg parts would receive higher attention. However, the attention was observed to be uniformly distributed on average, indicating a lack of selectivity. Additionally, some channels exhibited saturation, further complicating the interpretation of the attention patterns. These observations highlight the need for further investigation and improvement in the part-level attention mechanism for this particular dataset.

In general, attention mechanism may not be directly explainable as reported in (Bai et al., 2021). It is important to consider the possibility that different attention distributions can result in the same output. Which significantly increases the complexity of interpretation process. Moreover, **there may not be a unique solution to the problem, but there could be one solution that aligns with our intuition**.

Indeed, the model does not guarantee to converge to this specific solution. During the optimization process, the architecture may find a shorter path that lacks interpretability at the human level. Alternatively, if interpretability is a critical constraint, one potential solution is to enforce interpretability by a form of guidance incorporating supervised attention mechanism, which can provide insights into the model's decision-making process.

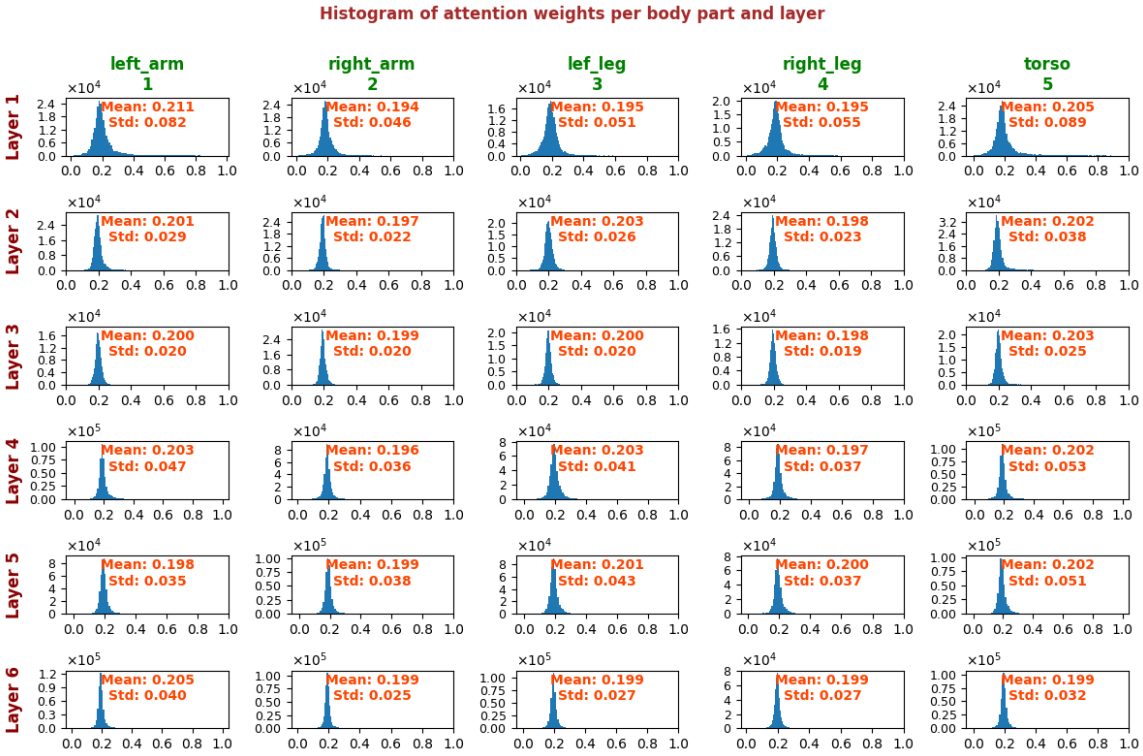
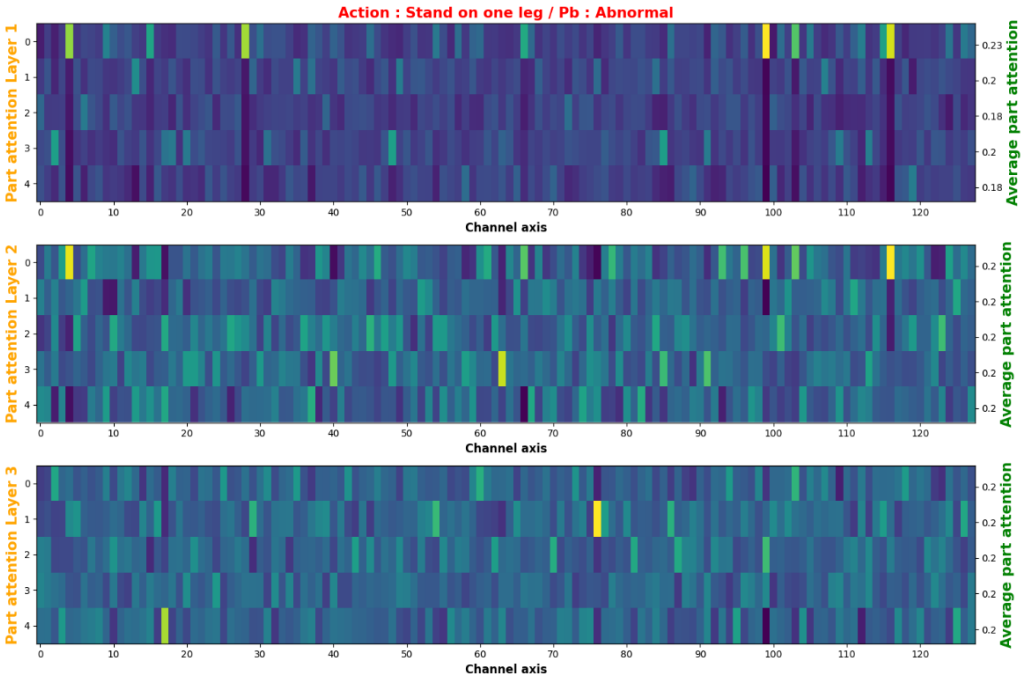
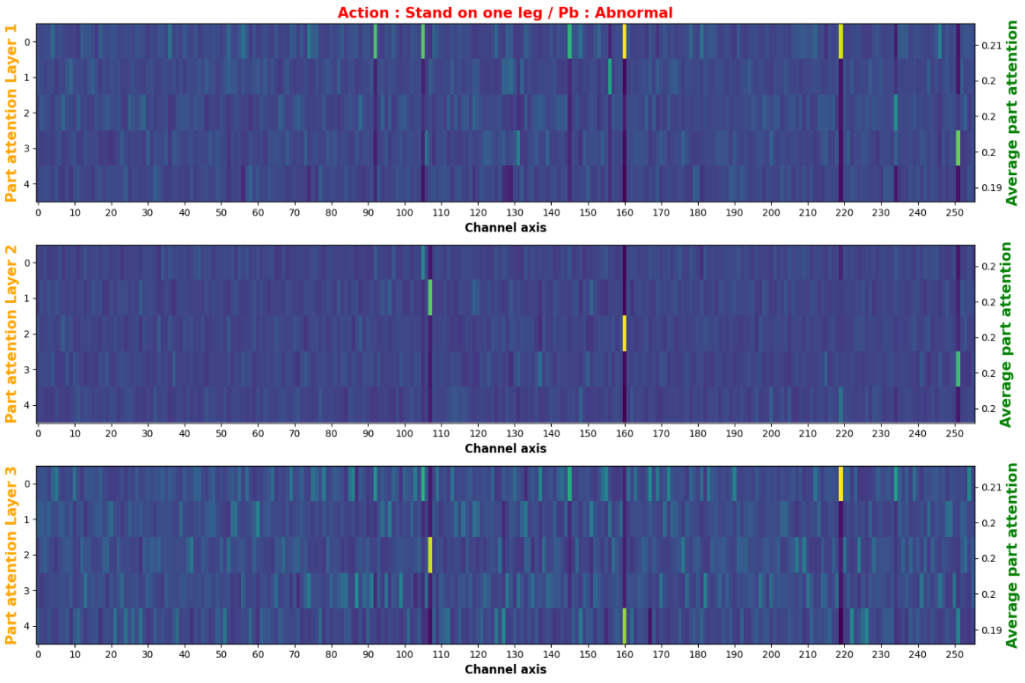


FIGURE 3.15: Attention distribution per part and layer across all validation samples.



(a) **First 3 layers** of Part attention with channel dimension of **128**.



(b) **Last 3 layers** of Part attention with channel dimension of **256**.

FIGURE 3.16: Part attention maps for the 6 six layers of PA-ResGCN-N51, the indexes from 0 to 4 respectively refer to *left arm, right arm, left leg, right leg and torso*. The right y-axis gives the average part attention.

3.5.2 Implicit: Class activation map

The Class Activation Map (CAM) method, proposed by (Zhou et al., 2016), introduces an intuitive idea of visualizing the regions that contribute significantly to an output through a weighted sum of feature maps. CAM provides a way to identify the discriminative regions in an input that are important for predicting a specific class. By computing the class activation maps, it becomes possible to visualize the *implicit attention* given to different regions, helping to understand which parts of the input contribute most to the model's decision-making process. In our context, the regions correspond to the nodes of the skeleton graph. We apply the CAM method to visualize the most important joints that contribute to the classification decision.

Formally, given the final convolutional feature maps F of size (H, W, C), where H and W represent the height and width, and C denotes the number of channels, the class activation map for a specific class c can be computed as follows:

$$\mathbf{M}_c(x, y) = \sum_{k=1}^{C} w_k^c \mathbf{F}_k(x, y) \tag{3.23}$$

where w_k^c is the weight associated with the $k-th$ channel feature map $\mathbf{F}_k$ given a class c. In our specific case, given the topology of the architecture (cf. Figure 3.9) the height $H = T/r$ and width $W = V$ and C is units number of the final fully connected layer, numerically $(H = \frac{180}{4}, W = 17, C = 256)$.

To obtain a heatmap visualization of the class activation map M_c, the *ReLU* activation is applied to clip negative values, then followed by a normalization step.

Joint-level contribution. In the models proposed with no attention mechanism, we don't have explicit attention weights. However, we can still show implicit learned attention using the Class Activation Map (Zhou et al., 2016) described above. By generating the CAM for a specific class, we can identify the key joints that are crucial for the model's decision-making process in distinguishing that particular class. This visualization provides insights into the specific joints that are implicitly attended to by the model, aiding in the understanding of the underlying features driving the classification task. Furthermore, this information can inspire architectural improvements and enhance the design of the model.

CAM method in the context of binary classification. Given that the output feature maps $\mathcal{F}$ have a size of $(1, 45, 17, 1)$ and the input has a shape of $(3, 180, 17, 1)$,

we observe a reduction factor of $r = 4$ in the number of frames $(45 = 180/4)$. In order to visualize the heatmaps across the time dimension of the input, heatmaps can simply be *interpolated*. This allows us to display the heatmaps with a resolution corresponding to the input size of $T \times V$ (*i.e.*, 180×17). The interpolation operation is given by the following Equation:

$$\begin{aligned} \Delta(t,j) &= \frac{M_c(t+1,j) - M_c(t,j)}{r-1} && \forall t \in [0, \frac{T}{r} - 1] \\ H_c(t \times r + i, j) &= M_c(t,j) + \Delta(t,j) \times i && \forall i \in [0, r-1] \end{aligned} \tag{3.24}$$

Where r is the previous-defined reduction factor and M_c is the class activation map for a class c. The value $M_c(t,j)$ represents the importance of joint j at time t. The heatmap of input resolution is denoted by H_c of shape (T,V).

In this context of protective behavior detection, we have a binary classification. Accordingly, we configure the final linear layer to yield only one output, followed by *sigmoid* activation function, which is then thresholded with a value of 0.5. In this particular case, we consider the positive values in the feature map H_c^+ as corresponding to the class $c = 1$, and negative values H_c^- as corresponding to the class $c = 0$. We decompose the resulting feature map H_c as :

$$\begin{aligned} H_c &= H_c^+ + H_c^- \\ H_c^+ &= Relu(H_c) \\ H_c^- &= -Relu(-H_c) \end{aligned}$$

By taking this aspect into account, we can utilize a single heatmap to visualize both, the regions that contribute to class 0 (Normal) and those that contribute to class 1 (Abnormal). This unified visualization provides a comprehensive view of the distinctive parts associated with each class. The predicted class, determine which of class contribution was higher.

Application using the ResGCN model. Initially, Table 3.6 provides a comprehensive mapping of joint indexes to their corresponding names. This mapping serves to enhance the comprehensibility of subsequent analyses and establish a direct correspondence with the original skeleton graph information input. Figure

3.18, showcases of several samples illustrating various actions belonging to the "Normal" (class 0) and "Abnormal" (class 1) categories.

Index	**0**	**1**	**2**	**3**	**4**	**5**
Joint Name	Pelvis	Spine	Neck	Thorax	Head	Left shoulder
Index	**6**	**7**	**8**	**9**	**10**	**11**
Joint Name	Left elbow	Left wrist	Right shoulder	Right elbow	Right wrist	Left hip
Index	**12**	**13**	**14**	**15**	**16**	
Joint Name	Left knee	Left foot	Right hip	Right knee	Right foot	

TABLE 3.6: Mapping of indexes to their skeleton joint names.

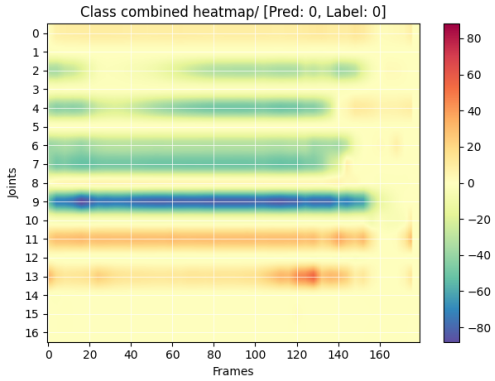
Class contribution visualization. The heatmaps visualizations in Figures 3.17 and 3.18 uses a blue color scale to indicate the contribution towards the prediction of the negative class (0), while the red color scale highlights the contribution towards the prediction of the positive class (1). The predicted class corresponds to the type of contribution with the highest value, where negative values indicate class 0 and positive values indicate class 1. The yellow regions in the heatmap indicate neutrality, suggesting that the information from those joints does not significantly impact the prediction.

Interpretability analysis. Starting with the sample shown in Figure 3.18a, where we observe that the higher contributing joints in the detection are 6 and 7, corresponding to the left arm's executing the action *reach forward*. Conversely, in Figure 3.18b, we display a case where the model incorrectly predicts the behavior, but still use the same relevant joint for decision. Despite this intuitive high contribution, the model fails to accurately classify the patient's behavior in this instance. In the case of the *Bend* action, which involves the joints 0 to 10 (Joint set of the upper body), we would expect to see a higher level contribution of joints within this range. As shown in Figure 3.18d, we observe significant contribution of joints 4, 5, and 9, which aligns with our expectations. However, in the sample depicted in Figure 3.18c, the model primarily focuses solely on the joint of index 0, known as the *root joint*.

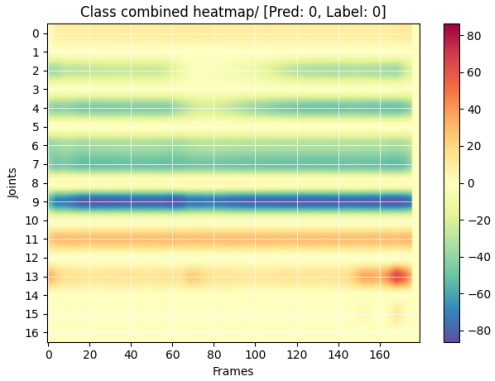
In Figure 3.18e, we can observe the significant contribution of joints 6 and 7, which correspond to the *left arm*, as well as joints 9 and 10, which correspond to the *right arm*. These joints play a crucial role in the action of *standing on one leg*. The model correctly identifies and focuses on these key joints, but still not sufficient for human to make decision.

In Figure 3.18f, we can observe that the important joints for the action *stand-to-sit* are primarily joint 11 and joint 13, which correspond to the left leg. These joints play a crucial role in the transition from standing to sitting. The model correctly identifies and focuses on these key joints, indicating its understanding of the relevant body parts involved in the action.

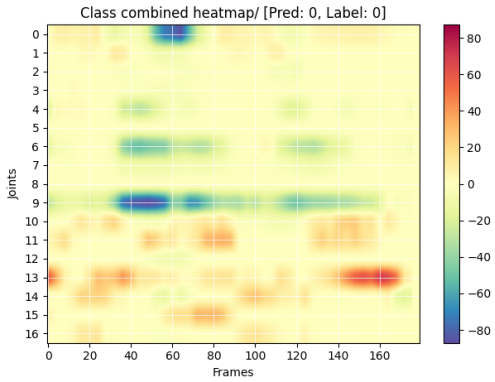
Returning to the feature extraction process, it is important to note that the Spatial Graph Convolutional Network (SGCN) operation **incorporates information from neighboring joints within a graph distance of 1**. This results in each joint containing not only its own information but also that of its adjacent joints. This phenomenon may elucidate why the contribution of a single or a few joints can be adequate for making the final decision. Hence, the significant contribution from an individual joint j can be viewed as a collective contribution from its entire neighboring set $\mathcal{N}(j)$.



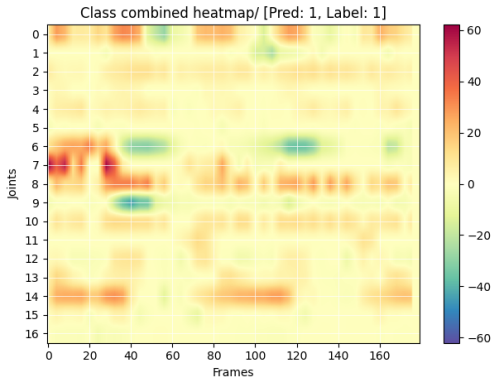
(a) Stand-to-sit/ Pb class 0.



(b) Stand-to-sit/ Pb class 0.

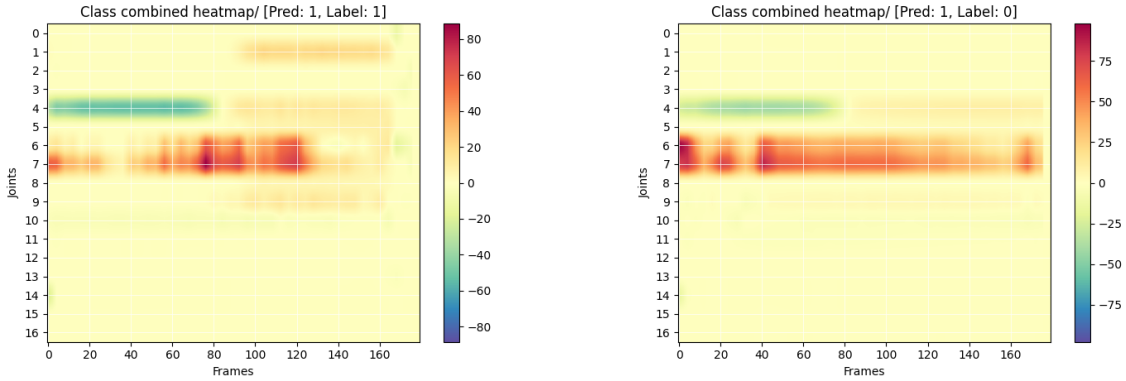


(c) Walk/ class 0.



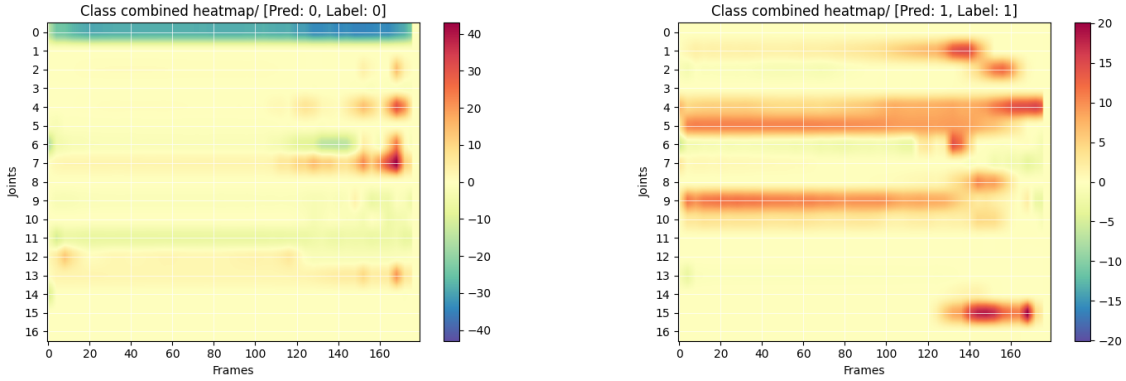
(d) Walk/ class 1.

FIGURE 3.17: Combined CAM for ResGCN-N51 (Part 1).



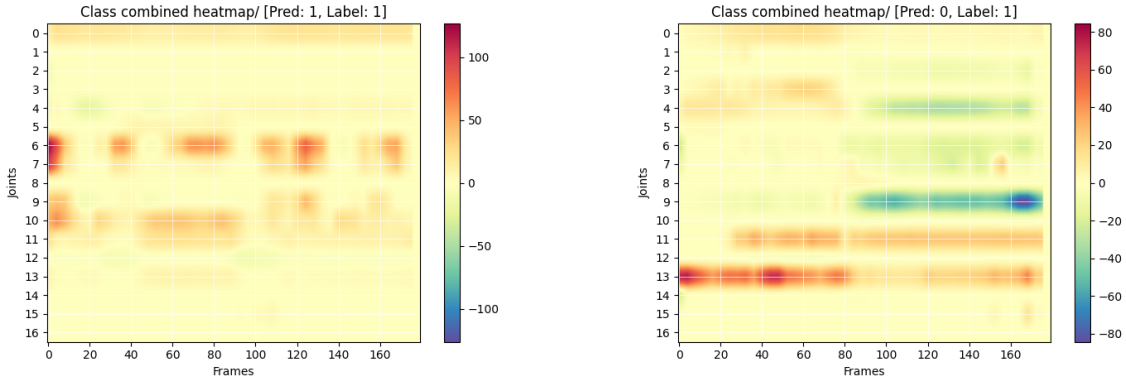
(a) Reach forward/ Pb class 1.

(b) Reach forward/ Pb class 0.



(c) Bend/ Pb class 0.

(d) Bend/ Pb class 1.



(e) Stand on one leg/ Pb class 1.

(f) Stand-to-sit / Pb class 0.

FIGURE 3.18: Combined CAM for ResGCN-N51 (Part 2).

3.6 Conclusion

In conclusion, our study delved into the utilization of graph convolutional networks (GCNs) for analyzing and modelling pose sequence data. We specifically applied GCN-based models in the AffectMove 2021 challenge (Olugbade et al., 2021), which involved solving a real-world task. Through a comprehensive threshold analysis, we assessed the influence of different thresholds on the F1 score and model robustness. This analysis highlighted the importance of selecting appropriate thresholds based on the chosen training methodology. We propose that optimizing differentiable metrics, such as those presented in (B'en'edict et al., 2021), could potentially improve the results. In the future, we can explore the incorporation of motion category information and leverage multitask learning with shared weights, which has the potential to greatly enhance protective behavior detection. Additionally, considering a more appropriate fusion of MoCap and EMG data, given the inherent heterogeneity of these two modalities, could lead to further improve the performance. The crucial concept of interpretability was discussed in details in the last part. While machine learning models often function as black boxes, understanding the reasoning behind their predictions is crucial for building trust and making informed decisions based on those predictions. Our analysis and exploration of the data have provided valuable insights into the performance of the models and the interpretability of the results. We observed that despite incorporating explicit attention in the architecture, it doesn't exhibit selectivity and can lack intuitive interpretation. We have explored the limitations posed by attention mechanisms, where models may find more efficient but less explainable paths.

Part II

Human motion captioning and segmentation

Chapter 4

Towards Synchronized Captioning of Human Motion

4.1 Introduction

In machine learning, captioning is the process of generating textual descriptions from a given input data, such as images or videos. The interest in captioning tasks stems from the need for a more efficient and effective way to understand and process visual data, such as images and videos, which are becoming increasingly prevalent in today's digital world. Current approaches, mainly focus on often vision-based input, thus, typically relies on a combination of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) or more recently use the Transformer (Vaswani et al., 2017). The aim is to produce detailed and human-like captions that can be used in several applications such as image and video retrieval and understanding. While captioning tasks have primarily focused on images and videos, limited research has explored motion captioning or human skeleton-based captioning. This approach generates captions for human motion based on estimated or ground truth poses. The human skeleton offers a concise and semantically rich representation of motion, enabling better understanding and description of human activities. The generated captions mainly focus on describing actions and poses, such as: *"a person is running then stops"* or *"a person is waving with his left hand"*.

In this chapter, we present available datasets and relevant techniques of motion representation. For motion/text representation and generation, we discuss in detail two key designs: i) Vector Quantized Variational AutoEncoder (VQ-VAE) (van den Oord et al., 2017), and ii) the Transformer architecture (Vaswani et al., 2017). Connecting attention mechanisms introduced for machine translation to the scope of our study as source of inspiration (Section 4.3). Then, we focus on proposed methodologies (Section 4.4). Particularly, this chapter will focus on experiments towards *Synchronized Captioning* (Section 4.5), involving qualitative analysis of human segmentation. Then, this semantic segmentation will be formally defined and quantitatively evaluated in the next Chapter 5. The following discussion and analysis constitute our second contribution based on our paper (Radouane et al., 2023a)[1].

[1]Code: https://github.com/rd20karim/M2T-Segmentation

4.2 Mapping between motion and language

In recent years, numerous motion encoders have been proposed to address the challenges of motion and text generation. Excluding studies focusing on bidirectional mapping (Plappert et al., 2018; Toyoda et al., 2022; Guo et al., 2022b), it is evident that the field of motion generation has witnessed significant advancements, with extensive research efforts dedicated to this task (Guo et al., 2022a; Zhang et al., 2023; Ghosh et al., 2021; Petrovich et al., 2022; Chen et al., 2023). In contrast, the progress in motion to language generation has been comparatively less substantial and advanced (Goutsu and Inamura, 2021; Takano and Lee, 2020).

This section presents techniques involved in motion and language mapping in both directions. In this context, we highlight existing datasets and benchmarks that make supervised learning possible (Section 4.2.1). Then, we categorize relevant methods of motion representation (Section 4.2.2) and related architecture design. Finally, we review existing methods for motion and language generation (Section 4.2.3).

4.2.1 Datasets

The study of complex human movements and actions often requires the use of motion capture based datasets. One of the most widely used datasets is the KIT Motion Language Dataset (KIT-ML) (Plappert et al., 2016). This dataset is created by combining records from the *CMU Motion Database* and the *KIT Whole-Body Human Motion Database*, and allows users from around the world to contribute to motion annotations using an online tool. The annotations describe the entirety of each movement, even when it consists of multiple actions, often in the form of single sentences or small paragraphs. Another dataset proposed by (Takano et al., 2016, 2019) that describe complex actions and movements. It contains 467 motions and 764 annotated segments, and was designed for human-robot interaction. The annotations in this dataset are time-indexed and describe successive phases of the movement as separate short sentences. In addition to these two datasets, there is also the HumanML3D dataset (Guo et al., 2022a). This dataset includes recordings of various movements such as walking, running, jumping, and dancing. The HumanML3D dataset can be used for tasks such as motion captioning, prediction, action recognition, and human-robot interaction.

For our initial work on motion-language synchronization, we utilized the KIT-ML dataset (Plappert et al., 2016). Later on, an updated version of the KIT-ML

dataset, along with a larger dataset named HumanML3D, was introduced in a recent paper by (Guo et al., 2022a). We present their statistics in Table 4.1. Consequently, we extended our experiments to include these recently released datasets in Section 4.5.6.

Subset	Number	Train	Test	Val.
KIT-ML	motions	2375	291	262
	samples	5071	612	568
KIT-ML-aug	motions	4886	830	300
	samples	10408	1660	636
HML3D-aug	motions	22068	4160	1386
	samples	66734	12558	4186

TABLE 4.1: Training splits and dataset statistics, for KIT-ML (Plappert et al., 2016), Human ML3D and KIT-ML after augmentation (Guo et al., 2022a).

4.2.2 Motion representation

In the previous Chapter 3, we explored the concept of graph modeling as an intuitive approach for encoding motion data based on skeletons. In this part, our focus shifts towards more specific methods for tasks involving language and motion generation. The success of an architecture heavily relies on its feature extractor, which plays a crucial role in capturing relevant information for a specific task. Within this context, an important concept to consider is the effective representation of motion, especially in relation to language. *Motion representation* can take on a continuous form, such as raw pose data that exhibits continuous variations in real human motion over time. Alternatively, motion can be represented discretely as tokens, which can be more suitable for utilization within sequential-to-sequential (seq2seq) architectures like the Transformer model (Vaswani et al., 2017). Thus, the methods of motion representation can be categorized into two types: continuous and discrete representations.

Learning continuous representations. Significant research has been conducted on the development of robust and denoised continuous representations in the field

(Odena, 2016; Chen et al., 2016). In our specific context, motion is naturally expressed as a continuous sequence of pose vector, allowing for the capture of temporal dynamics and subtle variations in joint positions. Raw pose data, which consists of joint positions at different time steps, is a common form of continuous representation. Other derived representations such as velocities and accelerations can be included as continuous representations over time. These types of representations offer a comprehensive description of motion patterns. Previous studies have extensively employed continuous representation in the analysis of motion based on skeletal data (Plappert et al., 2018; Takano et al., 2020; Takano and Lee, 2020). However, learning from continuous data presents several drawbacks that need to be considered. Firstly, it introduces increased computational complexity. Secondly, the sensitivity of continuous data to noise and measurement errors can negatively affect the accuracy of learned models. Furthermore, complex models trained on continuous data also face a higher risk of overfitting, as they may capture noise or irrelevant patterns.

Learning discrete representations. Several works have been proposed for learning of discrete representations (Jang et al., 2017; Hu et al., 2017). In this context, our focus will be on one particular method that has gained prominence in various generative tasks and is indispensable for comprehending the forthcoming methods that aim to learn motion tokens. Proposed by (van den Oord et al., 2017) to effectively learn discrete representation, the Vector Quantized Variational Autoencoder (`VQ-VAE`) is a variant of the Variational AutoEncoder (`VAE`). This discrete latent space is motivated by the prevalence of discrete entities in the real world, such as *words* and *phonemes*, and enables the `VQ-VAE` to capture the global structure of data while *avoiding the modeling of noise* and *fine-grained details*. By embedding data into a discrete latent space, the `VQ-VAE` achieves *data compression* and learns meaningful representations that encompass global information.

Additionally, the `VQ-VAE` is capable of modeling long-range sequences, operates in a fully unsupervised manner, mitigates the issue of *posterior collapse* and effectively models features that span multiple dimensions in the data space. These characteristics make the `VQ-VAE` a powerful design for learning informative representations from complex data.

The architecture of `VQ-VAE` is presented in Figure 4.1. The model follows the traditional *Encoder-Decoder* design, but introduces a quantization process between

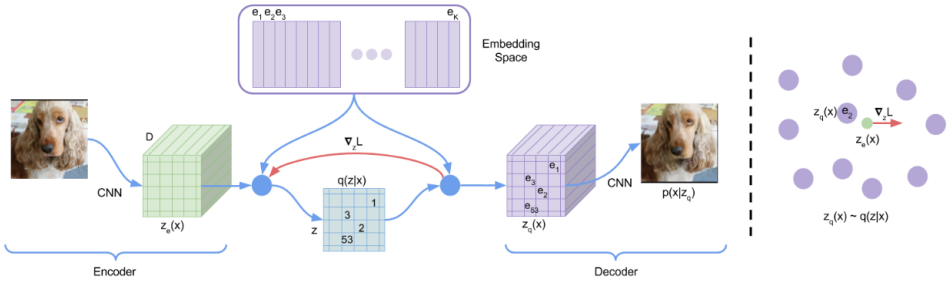


FIGURE 4.1: The *VQ-VAE* architecture (van den Oord et al., 2017).

the encoder and decoder. Formally, the encoder network models the *posterior distribution* $q(z \mid x)$ of discrete latent variables z given an input x, a *prior distribution* $p(z)$, and the decoder network generates data x given latent variables z following the distribution $p(x \mid z)$.

In this framework, the posterior and prior distributions are categorical, and the samples obtained from these distributions are used to index an embedding table $e = (e_1, ..., e_K)$ where K represents the size of the discrete latent space (cf. Figure 4.1). This latent embedding space $e \in \mathbb{R}^{K \times D}$, and D is the dimension of each latent embedding vector e_i. The space consists of K embedding vectors $e_i \in \mathbb{R}^D, i \in \{1, 2, \ldots, K\}$. The discrete latent variables z are obtained through a *nearest neighbor lookup* in the shared embedding space e.

The posterior categorical distribution $q(z \mid x)$ probabilities are defined as one-hot:

$$q(z = k \mid x) = \begin{cases} 1 & \text{for k} = \text{argmin}_j \left\| z_e(x) - e_j \right\|_2, \\ 0 & \text{otherwise} \end{cases}, \tag{4.1}$$

where $z_e(x)$ is the output of the encoder network. The equation 4.2 gives the discrete representation of the input x.

$$z_q(x) = e_k, \quad \text{where} \quad k = \text{argmin}_j \left\| z_e(x) - e_j \right\|_2 \tag{4.2}$$

The gradient for equation 4.2 is not explicitly defined, but it is approximated using a technique similar to the straight-through estimator. In this approach, as illustrated in Figure 4.1 the gradients from the decoder input $z_q(x)$ are directly copied to the encoder output $z_e(x)$. This approximation allows for the propagation of gradients through the *quantization process*, enabling the training of the model.

Application in human motion tokenization. Drawing inspiration from the successful implementation of discrete representations in various fields, such as image (Razavi et al., 2019) and video generation (Yan et al., 2021), recent approaches have embraced the concept of *motion tokens*. Specifically, these methods adopt the foundational principles of the `VQ-VAE` architecture and adapt the *encoder-decoder* framework to create *motion tokens* as intermediate representation.

In the following, we will explore the diverse techniques employed for motion representation and tokenization in the context of motion and language generation.

- **Codebooks.** In order to tackle the issue of lifeless motion, a motion tokenization approach is introduced by (Guo et al., 2022b). The authors propose to employ deep vector quantization to learn 3D spatial-temporal *codebooks*. This enables the transformation of 3D pose sequences into a sequence of tokens, referred to as motion tokens. Learning these tokens involves the following operations:

  Motion representation m: Given a pose sequence $\mathbf{m} \in \mathbb{R}^{T \times D_p}$, where T represents motion duration and D_p is the dimension of pose representation.

  Motion encoding $E(.)$: The encoder apply 1D convolutions along the time dimension of sequence poses to obtain latent vectors $\hat{\mathbf{b}} \in \mathbb{R}^{t \times d}$, with $t < T$ representing the number of convolution kernels. Formally, $\hat{\mathbf{b}} = \mathrm{E}(\mathbf{m})$.

  Motion quantization $Q(.)$: The encoded motion $\hat{\mathbf{b}}$ is quantized into a collection of codebook entries $\mathbf{b_q} \in \mathbb{R}^{t \times d}$ using discrete quantization. The learnable codebook $\mathcal{B} = \{\mathbf{b}\}_{k=1}^{K} \subset \mathbb{R}^d$ consists of K latent embedding vectors, each of dimension d. The quantization process $\mathrm{Q}(\cdot)$ replaces each row vector $\hat{\mathbf{b}}_i \in \mathbb{R}^d$ in $\hat{\mathbf{b}}$ with its nearest codebook entry $\mathbf{b}_k$ in $\mathcal{B}$, defined as:

$$\mathbf{b_q} = \mathrm{Q}(\hat{\mathbf{b}}) := \left(\mathrm{argmin}_{\mathbf{b}_k \in \mathcal{B}} \left\| \hat{\mathbf{b}}_i - \mathbf{b}_k \right\| \right) \in \mathbb{R}^{t \times d} \tag{4.3}$$

  Motion Reconstruction $D(.)$ The de-convolutional decoder D projects $\mathbf{b_q}$ back to the 3D motion space, yielding a pose sequence denoted as $\hat{\mathbf{m}}$.

  Finally, the entire process depicted in Figure 4.2 can be formulated as:

$$\hat{\mathbf{m}} = \mathrm{D}(\mathbf{b_q}) = \mathrm{D}(\mathrm{Q}(\mathrm{E}(\mathbf{m}))) \tag{4.4}$$

  The model is trained using a reconstruction loss combined with embedding commitment loss terms that encourage latent alignment and stabilize the training process:

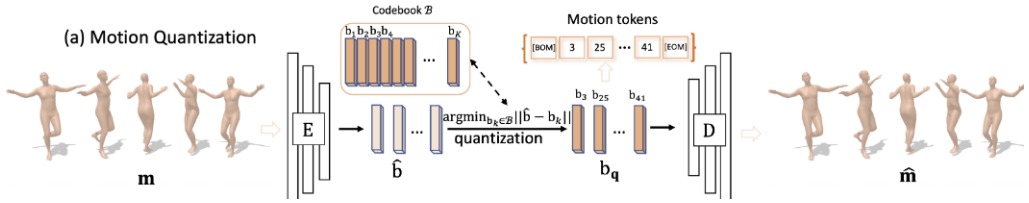
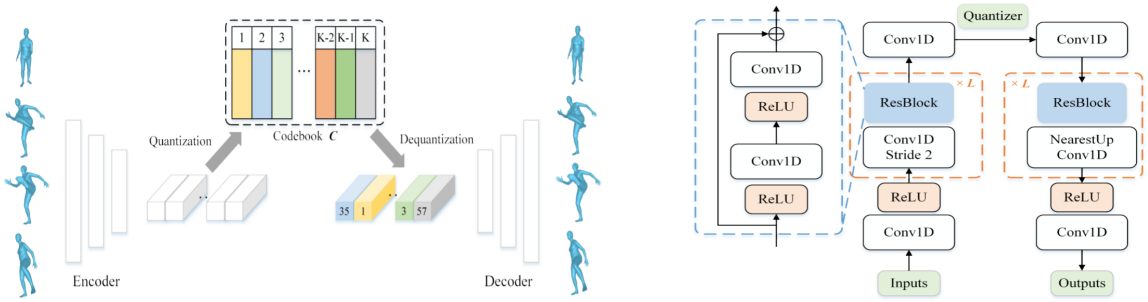


FIGURE 4.2: Mapping of motion tokens to their sequence of poses. We observe that each token represent a motion primitive. e.g., token 168 → *Wave with left hands*, 446 → *Bend* (Guo et al., 2022b).



(a) Codebooks.

(b) Encoder, Decoder design for the VQ-VAE.

FIGURE 4.3: Codebooks learning details (Zhang et al., 2023).

$$\mathcal{L}_{vq} = \|\hat{\mathbf{m}} - \mathbf{m}\|_1 + \left\|\text{sg}[\text{E}(\mathbf{m})] - \mathbf{b_q}\right\|_2^2 + \beta \left\|\text{E}(\mathbf{m}) - \text{sg}\left[\mathbf{b_q}\right]\right\|_2^2 \tag{4.5}$$

Figure 4.2 showcases intriguing motion tokens, generated through the utilization of learned tokens. Intuitively, these sub-movements can be perceived as motion basis, encapsulating a range of *primitive motions* present in the dataset distribution. By appropriately sequencing these tokens using the decoder D, it becomes feasible to reconstruct the original motions. The incorporation of motion tokens greatly facilitates the generation of both words and motions. As the words are inherently represented as tokens, the association between words and tokenized motion versions is streamlined due to the inherent proximity introduced by the quantization process.

The concept of codebooks was re-used by (Zhang et al., 2023). In their work, the authors propose the learning of codebooks (cf. Figure 4.3a) inspired by the `VQ-VAE` architecture, a custom design is proposed for encoding and decoding motion (cf. Figure 4.3).

- **Motion snippet code.** This type of representation learning was proposed by (Guo et al., 2022a), for the generation of ***natural*** and ***diverse*** 3D human motion

conditioned on text input. They introduce motion snippet code as the internal motion representation. The motion snippet is the intermediate-learned representation between encoder and decoder (cf. Figure 4.4). Compared to individual poses, the snippet code effectively captures temporal semantic information that is essential for generating coherent and realistic motion. The motion snippet code has an 8-frame receptive field, corresponding to approximately 0.5 seconds for a pose streaming rate of 20 frames per second (fps). This design also results in a more condensed internal code sequence with a length of $T = T'/4$.

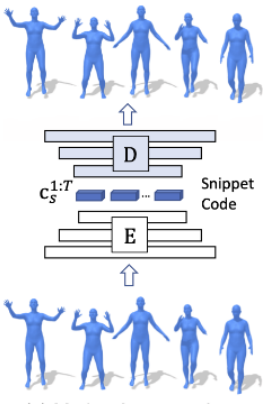


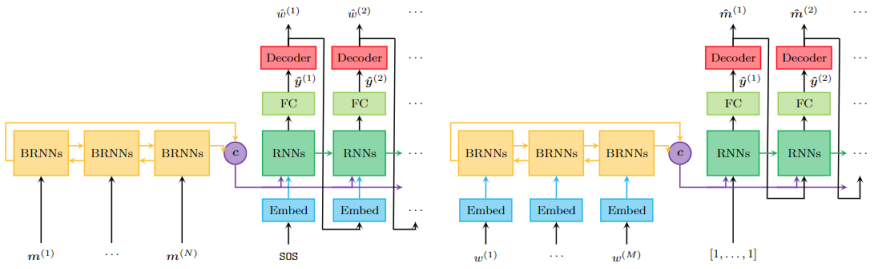
FIGURE 4.4: The auto-encoder architecture ($E - D$) comprises two-layer convolutions with a filter size of 4 and a stride of 2 (Guo et al., 2022a).

- **Motion Codebook.** Recently, in the context of multitask learning on a motion-text dataset, (Jiang et al., 2024) proposed an architecture design similar to previous works for learning a discrete representation of motion, known as *motion tokens*. The architecture is entirely based on the `VQ-VAE` framework. The Encoder applies a series of 1D convolutions along the frame time to obtain a sequence of latent representations, followed by a quantization process as described earlier.

4.2.3 Motion and language generation

The literature provides numerous examples of works aimed at generating motion from textual descriptions. Particularly, transformer operations (Vaswani et al., 2017) are at the forefront of advanced architecture design, making them a crucial tool for tackling a wide range of machine learning tasks. In this part, we will focus on classic and most recent approach for motion and language generation. These methods are respectively based on recurrent neural networks (RNNs) and Transformers.

RNN-based design. A first model addressing the two-direction generation tasks was proposed by (Plappert et al., 2018), using the KIT-ML dataset. Their technique, depicted in Figure 4.5, proposes a similar architectural design for both motion and language generation. Specifically, in Figure 4.5.(a), the model learn the mapping of motion sequences $M = (m^{(1)} \dots m^{(N)})$ to language. The motion sequences is initially encoded using a stack of bidirectional RNNs to obtain a context vector c. This context vector is further decoded by another stack of unidirectional RNNs, with the embedded word generated in the previous time step serving as an additional input. The fully-connected layer (FC) produces the parameters $\hat{\boldsymbol{y}}^{(t)}$ representing the output probability distribution, from which a concrete word $\hat{w}^{(t)}$ is sampled using the decoder. This process continues iteratively until the end-of-sequence (EOS) token is emitted, resulting in the complete generated description $\hat{w} = (\hat{w}^{(1)}, \hat{w}^{(2)}, \dots)$. On the other hand, Figure 4.5.(b) illustrates the reverse direction, where a natural language description $\boldsymbol{w}$ is used to generate the corresponding whole-body motion $\hat{M}$. This process follows a similar approach, employing a stack of RNNs to encode the description and generate the motion sequence. Importantly, it should be noted that the models for both directions are trained separately and do not share any weights.



(a) Motion-to-language model. (b) Language-to-motion model.

FIGURE 4.5: The bidirectional mapping model proposed by (Plappert et al., 2018).

More recent and advanced work proposed by (Guo et al., 2022a) focus only on the motion generation. The architecture details presented in Figure 4.6. The `GRU` variant of RNNs is the main compound. The `VAE` is composed of a *prior* and *posterior network*, both are simply GRU-based. Two modules not displayed are *Text2length*, which samples the motion length, and *Text2Motion*, an architecture based on a *temporal* `VAE` used for motion generation. With these modules in place, it becomes possible to generate **natural** and **diverse motions**.

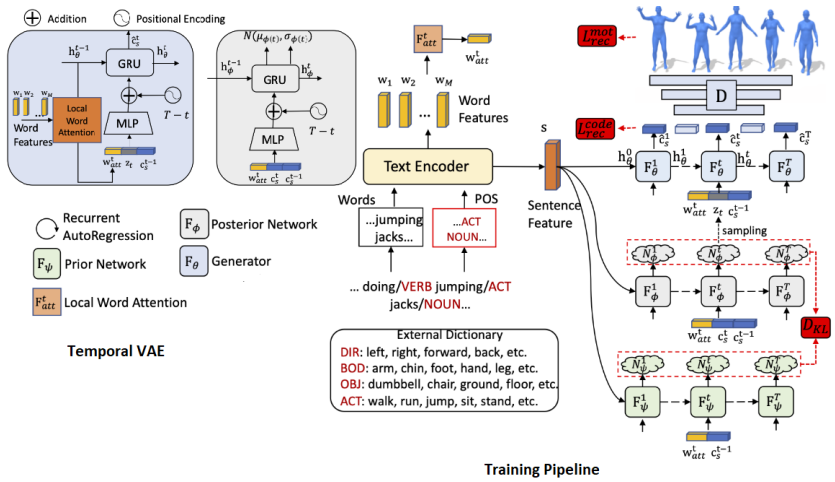


FIGURE 4.6: Motion generation based on GRU and motion snippet code (Guo et al., 2022a).

Transformer-based design. More recently, both modes of generation have been addressed by (Toyoda et al., 2022; Guo et al., 2022b). In the context of motion generation, a variety of techniques have been applied. (Ghosh et al., 2021) propose motion synthesis (text-driven animation) using a model based on recurrent networks, specifically Gated Recurrent Units (GRU), with a hierarchical design that encodes the upper and lower body parts of the human skeleton. Other more advanced techniques leverage the use of transformers. For example, (Petrovich et al., 2022) propose `TEMOS` for the generation of the 3D human motion conditioned on text input, through a motion encoder and text encoder. As depicted in Figure 4.7 the motion and text are encoded using the transformer-based Variational Autoencoder `VAE` introduced by (Kingma and Welling, 2022). In the inference phase, only the text encoder is used for motion generation through a transformer-based decoder.

Other studies explore bidirectional mapping. The authors proposed a transformer-based architecture (Guo et al., 2022b) (cf. Figure 4.9) to handle the generation of both text and motion. This is achieved straightforwardly by representing motion as token sequences using a codebook obtained through pretraining a VQ-VAE, as discussed in the previous subsection.

More recently, (Zhang et al., 2023) introduced an architectural design (cf. Figure 4.9) that builds upon the ideas presented by (Guo et al., 2022b). In this approach, motion is represented also as tokens, with an additional special token, namely *End*, which indicates the end of motion generation. The generation process begins by providing the start token, which are obtained by encoding the natural language

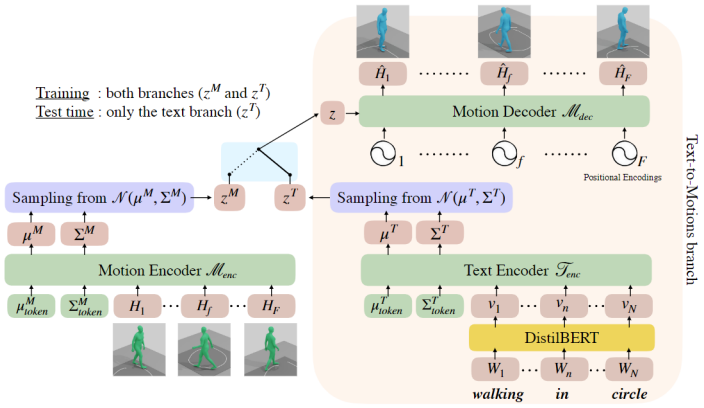


FIGURE 4.7: Motion generation with Transformer-based VAE (Petrovich et al., 2022).

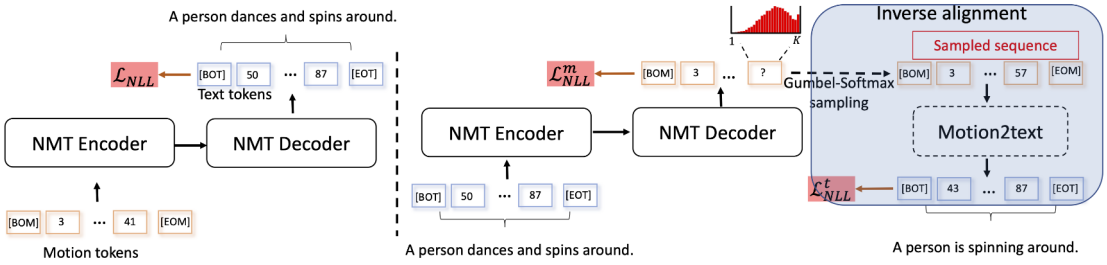


FIGURE 4.8: Motion generation with motion tokens as an NMT task (Guo et al., 2022a).

description using CLIP (Radford et al., 2021). Subsequently, a pretrained decoder is utilized to generate the motion sequence from these tokens. This architecture led to improve performance in motion generation on various metric levels compared to the approach proposed by (Guo et al., 2022b).

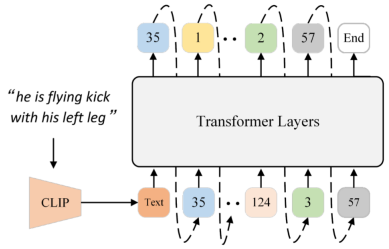


FIGURE 4.9: T2M-GPT architecture for text-to-motion generation (Zhang et al., 2023).

Another sophisticated technique was used by (Chen et al., 2023), which employs a *diffusion mechanism* to generate motion conditioned on text or action. In these works, the researchers propose learning an internal representation of motion

through the VAE model. More recently, (Jiang et al., 2024) introduced a novel aspect of multitask learning (cf. Figure 4.10) involving instruction tuning tasks. Their framework simultaneously learns *motion captioning*, *motion generation*, *motion prediction*, and *"in-between"* motion filling (connecting a given start and end motion). The key concept enabling this multitask learning is the use of a ***mixed vocabulary of motion and text tokens***, treating *motion as a foreign language*. This approach is illustrated in Figure 4.11.

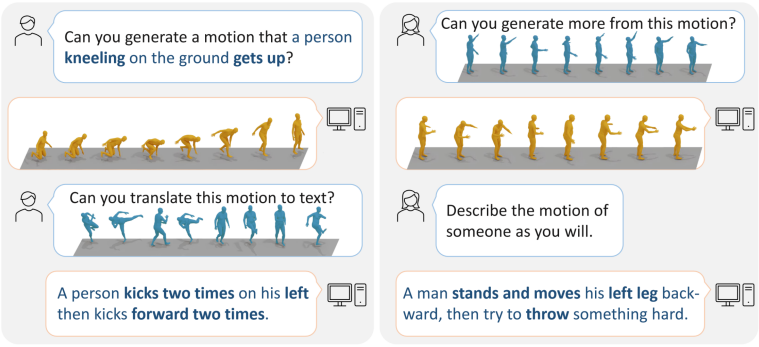


FIGURE 4.10: MotionGPT: Example of tasks learned during instruction tuning (Jiang et al., 2024)

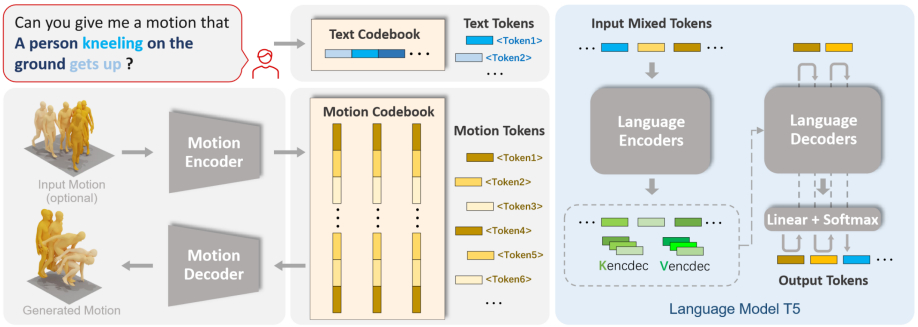


FIGURE 4.11: MotionGPT: Overview of framework design for multitask learning (Jiang et al., 2024).

4.2.4 Vanilla Transformer

In this section, we will specifically review important details about the Transformer (Vaswani et al., 2017), which has been extensively employed across numerous tasks, including motion, language encoding, and generation (Zhang et al., 2023; Guo et al., 2022a,b). However, we demonstrate in Section 4.5.6 that despite being a successful architecture, the Transformer-learned attentions are not directly useful

for interpretation and motion-word alignment. The original Transformer design was proposed for NMT tasks and consists in an Encoder-Decoder framework:

Transformer Encoder. First, the input sequence of T elements from a vocabulary of size Vs are embedded using a matrix $\mathbf{E} \in \mathbb{R}^{Vs \times d}$, resulting in a sequence of vectors $X \in \mathbb{R}^{T \times d}$. Since the Transformer does not utilize recurrent connections, a fixed positional encoding $PE_{\text{pos},i}$ is added to the embedded tokens to incorporate positional information.

$$PE_{(pos,i)} = \begin{cases} \sin\left(pos/10000^{i/d_{\text{model}}}\right) & \text{if } i \text{ is even} \\ \cos\left(pos/10000^{(i-1)/d_{\text{model}}}\right) & \text{if } i \text{ is odd} \end{cases} \tag{4.6}$$

Where $PE_{(pos,i)}$ represents the position encoding value at the vector dimension i for position pos. The resulting sequence of vectors is transformed through a series of N identical layers of *Multi-Head Attention* and *Feed-Forward neural networks* (cf. Figure 4.12).

Single-Head Attention. Implemented as a scaled-dot product attention mechanism and enables the model to attend over the entire sequence:

$$Q = X.W^Q \quad K = X.W^K \quad V = X.W^V \tag{4.7}$$

Here, Q, K, and V are the Query, Key, and Value matrices computed through linear projection of the inputs embedding X. The dot product of Q and K^T produces a matrix of similarities between each query and key, which are scaled by $\sqrt{d_k}$ to reduce the magnitude of dot-product. The *Softmax* function is then applied to obtain a probability distribution over the keys for each query. Finally, the values are weighted by these probabilities and summed to obtain the output of the self-attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.8}$$

Multi-Head Attention. In many cases, a single weighted averaged representation is insufficient to capture different details about the input, as results, the attention mechanism is extended to multiple heads, allowing multiple query-key-value triplets. Information from each head i are aggregated through linear projection W^o as in Equation 4.9.

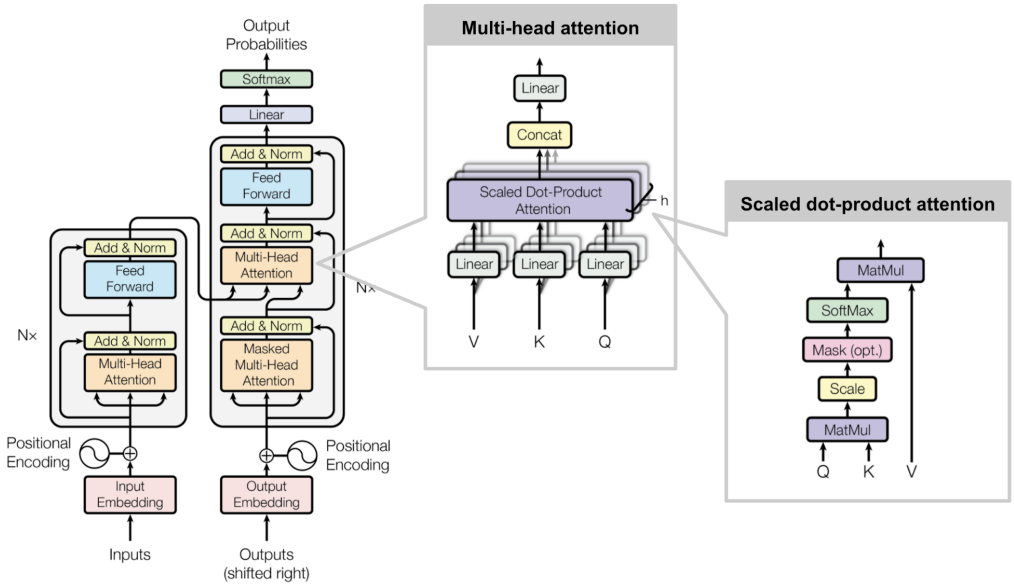


FIGURE 4.12: Transformer model compounds (Vaswani et al., 2017).

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O \\ \text{head}_i &= \text{Attention}(Q_i, K_i, V_i) \end{aligned} \tag{4.9}$$

Feed-Forward network. The output of multi-head attention is transformed with a two-layer neural network with ReLU activation as follows:

$$\begin{aligned} \text{FFN}(x) &= \max(0, xW_1 + b_1)W_2 + b_2 \\ x &= \text{LayerNorm}(x + \text{FFN}(x)) \end{aligned}$$

Transformer Decoder. Similarly, the output sequence Y is also embedded into a sequence of vectors $Y \in \mathbb{R}^{T_{out} \times d_{out}}$. These representations are transformed through the self-attention mechanism (Equation 4.9), where the future predictions from each time step t are masked in the matrices Q and K with $-\infty$, so that each token attends only to the previous tokens. Then, a cross-attention mechanism is applied, followed by a feed-forward pass, representing a single decoder layer. This process

is repeated N times. The final output representation is linearly projected and followed by a softmax function, providing a probability estimation over the output vocabulary at each time step t.

Cross-attention. This mechanism is applied using the multi-head attention mechanism (Equation 4.9), where the head queries are derived from the decoder self-attention output, while the values and keys are extracted from the final layer representation provided by the Transformer Encoder.

4.3 Attention and language modeling

Machine translation, one of the most popular tasks, consists in converting sentences from one language to another. A fairly old task, several approaches have been used over the years.

- *Rule-Based Machine Translation (RBMT)*: Is an approach that relies on the syntactic structure of both source and target languages. It operates by replacing words or phrases with their equivalent counterparts in a syntactically appropriate manner. This method follows predefined rules and patterns to ensure grammatical correctness and maintain the intended meaning.

- *Statistical Machine Translation (SMT)*: Involves segmenting the source sentence into smaller units and generating the translation based on statistical language models. These models utilize large amounts of bilingual training data to estimate the probability of generating target language segments given the source language segments. SMT has been widely used and has shown success in various translation tasks.

- *Neural Machine Translation (NMT)*: Is a more recent approach to machine translation that utilizes neural networks, specifically deep learning architectures, to model the translation process. Initially, NMT faced performance limitations due to computational constraints. However, advancements in architecture design and increased computational power have led to significant improvements in translation quality. NMT models are capable of capturing complex linguistic patterns and long-range dependencies, resulting in more accurate and fluent translations.

In SMT approaches, a considerable number of linguistic models have been formulated under the following Markov assumption:

$$p(x_1, \cdots, x_T) = \prod_{t=1}^{T} p(x_t|x_1, \cdots, x_{t-1}) \approx \prod_{t=1}^{T} p(x_t|x_{t-n}, \cdots, x_{t-1}) \tag{4.10}$$

Where n representing the number of previous words used for probability computation, this simplifies the computation. However, this configuration imposes limitations on capturing long dependencies within the target sequence.

In contrast, NMT approaches use neural networks, particularly sequence-to-sequence models, which do not explicitly rely on the Markov assumption. The probabilistic view of NMT tasks involves generating a probability distribution over the target vocabulary and then selecting the output that maximizes the individual probabilities at each prediction step. In recent years, NMT approaches have witnessed the introduction of attention mechanisms in various forms, significantly enhancing the quality of generated text and offering a new perspective for interpretability analysis through learned attention weights. The difference in attention types stems from how the context vector is computed. Utilizing all encoder outputs corresponds to *soft attention* (Bahdanau et al., 2015), while employing only a subset of the encoder outputs is referred to as *local attention* (Luong et al., 2015).

In the following, we provide a detailed definition of these two attention modes after a brief review of the state of NMT without attention. Later on, we investigate their respective strengths and weaknesses in motion captioning context (Section 4.4). Next, we formulate our proposition of *local recurrent attention* as an effective solution for improving the quality of generated text (Section 4.4.3), particularly enabling synchronized captioning (Section 4.5).

4.3.1 RNNs without attention

Focusing on the neural machine translation approach, before the incorporation of attention mechanisms (Cho et al., 2014; Mikolov et al., 2010), RNNs networks (e.g., LSTM) were employed to read the sequence of input words and generate an output vector of fixed size. This vector was then used by an RNN decoder to generate the output sentence in the target language's vocabulary in an autoregressive manner. However, this technique exhibited less stability in performance when the source sentence exceeded an average length of 20–50 words. The phenomenon is attributed to RNN-based architectures compressing extensive information into a fixed-size vector, causing information loss in longer sentences. This problem was

mitigated by (Sutskever et al., 2014) with a simple word order reversal technique, the architecture encodes the source sequence reversing from an "ABC" sentence and give instead "CBA" to the target sequence which improved the BLEU score results from 25.9 to 30.6, especially on long sentences. This method exceeded the performance derived from the attention mechanism in its form introduced by (Bahdanau et al., 2015), nevertheless the architecture design does not allow an explanation of the predictions. For the interpretability analysis, architectures based on attention mechanisms may be more pertinent. In the following, we discuss different attention mechanisms, starting with the first proposed attention mechanism in machine translation known as soft attention, and then we discuss local attention. Both will be experimented with in our motion captioning task.

4.3.2 First attention mechanism

In the paper (Bahdanau et al., 2015), the authors incorporated a novel technique called the attention mechanism into the decoding process of an RNN. This technique enables the decoder to dynamically select relevant information from the input sequence, leading to *improved performance*, *robustness for long sequences*, and *interpretable results*. This attention mechanism has been formulated in various ways. In this discussion, we will focus on the original formulation. Notably, this mechanism has since been widely applied in different tasks and further developed by other researchers (Vaswani et al., 2017).

Formally, given an input sequence $\mathbf{x} = (x_1, \ldots, x_{T_x})$, $x_i \in \mathbb{R}^{K_x}$ and a target sequence $\mathbf{y} = (y_1, \ldots, y_{T_y})$, $y_i \in \mathbb{R}^{K_y}$, where K_x and K_y are the sizes of the vocabularies of the source and target respectively. The notations T_x and T_y denote the lengths of the source and target, respectively. The NMT task aims to map the input x to the target y. In the RNN Encoder/Decoder based approach, the context vector c and the hidden state $h_t \in \mathbb{R}^n$ are generally expressed as follows:

$$\begin{aligned} h_t &= f\left(x_t, h_{t-1}\right) \\ c &= q\left(\{h_1, \cdots, h_{T_x}\}\right) \end{aligned} \tag{4.11}$$

Where f and q represent model functions. (Sutskever et al., 2014) modeled the function f as an $LSTM$ and $q(\{h_1, \cdots, h_T\}) = h_T$. The decoder predicts the next

word y_t given the context described by the vector c and the vector representation of all previous words $\{y_1, \cdots, y_{t-1}\}$.

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) \tag{4.12}$$

This probability function in Equation 6.1 is modeled by g representing the decoding function which uses an RNN followed by a fully connected layer. Formally, we can write:

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c) \tag{4.13}$$

Attention-based context vector c_i. Computed using attention weights α_{ij} obtained by a softmax operation on energy coefficients e_{ij} as follows:

$$e_{ij} = v_a^\top \tanh\left(W_a h_{i-1} + U_a s_j\right) \quad \alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \qquad c_i = \sum_{j=1}^{T_x} \alpha_{ij} s_j \tag{4.14}$$

The alignment model is denoted as $\mathbf{a}(.)$, thus $e_{ij} = a(h_{i-1}, s_j)$, where s_j is j-th source hidden state. The parameters $v_a \in \mathbb{R}^{n' \times 1}, W_a \in \mathbb{R}^{n' \times n}$ and $U_a \in \mathbb{R}^{n' \times m}$ are learnable, with $m = n$ for the GRU and $m = 2n$ for the Bi-GRU. In the case of a bidirectional GRU, the source hidden state s_j is the concatenation of the j^{th} forward and backward encoder hidden states, such that for each encoding step j we have $s_j = [\overrightarrow{s_j}; \overleftarrow{s_j}]$ which gives the sequence of encoder outputs $(s_0, s_1, \cdots, s_{T_x - 1})$.

4.3.3 Local attention

This type of attention consists of using a subset of source hidden states (cf. Figure 4.13). There are many methods to find the subset to select at each decoding time step. In the work by (Luong et al., 2015), the authors propose a method based on the calculation of an alignment position p_t. To compute p_t two techniques were proposed: (i) *Monotonic alignment* which assumes a monotonic relation between the source and target sequences so $p_t = t$, and (ii) *Predictive alignment*, where the position p_t is learned by the model, the prediction is based on Equation 4.15.

$$p_t = T_x . sigmoid(v_p^T tanh(\mathbf{W_p} h_t)) \tag{4.15}$$

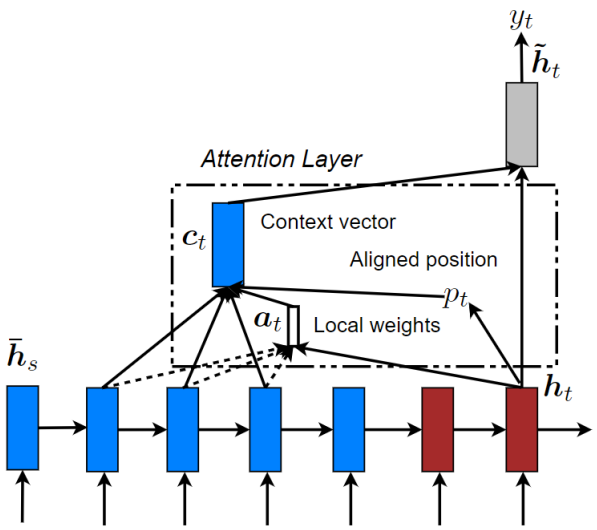


FIGURE 4.13: Schematic description of the process of encoding and decoding, which involve the computation of alignment position p_t (Luong et al., 2015).

Where T_x is the source length and $p_t \in [0, T_x]$. The learnable parameters are $\mathbf{W_p}$ and v_p. Then, the context vector c_t is calculated using the subset of tokens inside the range $[p_t - D; p_t + D]$, where $2D$ is the window length. Additionally, the previous weights α_{ij} are multiplied by a Gaussian window centered on p_t as described in Equation 4.16 with $\sigma = D/2$.

$$\hat{\alpha_{ij}} = \alpha_{ij}.\exp\left(-\frac{(i - p_t)^2}{2\sigma^2}\right) \tag{4.16}$$

Except for the calculation of α_{ij} the authors use the current hidden state h_t differently compared to the work (Bahdanau et al., 2015), where h_{t-1} was used. As a result, an attentional hidden state $\tilde{h}_t$ is calculated after the context vector c_t. Then, $\tilde{h}_t$ is passed through a linear layer with parameters W_p (cf. Figure 4.13). Finally, a $softmax$ function is applied to produce a probability distribution over the vocabulary size. These two operations are formally described by Equations 4.17.

$$\tilde{h_t} = \tanh(W_c[c_t; h_t]) \qquad p(y_t|y_{<t}, x) = \text{softmax}(\mathbf{W}_s\tilde{h_t}) \tag{4.17}$$

For the calculation of the coefficients e_{ij}, three formulations were tested by the authors, defined as follows:

$$e_{ij} = \begin{cases} h_i^T s_j & dot \\ h_i^T W_a s_j & general \\ v_a^T \tanh(h_i^T W_a s_j) & concat \end{cases}$$

The *dot* operation computes similarity without learnable weights. In contrast, the *general* formulation includes a learnable matrix, which generalizes the *dot* formulation. Lastly, the *concat* operation is equivalent to the attention model proposed by (Bahdanau et al., 2015). These attention mechanisms comes with an important possibility of *interpretability*. The explainability of predictions can be investigated through the visualization of attention weight matrix. This matrix describes the alignment between the source and target sequence. As soon as the idea of the attention mechanism emerged, other formulations and implementations of this mechanism have been developed, allowing significant improvements in overall NMT performances reached until 2015. Especially with the introduction of the Transformer (Vaswani et al., 2017), given the powerful *Multi-head self attention mechanism* incorporated in both encoder and decoder sides, as detailed previously in the Transformer (Section 4.2.4).

4.4 Methods

In this part, we present the pre-processing used for the KIT-ML dataset and provide an overview of the proposed architecture design with formal definitions of each attention block (Section 4.4.2). Then, we introduce the local recurrent attention mechanism (Section 4.4.3). Subsequently, we present the formulation of relevant metrics for quantitative evaluation (Section 4.4.4).

4.4.1 Dataset pre-processing

The KIT-ML [2] dataset (2016 version for comparable results with related work) is designed to map natural language descriptions to sequences of joint angles and positions. However, as a crowdsourced dataset, it suffers from several issues that make its use challenging. In particular, there are significant grammatical and

[2] https://motion-annotation.humanoids.kit.edu/dataset/

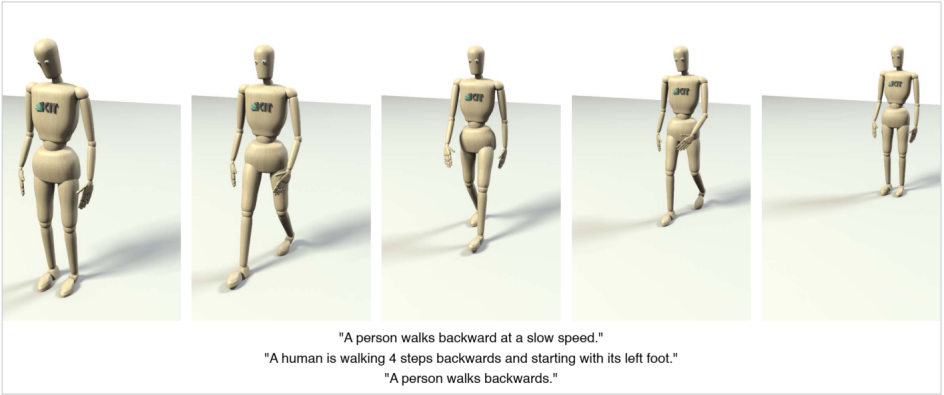


FIGURE 4.14: An example of walking motion with text references from the KIT-ML (Plappert et al., 2016).

spelling errors in the annotated text, which can hinder reliable generation. To address this, we apply automated spelling corrections to eliminate obvious and recurring errors. Additionally, we manually examine the correctness of resulting sentences. The KIT-ML dataset provides 44 joint angles for each motion frame, which we convert into Cartesian space using the script provided in Language2Pose (Ahuja and Morency, 2019). This conversion process allows us to represent the joint angles as 21 joint Cartesian coordinates. We apply normalization across the dataset and use the root coordinates of the first frame to shift each motion to the origin of the Cartesian system reference. The KIT-ML dataset comprises 3911 motions with 6728 natural language descriptions. However, multiple motions do not have any corresponding descriptions, this leads to a reduced number of motions, as shown in Table 4.1. The motions are captured at a frequency of $100Hz$, and we down sample them by a factor of 10 starting from the 5-th frame. We also limit the motions to those that are under 30 seconds, following the proposition of (Plappert et al., 2018). We exclude a one sample with invalid reference.

4.4.2 Proposed architecture

This section give details about our architecture designed for unsupervised learning of human motion semantic segmentation through the motion captioning task.

Global view. In this part we give an overview of the proposed architecture for motion-to-language mapping, and segmentation process, achieved through the association of intervals of frames to a set of words. The architecture operations are detailed through two figures: Figure 4.15 illustrating the different steps involved

in the synchronization between motion and language, and Figure 4.16 depicting the flow of model operations. The attention block is illustrated for the case of the proposed *local recurrent attention*, introduced in Section 4.4.3. A comparison with other cases of soft attention and local non-recurrent attention ($p_{t-1} = 0$) is discussed with visual details in Section 4.5.

1. **Encoder:** We experiment an GRU and MLP based encoder, we further compare these decoders and explain the limitation and advantages for motion analysis.

2. **Attention block:** We adapt two attention mechanism, soft and local attention to our task. Then discuss the limitations through attention map visualization. Then we proposed an enhanced attention mechanism: *Local recurrent attention*

3. **Alignment position:** In all following analysis, the predicted position p_t represents the most relevant frame corresponding to a given word at time step t.

4. **Context vector:** Both information of attention weights are used to compute the context vector. This dynamic vector represents the motion encoding relevant for the prediction of the next word.

5. **Decoder:** The GRU-based decoder serves to predict the next word, conditioned by the context vector and the previously generated words.

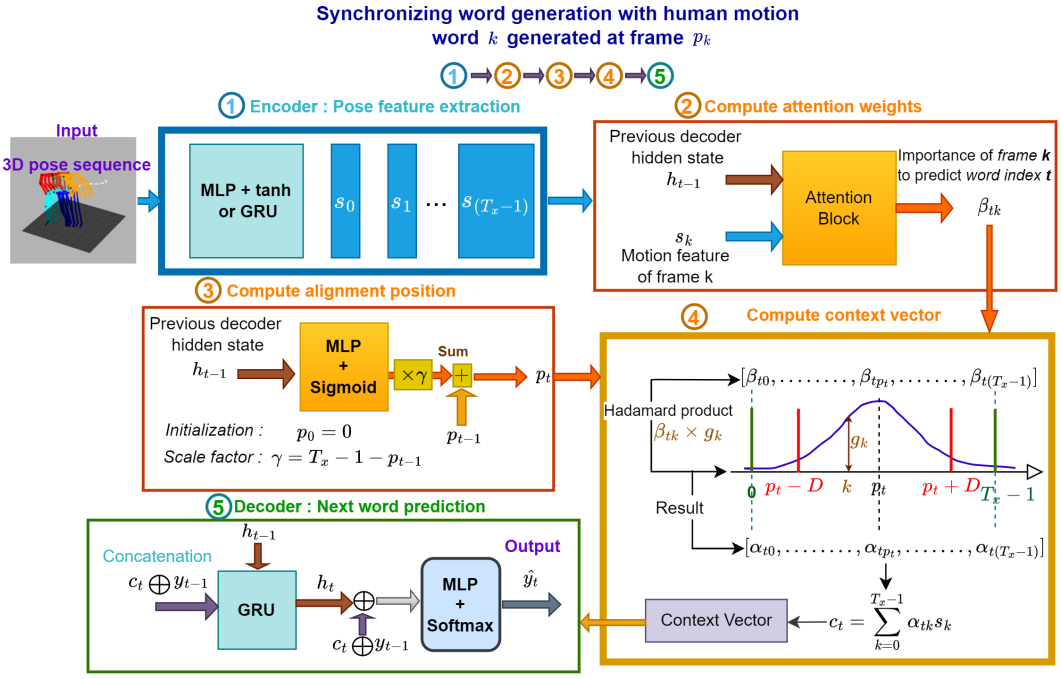


FIGURE 4.15: The five steps illustrating the motion to language generation (3D animation for synchronous generation of text with motion can be found in the Code).

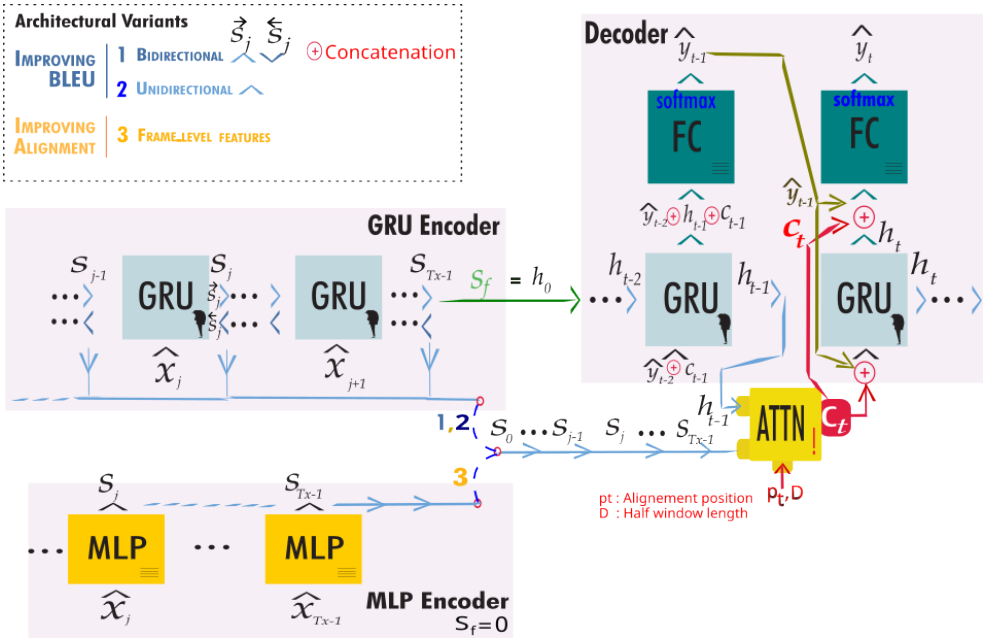


FIGURE 4.16: Experimented architectures, GRU, Bi-GRU and MLP-GRU (The word embedding layer is not represented).

Encoder. We have experimented three types of encoders: GRU, Bidirectional GRU (Bi-GRU) and Multilayer perceptron (MLP). We will formalize the sequence to sequence learning in the context of motion-language mapping. Lets's denote by x the motion input, and by y the natural language description. Firstly, we recall the main design architecture for this type of task. The typical diagram, in the case of seq2seq, consists of an Encoder and Decoder network. The two blocks are connected and trained to model the probability given in Equation 4.18.

$$p(y|x) = \prod_{t=1}^{|y|} p(y_t|y_{<t}, x) \tag{4.18}$$

Attention block. We present the formulation and results for each of the three attention mechanism, and discuss their relevance for human motion analysis, specially for segmentation process.

Soft attention. In this context of soft attention, the alignment position p_t is defined by the following Equation:

$$p_t = argmax_{j<T_x} \alpha_{tj} \tag{4.19}$$

We recall that T_x is the number of motion frames.

Context vector. Denoted as c_t, represents the weighted sum of the encoder outputs s_j and attention coefficients α_{tj}, such that $\boldsymbol{c_t} = \sum_{j=1}^{T_x} \boldsymbol{\alpha_{tj} s_j}$. The decoder learns to select important information by attributing higher weights α_{tj} to the most relevant encoder outputs s_j. These weights are computed as discussed in Section 4.3.2.

Local attention. As mentioned before, soft attention uses all source outputs, which results in a loss of the main source information that is relevant for the prediction of a given motion word. We propose reducing the utilization of all source outputs by attenuating and/or filtering the attention weights around the alignment position. These techniques are implemented through a masking strategy. When the mask is not enabled, the weights are simply multiplied by a Gaussian window centered around the position p_t, which attenuates attention weights assigned depending on the corresponding frame. When the mask is enabled, the attention weights out of the interval segment S_t, defined as $[\![p_t - D, p_t + D]\!]$ are discarded.

Noting by $s_x(j)$ the source hidden state at time $j \in [\![0, T_x - 1]\!]$ for a given motion x, the context vector is defined in the two cases through Equation 4.20. Where a_{tj} are attention weights computed using Equation 4.22.

$$c_t = \begin{cases} \sum_{j=0}^{T_x-1} a_{tj} s_x(j) & \text{Mask is } \textbf{False} \\ \sum_{j \in S_t} a_{tj} s_x(j) & \text{Mask is } \textbf{True} \end{cases} \tag{4.20}$$

Alignment position prediction. The position p_t is computed similarly to that of (Luong et al., 2015), the minor difference is that we use the previous hidden state h_{t-1} instead of the current hidden state h_t. Maintaining the same design idea in the work of (Bahdanau et al., 2015) for soft attention model. The Equation 4.21 gives the model formulation used to predict the position p_t, $\sigma(.)$ denotes the sigmoid function.

$$\boldsymbol{p_t = T_x.\sigma(v_p{}^T \tanh(W_p h_{t-1}))} \tag{4.21}$$

The trainable parameters consist of $W_p \in \mathbb{R}^{n' \times n'}$ and $v_p \in \mathbb{R}^{n' \times 1}$, where T_x denotes the length of the motion sequence. A Gaussian distribution that centers around p_t is utilized to prioritize alignment points in proximity to frame index p_t. The attention weights are adjusted using Equation 4.22. The variable factor $\exp(-(j - p_t)^2/2r^2)$ gives the amount of attenuation applied to the attention weights computed using the soft attention. As result a_{ij} are the new attention weights in the case of local attention.

$$a_{ij} = \alpha_{ij} \cdot \exp\left(-\frac{(j - p_t)^2}{2r^2}\right) \quad \text{where } r = \frac{D}{2} \tag{4.22}$$

The variable r denotes the standard deviation, which control the degree of attenuation of frame attention weights around the frame of index p_t. When r assumes smaller values, only few frames around p_t are employed to predict the next word. On the other hand, as r increases, the local attention mechanism gradually converges towards the soft attention mechanism. The parameter D represents the radius of the motion segment.

4.4.3 Local recurrent attention with frame wise encoding

In local attention, the position p_t depends only on the previous hidden state h_{t-1}, which is problematic in our case, we observe that the network simply learns to

almost always compute p_t at the start, middle, or at the end of the motion as will be highlighted in the experiments phase (Section 4.5). Moreover, the disorder in the alignment positions p_t doesn't allow synchronized generation of text with human motion evolution. To address these two problems of attention inconsistency and disorder, we propose the following two solutions:

1. **Solving the incorrect ordering of p_t positions**: To achieve synchronous and progressive generation solely using p_t, we impose a constraint on this alignment position such that $p_{t-1} \leq p_t$ for all time steps. While this constraint may not be universally applicable at the word level due to language-dependent variations, it holds true when considering phrases such as (*"person walks"*, *"then jumps"*...). For instance, certain words such as *"the"*, *"a"*, and *"person"* may not be directly monotonically linked to frames. However, for action-related words like *"walk"* or *"jump"*, there exists a temporal succession wherein the actions are performed successively. Consequently, the words used to describe these actions will also tend to appear successively in human descriptions. We hypothesize that a description is produced step by step as the motion is generated. The positions p_t will be more relevant for words that describe actions. The other intermediate positions, for language connection, will serve as relative positions from which the next action is localized.

   When we have simultaneity of two actions described respectively by words w_i and w_j, ideally, the model will learn to predict approximately same position alignment meaning that $\boldsymbol{p_i \approx p_j}$. This implies that the position p_t will remain approximately constant in the range $[\![i, j]\!]$. On the side of language generation, the words connecting the two action words will be independent of motion. *The main goal is to associate every set of words in the sentence describing one action to the relevant set of frames based on p_t and attention weights.*

   Formal definition. To overcome this limitation, we aim for the model to associate a subset of successive frames to the most relevant word(s) to achieve an exact alignment and synchronization of segments of human motion with the generated text. To this end, we propose a new formulation to compute the position p_t in Equation 4.23, enforcing the important constraint that $\boldsymbol{p_{t-1} \leq p_t}$ as explained in Paragraph 1.

$$p_t = p_{t-1} + \epsilon + (T_x - 1 - p_{t-1} - \epsilon).\sigma(v_p^t \cdot W_p h_{t-1}) \tag{4.23}$$

Where "$\cdot$" denotes the scalar product. The scaling factor $(\boldsymbol{T_x - 1 - p_{t-1} - \epsilon})$ maintains the position p_t within the range of the motion frames $[\![0, T_x - 1]\!]$. The position p_t indicates where to look in the source frames at a time t. The shift position $\boldsymbol{p_t - p_{t-1}}$ depends on the previous hidden state of the decoder h_{t-1}. For parameter D, we attempted to make it learnable through various formulations, but the values ended up being too high to allow for a correct segmentation in most cases, even when using a boundary function. However, when selecting its value as a hyperparameter we found a constant value for D leading to consistently better results.

2. **Frame wise feature extractor**: We opt to employ a motion encoder that computes motion features independently for each frame. The input, which is represented as a sequence of 3D human poses, can be encoded using various sophisticated methods that rely on Spatial Graph Convolutional Networks, as previously described in Chapter 3. However, due to the limited size of the dataset, we utilize a simple Multilayer Perceptron (MLP) instead.

   Given a motion $x \in \mathbb{R}^{T_x \times 63}$, the MLP encodes the sequence motion on a per-frame basis and produce the encoder outputs $s_x \in \mathbb{R}^{T_x \times d_{enc}}$ where d_{enc} is the dimension of the encoded vector. Here, d_{enc} is the same as the decoder hidden size ($d_{enc} = n'$).

   The MLP network consists of k successive non-linear layers as defined by equation 4.24, where $f_i(x(t)) = \tanh(W_i.x(t) + b_i)$. The learnable parameters are $W_i \in \mathbb{R}^{63 \times d_i}$ and $b_i \in \mathbb{R}^{d_i}$, tanh is the hyperbolic tangent, and where L is the number of layers.

$$s_x(t) = (f_0 \circ f_1 \circ \cdots \circ f_{L-1})(x(t)) \qquad \forall t \in [\![0, T_x - 1]\!] \tag{4.24}$$

4.4.4 Evaluation metrics

Bias in model evaluation using BLEU score. While BLEU is a standard metric, it only captures surface correspondence between the references and generated text. In the dataset, a sequence of motion is mapped to multiple references. If we take the case of two samples which have the same sequence of motion approximately and different sets of references. Considering as example the prediction *"a person wipes a table."*, and it references *"a human wipes on a table."*,*"a person stirs with their left hand."*. The prediction is counted correct only if the words are the same.

A description with the same meaning and a different formulation will have low BLEU, which doesn't correlate with human judgment. Consequently, the BLEU score doesn't compare semantic correspondence, to overcome this limitation we propose a semantic similarity metric, invariant under paraphrase, to evaluate the semantic adequacy of generated text compared to references.

Sentence semantic similarity score. We can devise a simple transformer-based metric, similar to a triplet-loss approach: by exploiting a sentence transformer (Reimers and Gurevych, 2019) model trained for paraphrase detection using triplet loss, we can compute a normalized cosine similarity score between each reference and the generated sentence. Since movement descriptions are composed of very common general-domain vocabulary, a generalist sentence embedding model trained to assign high scores to vector embeddings of paraphrased sentences, can be expected to perform very well at evaluating the semantic equivalence of the generated sentence with regard to the references.

$$score(RC, P) = \frac{1}{\mid RC \mid} \sum_{i \in |RC|} sim(RC_i, P_i)$$

$$sim(RC_i,P_i) = \frac{1}{\mid P_i \mid} \sum_{p \in P_i} \max_{r \in RC_i} sim_{cos}(emb(p), emb(r))$$

Where RC represents the reference corpus, with each $r_i \in RC$ being a set containing multiple references, and P is the predictions. $p_i \in P$ is a set containing one or more predictions. The function emb denotes the sentence-embedding model, which generates an embedding vector for a given sentence or phrase. The function sim_{cos} calculates the normalized scalar product between two vectors. Since references can vary significantly from one another, even when both are correct, we select the highest similarity score for a prediction compared to all corresponding references. Here, the sim_{cos} is utilized because the sentence transformer paraphrase model used in this context is trained using cosine similarity as a regression loss. It's worth noting that different sentence-transformer models may necessitate the use of alternative similarity metrics.

4.5 Experiments

Here we experiment our proposed methods in Section 4.4. First, we compare the performance of motion captioning based on pose joint angles versus the 3D Cartesian coordinates (Section 4.5.1). We report quantitative results (Section 4.5.2), and conduct a comparison with previous state-of-the-art methods (Section 4.5.3) on the original KIT-ML dataset. Then, we analyze the limitation of soft and local attention mechanism in the context of synchronized captioning (Section 4.5.4). Next, we experiment with the proposed local recurrent attention and frame-wise feature extraction (Section 4.5.5). Finally, we extend our evaluations to recent datasets (Section 4.5.6).

4.5.1 Comparing input features and evaluation measures

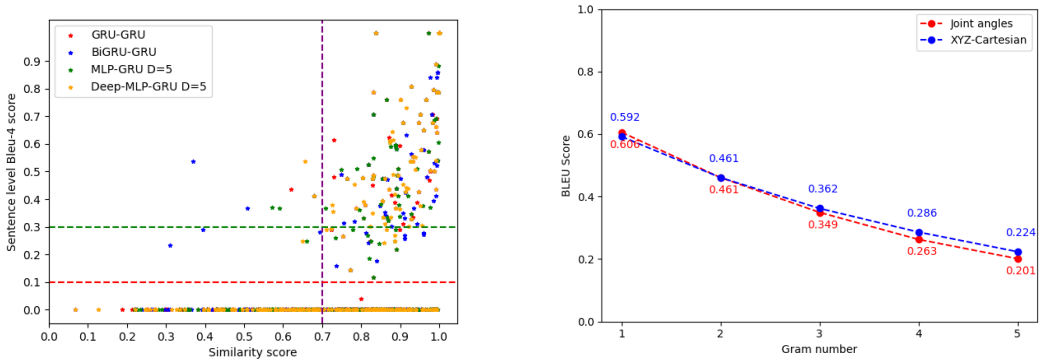
This section discusses preliminary experiments on the choices of input motion representation and evaluation metrics.

Cartesian coordinates vs Joint angles. We conducted an evaluation comparing the angle-based text generation model with the bidirectional and unidirectional Cartesian coordinate as inputs. The results for both input types are shown in Figure 4.17b. Using angle joints, we were able to achieve a BLEU score (4-gram) of 25.9%, whereas the use of Cartesian coordinates increased this performance to 30.1% on the test set. However, based on the semantic evaluation presented in Table 4.2, the difference is not significant. We observed that in most cases, the predictions are semantically correct, even if the BLEU-4 score is below 32%, which does not represent a fair and realistic evaluation of performance.

System	Soft.att	
	Similarity	BLEU score
GRU-Angles	79.50	28.5
BiGRU-Angles	78.90	26.3
GRU-Cartesian	**79.56**	**31.8**
BiGRU-Cartesian	78.84	28.6

TABLE 4.2: GRU-based system : Results of soft attention

BLEU vs similarity score. We plot all scores distributions of BLEU and our semantic similarity score obtained by the systems we evaluate in the following sections with optimal parameters in Figure 4.17a. The region of lower similarity and higher BLEU is approximately empty for all systems. Contrary to the opposite region (low BLEU, high similarity) which contain a lot of examples. For the example taken the system based on GRU, the region of lower BLEU < 0.1 and higher similarity (≥ 0.7) contain around 38.8% to 43% of samples for the three models most of them have approximately zero BLEU score which is far away from a realistic evaluation.



(a) Similarity vs sentence level Bleu score [Cartesian Local.rec.att Mask=True (D=5)].

(b) BiGRU+GRU: Joint angles vs Cartesian coordinates.

FIGURE 4.17: Impact of input features and evaluations measures.

Based on the analysis results of these two preliminary experiments, we proceed to present detailed experiments using the Cartesian representation and analyze the performance in terms of both BLEU score and similarity scores.

4.5.2 Quantitative evaluation

Results of soft attention. The Cartesian representation demonstrates superior performance across GRU and BiGRU systems, yielding higher similarity scores (78.87%) and significant BLEU-4 scores (30.0% and 31.8%, respectively) compared to the GRU-Angles system.

Results of local and recurrent attention. In Table 4.4, quantitative performance of GRU-based systems is presented, comparing the impact of local attention (Loc.att) and recurrent attention (Loc.rec.att) on similarity and BLEU scores. For GRU-Cartesian systems, when the mask is set to True, Loc.rec.att achieves the highest

System	Similarity	BLEU-4
GRU-Angles	79.40	25.9
GRU-Cartesian	78.87	30.0
BiGRU-Cartesian	78.87	31.8

TABLE 4.3: GRU/BiGRU results of BLEU scores (Radouane et al., 2023a).

similarity (80.54%), while Loc.att slightly lags behind (78.84%). However, in BLEU scores, Loc.att (30.0%) outperforms Loc.rec.att (29.7%). Without the mask (False), Loc.att exhibits a higher similarity (80.54%) compared to Loc.rec.att (78.34%), while Loc.rec.att yields a higher BLEU score (31.1% vs. 29.3%). Similar trends are observed for BiGRU-Cartesian systems, emphasizing the impact of attention mechanisms on system performance.

System	Mask (D=5)	Similarity		BLEU score	
		Loc.rec.att	Loc.att	Loc.rec.att	Loc.att
GRU-Cartesian	True	78.84	78.44	30.2	**30.0**
GRU-Cartesian	False	**80.54**	78.34	29.7	29.3
BiGRU-Cartesian	True	78.64	79.53	**31.1**	29.3
BiGRU-Cartesian	False	79.31	**79.58**	26.3	27.6

TABLE 4.4: GRU-based system: Results of local attention and local recurrent attention (Radouane et al., 2023a).

Results of local recurrent attention with an MLP based encoder. Table 4.5 displays the performance of MLP-GRU with varying design parameters, including the number of layers (L), the presence of a mask, and the window radius (D). Notably, the results indicate a positive influence of the mask on both BLEU and similarity scores, while an increase in the radius generally corresponds to a decrease in performance. The MLP-GRU configuration with 4 layers (L=4), with the mask enabled, and $D = 5$ stands out as the most successful, achieving the highest BLEU score of 32.1% and a similarity score of 78.29%.

Decoding strategies. In addition to attention type and architecture, the quality of generated text is also impacted by the decoding approach and its corresponding parameters. Therefore, we compare two different decoding approaches to further investigate their impact.

Design	Mask	D	BLEU	Similarity
MLP-GRU (L=4)	True	5	**32.1**	78.29
MLP-GRU	True	5	**31.6**	**78.87**
MLP-GRU	False	5	28.8	78.29
MLP-GRU	True	9	29.1	78.52
MLP-GRU	False	9	28.4	76.38

TABLE 4.5: Results of MLP with 2 layers, except for the case $(L = 4)$ we use 4 layers.

Action	References	Predictions (beam size=3)
Kneeling	Human ducks down and takes her hands over her head	a person lies *on all fours* and stands up
		a person lies *on the ground* and stands up
		a person lies *on the floor* and stands up
Stomping	Human is lifting his leg	a person is stomping with the right foot
		a person is stomping with the left foot
		a person is stomping with his right foot
Squatting	A person stretches their arms out and performs *a single squat*	a person performs a squat
	A person does some squad exercises	a person performs a squat squat
		a person performs a single squat
Walking	Slow walking motion	a person walks 4 steps forward
	A person walks forward slowly	a person walks *slowly* 4 steps forward
	Human slowly goes forward	a person walks forward
Waving	A person raises his right arm in front of his face then starts waving	a person waves with his left hand
	Human performs a waving motion with right hand	a person waves with his right hand
	A person waves with its right hand	a person waves with the left hand

TABLE 4.6: Predictions of model MLP-GRU (D=5) with a beam size of 3.

Greedy search: consists in selecting the word from the vocabulary, maximizing the probability at each step independently, rather than evaluating all combinations.

Beam-search: selects the best decoding by considering the first k maximum probabilities at each prediction time (*top-k*), where k represents the beam width *Beam-width*. This evaluation method allows selecting and classify the best k translations, but tends to favor the selection of short sentences. To address this issue, the log-propabilities are normalized with sentence length. This decoding strategy often improves performance but increases prediction computation time, which becomes particularly significant as the width k of the *Beam* increases.

Beam-search vs. Greedy search. In Table 4.6, we present a sample of predictions for various actions. It is evident that the model has learned a coarse classification of different meanings: *(on all fours, on the ground, on the floor)*. For the *Stomping* and *Waving* actions, we observe that the first and second predictions focus on which body part is executing the motion (left or right/leg or hand), but the first prediction is correct. In the case of the *walking* action, we notice that information about the speed of motion is only present in the second prediction.

Model	D [Mask]	BLEU score (beam size)				
		1	2	3	4	6
MLP	5 [True]	**31.6**	30.6	30.5	30.7	30.6
	5 [False]	28.8	29.7	**30.7**	30.4	30.7
	9 [True]	29.1	30.4	**30.4**	30.1	30.1
	9 [False]	28.4	**29.2**	28.8	28.9	28.6
Deep-MLP	5 [True]	**32.1**	29.4	29.1	29.1	29.8

TABLE 4.7: Impact of beam searching on MLP based models (Radouane et al., 2023a).

We confirm that applying beam search with different beam sizes does not lead to an improvement in the BLEU score for MLP-based models. Additional results are provided in Table 4.7. The best BLEU score is already achieved without beam sampling, indicating that there is no need to perform beam search in the prediction process, which is computationally expensive.

4.5.3 Global comparison

Comparison with SOTA on KIT-ML (Plappert et al., 2016). After evaluating various architectural variants within our study, we now proceed to compare our results with systems described in the literature. However, conducting this comparison posed challenges due to discrepancies in dataset splitting methods among different authors. Although the proportions of the dataset are consistent, precise information regarding the specific random seed used for splitting is often not reported. To ensure a fair comparison, we not only report the performance of state-of-the-art (SOTA) systems but also evaluate our models using three different random seeds. This evaluation aims to verify that the performance variations caused by different seeds are minimal, with differences of less than 0.01. This approach allows for some level of comparability with the SOTA systems. The results are reported in Table 4.8. Across all random splits, our BLEU score is higher than reported systems by a margin significantly larger than seed-related variations.

Unknown token replacement. An additional parameter to consider is the influence of unknown tokens on the BLEU score. We restrict the vocabulary to words with a minimum frequency of 3. This is particularly important for the KIT-ML crowdsourced dataset, as it allows us to filter out noisy and incorrect contextual words from the vocabulary. This leads to more stable training as rare words, which

Random state	Model	BLEU score 1st	2nd	3rd	Avg
0	MLP	0.286	0.272	0.267	0.275
	DeepMLP	0.286	0.298	0.262	**0.282**
35	MLP	0.291	0.274	0.267	0.277
	DeepMLP	0.297	0.268	0.270	**0.278**
42	**MLP**	0.304	0.286	0.265	**0.285**
	DeepMLP	0.256	0.265	0.255	0.259
_	GASSL(LSTM) (Goutsu and Inamura, 2021)	0.288	0.263	0.257	*0.269*
_	GASSL(GRU)(Goutsu and Inamura, 2021)	0.262	0.243	0.231	*0.245*
_	BiLSTM (Plappert et al., 2018)	0.259	0.237	0.249	*0.248*

TABLE 4.8: Comparison with other models on different level of beam searching (Radouane et al., 2023a).

Model	Replace unk	BLEU score (D [Mask]) 5 [True]	5 [False]	9 [True]	9 [False]
MLP-GRU	Before	31.6	28.8	29.1	28.4
MLP-GRU	After	31.4	28.8	29.1	28.4

TABLE 4.9: Impact of using the unknown token on the evaluation.

may also be unreliable, are filtered. We perform evaluation after replacing the unknown token by corresponding words, the performance of Deep-MLP remain unchanged 32.1%. We confirm this stability on the other models as reported in Table 4.9. This process has no significant effect on the BLEU score in our case. Consequently, it doesn't impact the comparisons made before.

4.5.4 Qualitative analysis

This section discuss the drawbacks of previous attention mechanisms. We perform a qualitative analysis of attention maps using Cartesian coordinates to illustrate the limitations of both soft and local attention. This analysis motivated the introduction of local recurrent attention and the use of frame-wise feature extractor, as mentioned earlier (Section 4.4.3). In the following, the mask is assumed to be enabled by default, unless explicitly stated otherwise.

Limitations of soft attention. The context vector in soft attention utilizes all source outputs s_i, thereby enabling the decoder to access information about the past and

future of the motion. This computation approach is inconvenient for precisely localizing the motion corresponding to a given set of words in the description. The attention block has learned to utilize compressed information. In the case of a GRU-based encoder, the maximum position of information used is mostly at the very end of the motion. This behavior is demonstrated in Figure 4.18.

FIGURE 4.18: GRU-cartesian[Soft.att]: Pushing action in the range $[\![11, 20]\!]$.

For the Bi-GRU based encoder, the maximum attention is mostly towards the middle of the motion. We observe this results in multiple figures (4.19,4.20). Specifically, in Figure 4.19, the attention weights give only partial information about the range of motion $[\![27, 36]\!]$ with imprecise timing for the start and end of each of primitive motion (*walk forward*, *turn*, *walk backward*) composing the *U-turn* motion.

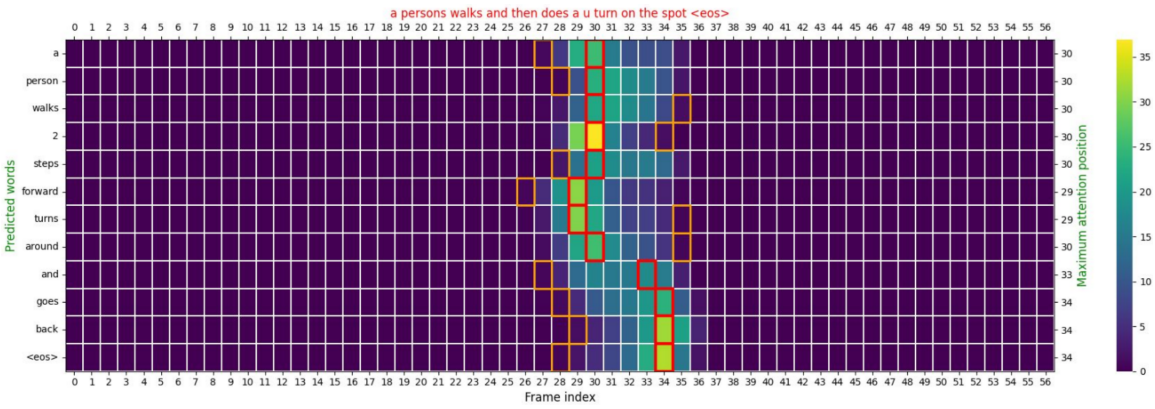


FIGURE 4.19: BiGRU-cartesian[Soft.att]: Walk forward in the range $[\![0, 28]\!]$, Turn around $[\![29, 35]\!]$ and Walk backward $[\![36, 56]\!]$.

While in Figure 4.20, the distribution of attention weights gives correct information on the range of the motion. This observation is common across different samples with single actions.

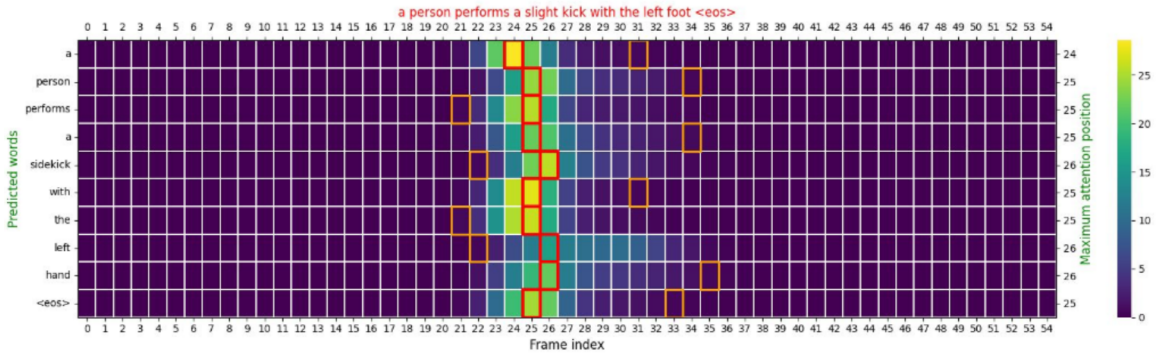


FIGURE 4.20: BiGRU-cartesian[Soft.att] : Kick in the range $[\![20, 35]\!]$.

Local attention analysis. We empirically set the value of D to 5. Figures 4.21 and 4.22 represent the cases of atomic and compositional motion, respectively, when the mask is not enabled. These distributions still only provide an approximation of the range of motion execution. In the case of compositional motion as shown in Figure 4.22, it demonstrates clearly a lack of information regarding the time transition of atomic motions. This analysis is particularly focused on the attention weights of words describing motion, such as *"kick"* and *"forward"*. These observations are similar for multiples samples.

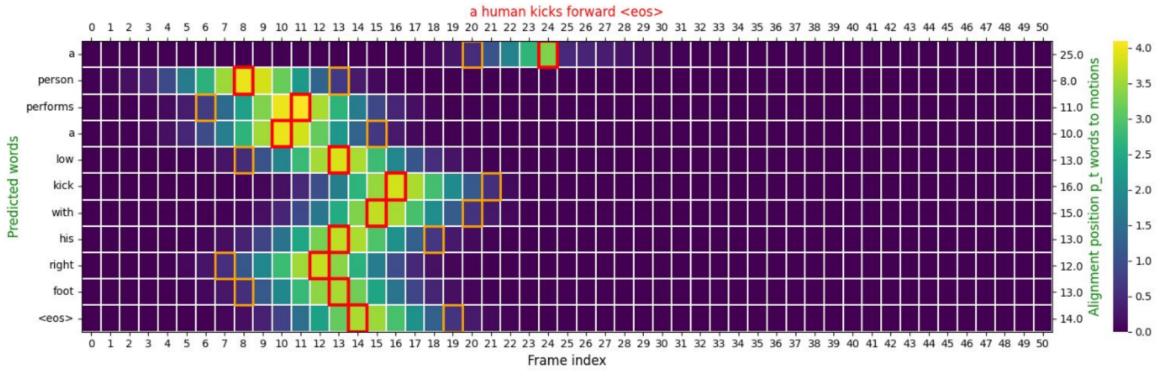


FIGURE 4.21: BiGRU[Local.att Mask False]: Kick action in the range $[\![10, 28]\!]$

Figures 4.23 and 4.24 represent the cases of atomic and compositional motion, respectively, when the mask is enabled. In both cases, the distribution of attention weights is towards the end, similar to the case of soft attention (cf. Figure 4.18). In this configuration, we lose all information about the motion's time execution, making it impossible to infer the correct end and start times.

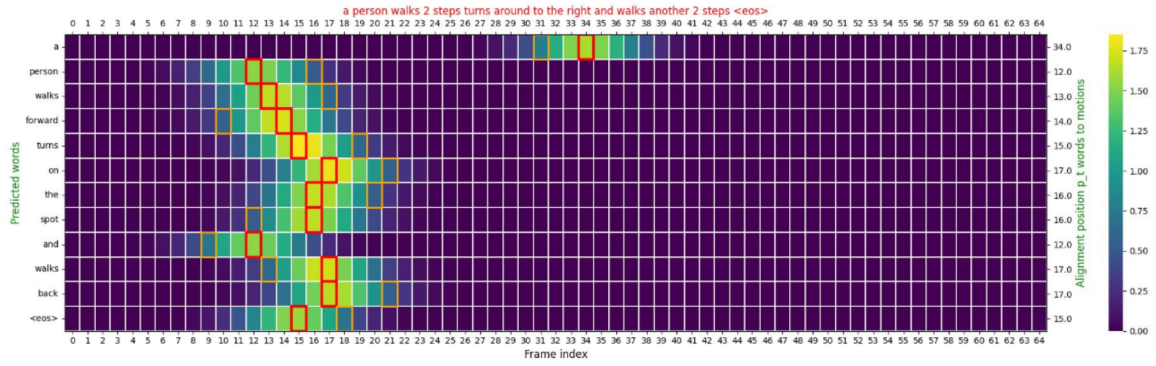


FIGURE 4.22: BiGRU[Local.att Mask False]: Walk forward $[\![15, 38]\!]$, Turn around $[\![39, 48]\!]$ and Walk backward $[\![49, 53]\!]$

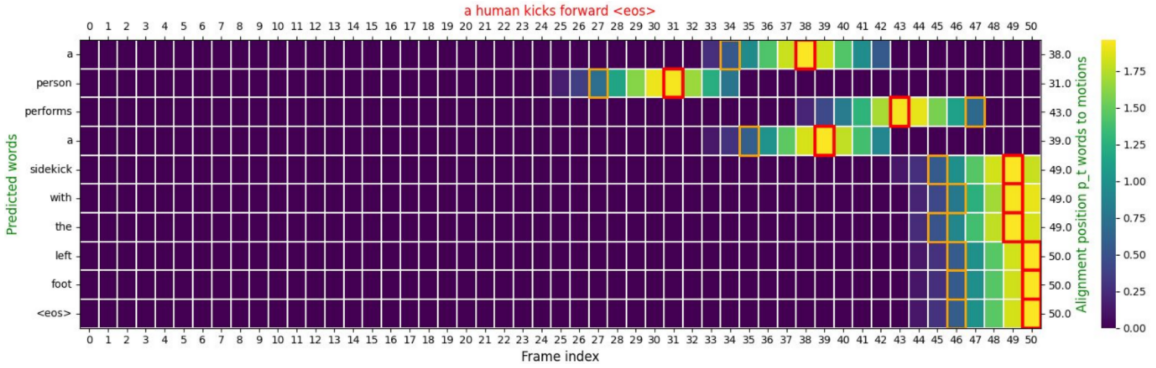


FIGURE 4.23: BiGRU[Local.att] kick action in the range $[\![10, 28]\!]$

Soft vs local attention in synchronization. In the case of the BiGRU-Cartesian model, the attention map depicted in Figure 4.19 exhibits a lack of consistency in the position of the maximum attention, resulting in unsynchronized word generation. Conversely, the unidirectional GRU model displays a clear maximum attention peak (cf. Figure 4.18) at the end frames. This observation is consistent with the attention maps of the test set, indicating that the highest attention distribution is typically located at the frame marking the completion of the movement. This again is far from the actual action time execution. When replacing soft attention with local attention, the problem is still not solved regarding the human motion segmentation. However, when it comes to the BLEU score, we note some differences in the performances of each model.

4.5.5 Local recurrent attention maps

Previously, the BiGRU is biased more towards the start and middle of the motion in Figure 4.20 (human movement ends at the frame 27), while a unidirectional GRU

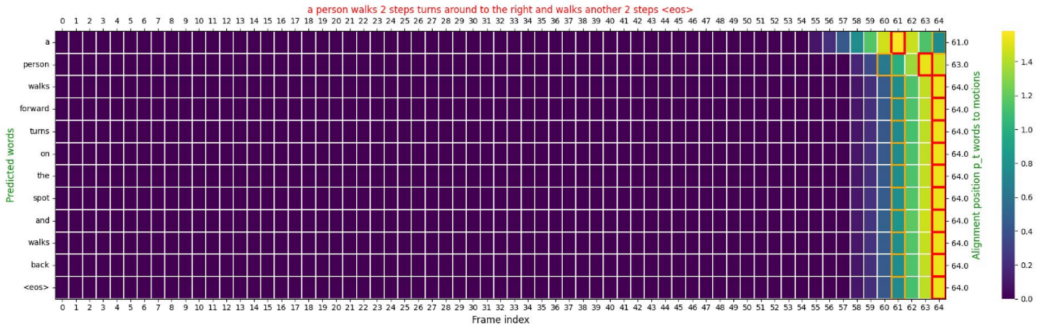


FIGURE 4.24: BiGRU[Local.att]: Walk forward $[\![15, 38]\!]$, Turn around $[\![39, 48]\!]$ and Walk backward $[\![49, 53]\!]$

favored the positions after the end of the human movement in Figure 4.18. Intuitively, the network relies on compressed information about the entire sequence and outputs a prediction when the amount of information is maximal on average. Consequently, resulting attention maps can't provide relevant information to perform synchronous predictions. Indeed, the network chooses the path of least effort without searching along the source length for finding the most relevant motion interval to output for a given phrase.

Qualitative analysis using Bi-GRU/GRU based encoder. We present some attention maps of representative cases in Figure 4.25. The recurrent attention mechanism performs an ordered attention, but a problem of late detection of the phases of motion remains unsolved. The late detection is present in most of the test samples. When enabling the mask, the alignment position is always at the end of the motion. The unidirectional GRU also suffers from the same issues as demonstrated by Figure 4.26. In some rare cases, as shown by Figure 4.27, it seems to give a correct localization for the single action *"Wiping"*.

Regarding Figures 4.25, 4.26, and 4.27, we conclude that introducing local recurrent attention constraints an ordered alignment position prediction but still does not produce correct action localization through attention weights, except for very few cases which do not include compositional motion. In the following, we build a theoretical explanation of these phenomena that we apply further, and we demonstrate their effectiveness subsequently.

Explaining the unsynchronized attention weights distribution. During qualitative examination, the observed alignments produced by different attention formulations in the initial experiments using the GRU architecture show imperfect

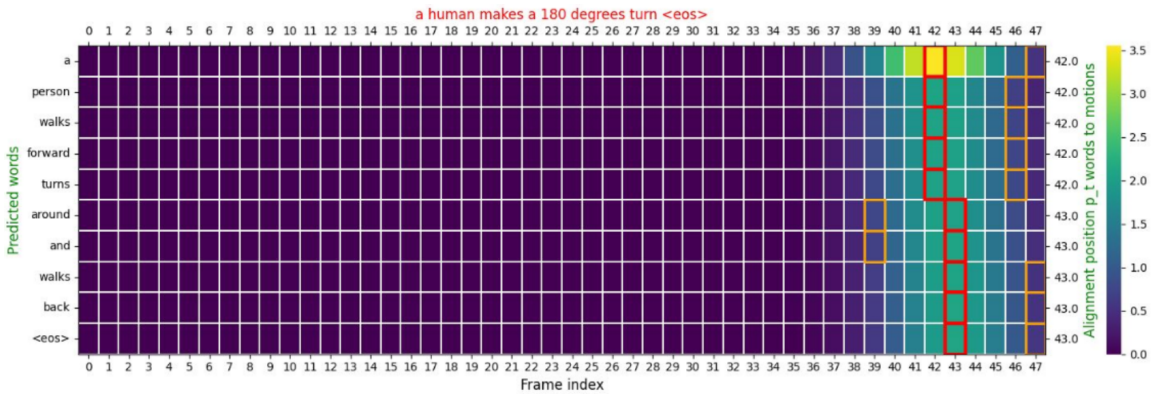


FIGURE 4.25: BiGRU[Loc.rec.att][Mask False]: Walk forward $[\![0, 21]\!]$;U-turn $[\![22, 34]\!]$, Walk backward $[\![35, 47]\!]$

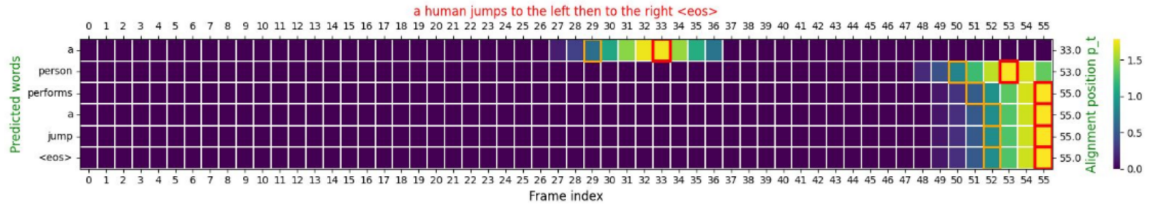


FIGURE 4.26: GRU[Loc.rec.att]: Jump to the left $[\![0, 31]\!]$,Jump to the right $[\![32, 54]\!]$

synchronization. Our hypothesis attributes this to the recurrence effect within the GRU design, where the encoder's hidden state s_t relies on all preceding states. Formally, we can write $s_t = \varphi(s_{t-1}, x_t) = \varphi(\cdots \varphi(s_{t-\overrightarrow{k}}, x_{t-\overrightarrow{k}}))$. We note by φ the classical GRU model equation and by $\overrightarrow{k}$ and $\overleftarrow{k}$ respectively the limit level of compression without important information loss for the forward and backward direction. Therefore, in the context of a unidirectional scenario, the hidden state s_t already contains all the essential information regarding the $\overrightarrow{k}$ preceding steps $\{s_{t-\overrightarrow{k}}, s_{t-\overrightarrow{k}+1}, \cdots, s_{t-1}\}$. Assigning maximum attention to s_t is analogous to perceiving the $\overrightarrow{k}$ preceding motion frames: an event occurring around frame time $t - \overrightarrow{k}$ can possess significant consolidated global information that is relevant to the frame t. The subsequent explanation provides a clear understanding of both the misaligned position and the delayed detection of primitive motion, as was illustrated by Figure 4.18.

For the Bi-GRU case, the analysis is similar, except that s_t now encodes all the necessary information about the $\overrightarrow{k}$ previous steps and $\overleftarrow{k}$ next information, this explains the localization of attention weight distribution towards the middle of motion segments (cf. Figure 4.20).

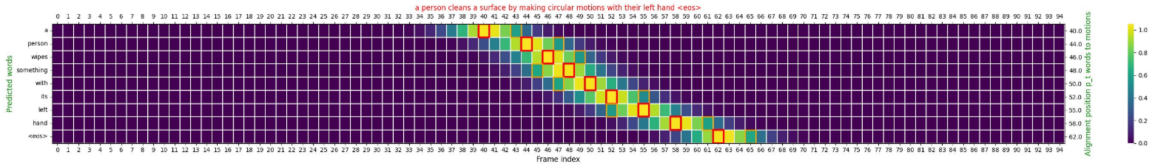


FIGURE 4.27: GRU[Loc.rec.att][Mask False]: wiping $[\![25, 56]\!]$

Based on this reasoning, we have concluded that employing a frame-by-frame feature extractor would effectively address the issue of delayed detection. Extracting features on a per-frame basis encourages the network to utilize all individual features of the source sequence.

The adoption of this type of feature extraction, matches similar choices in other works pertaining to alignment tasks, such as sign language/movie subtitle transcription (Bull et al., 2020, 2021). Although, quantitative evidence does not support the choice. The main difference in our approach is that we do not have a supervised timing annotation of movement phases/primitives.

Hence, the main innovation in our work (cf. Equation 4.23) is the ability of inferring a synchronous alignment only through attention weights using positions p_t.

Qualitative analysis with MLP based encoder. After replacing the encoder with an MLP of 2 layers, we start to observe separable distributions for each phase of the motion, as depicted in Figure 4.28. However, using an MLP with 4 layers (DeepMLP), the model extracts stronger features, providing a better semantic description, especially in the case of compositional actions as illustrated in Figures 4.29 and 4.30.

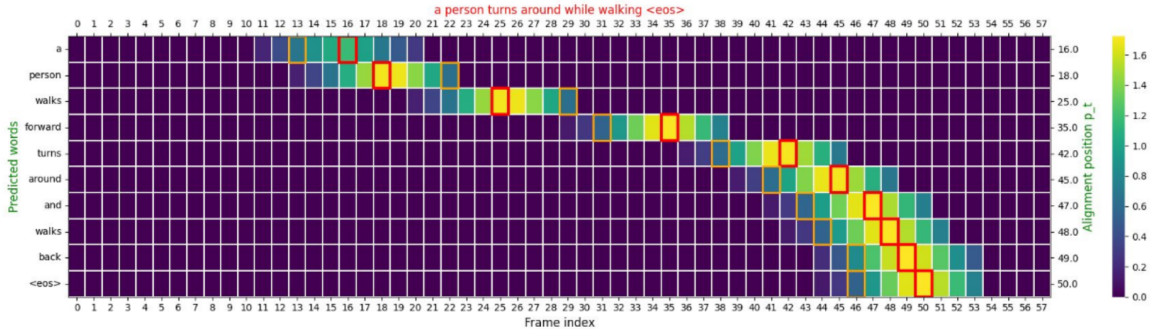


FIGURE 4.28: MLP[Local.rec.att]: Walk forward $[\![0, 30]\!]$, Turns around $[\![31, 40]\!]$ and Walk backward $[\![41, 50]\!]$

FIGURE 4.29: DeepMLP[Local.rec.att]: Walk forward $[\![0, 37]\!]$, pushed to the right $[\![38, 50]\!]$

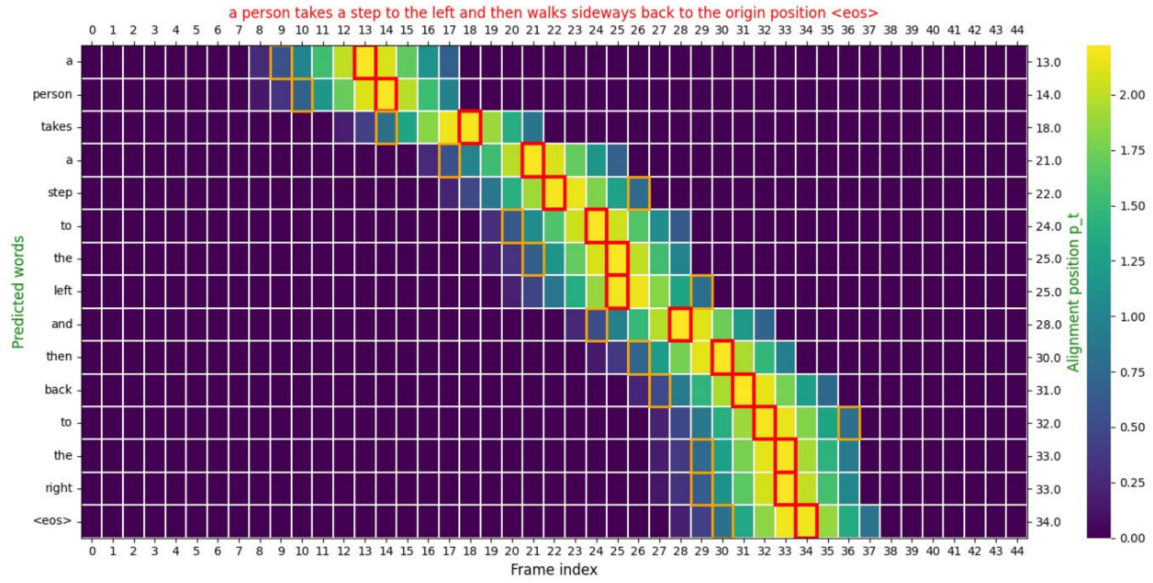


FIGURE 4.30: DeepMLP[Local.rec.att] $D = 5$: Move to the left in the range $[\![8, 23]\!]$, move to the right in the range $[\![24, 35]\!]$

Effect of the Mask window. When the mask is enabled, a truncated Gaussian window is applied in each decoding step t. When the mask is disabled, no truncation is performed, and the alignment is bad in several experiments (see Section 5.5.2), along with a lower BLEU score ($<$ 29%) as reported in Table 4.5. Alignment was not possible without applying the mask, which acts as regularizer during training and reduces overfitting. The strict limitation of visible encoder outputs, along with frame-level feature extraction, pushes the network to learn a precise position solely through semantic motion segmentation based on the text. Another important factor is the D value, for $D = 9$ applying the mask doesn't affect the alignment, and in general when D is high, masking has almost no effect.

Summary. Our experiments demonstrate that models utilizing recurrent encoders rely more on global information as a simple way to make correct predictions instead of learning motion segmentation, especially for short-time motion samples. Training these architectures is easier and leads to faster convergence towards higher BLEU scores. However, the distribution of attention weights along the frames axis does not align with the intuitive semantic segmentation observed through human analysis. Introducing recurrent local attention and using a simple MLP as an encoder encourages the network to improve the BLEU score only by learning a correct synchronized semantic segmentation. The convergence for such designs takes longer as the decoder is limited to the visible encoder outputs (finding the correct position p_t takes longer).

4.5.6 Evaluation of proposed architecture on recent datasets

To further investigate the generalization capability of our method, we compare it to a Transformer baseline approach. We utilize the same split on which we conducted all previous experiments. After our initial experiments, an augmented version of the KIT-ML dataset was released by (Guo et al., 2022a). Therefore, we extend our experiments to include this new version. Moreover, we adapt the proposed architecture to handle much larger datasets such as HumanML3D (Guo et al., 2022a). Consequently, we retrained both our model and the best Transformer baseline and compared them with the state-of-the-art model TM2T (Guo et al., 2022b).

Comparison with Transformer baseline. To investigate the efficiency and sufficiency of our architecture design with *local recurrent attention*, compared to more advanced architectures such as Transformer (Vaswani et al., 2017).

Training configurations: The original Transformer (Vaswani et al., 2017) uses $L = 6$ layers and $H = 8$ heads for both the decoder and encoder. However, in our context, we have a small-sized dataset. Instead, we run two configurations: a minimal one with $(H = 1, L = 1)$ and another with $(H = 4, L = 3)$. We use the same split on which we conducted all previous alignment experiments.

Hyperparameters for KIT-ML-original (Plappert et al., 2018): Similar to the hidden size of our `MLP-GRU`, we set $d_{model} = 64$ for the query, key, and value projection dimensions. The batch size was set to 128, and the motion samples were downsampled from $100Hz$ to $10Hz$. The corresponding split was generated as described before, with a Random Seed (RS=11). In the case of the Transformer model, for all

splits and dataset versions, the maximum length is fixed to 500 for the positional-wise feed-forward layer.

Hyperparameters for KIT-ML-augmented (Guo et al., 2022a): For *KIT-ML*, we set $d_{model} = 128$, the batch size to 256, and downsampled the motion from $20Hz$ to $10Hz$. The other model hyperparameters remain the same for this augmented version.

Hyperparameters for HumanML3D (Guo et al., 2022a): For this larger dataset, the word embedding dimension is set to 128 and the hidden size to $d_i = 256$, while the hidden dimension of the `MLP` with 2 layers have respectively an output dimensions of 512 and 256 (represent d_{enc}) and $D = 10$.

Transformer cross-attention. One interesting aspect of the first experimented configuration $(H = 1, L = 1)$ is the possibility of easily visualizing the learned cross-attention maps without the need for advanced aggregation methods per heads and layers. Although this architecture gives a BLEU score of 32.2 versus 32.1 (cf. Table 4.10) using our `DeepMLP`, the attention weights do not permit semantic segmentation. For both single action (cf. Figure 4.31) and compositional motion (cf. Figure 4.32), the cross-attention does not exhibit segmentation and motion localization capability. This is likely due to the mixture of frame information introduced earlier by the self-attention mechanism in the encoder.

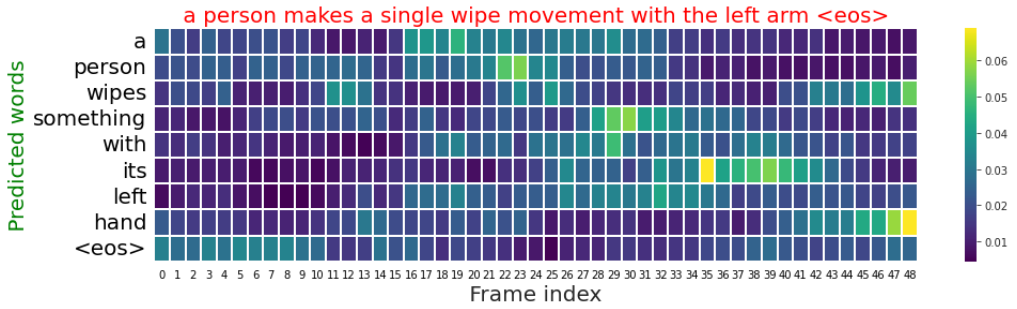


FIGURE 4.31: Transformer [H1L1] cross attention visualization (*Wiping*).

Quantitative comparison. In Table 4.10, we report BLEU scores for different splits and dataset versions, comparing the Transformer and GRU-based generation. We evaluate both the quality of text generation and the ability to perform implicit motion segmentation. Despite the close BLEU score performance after tuning our

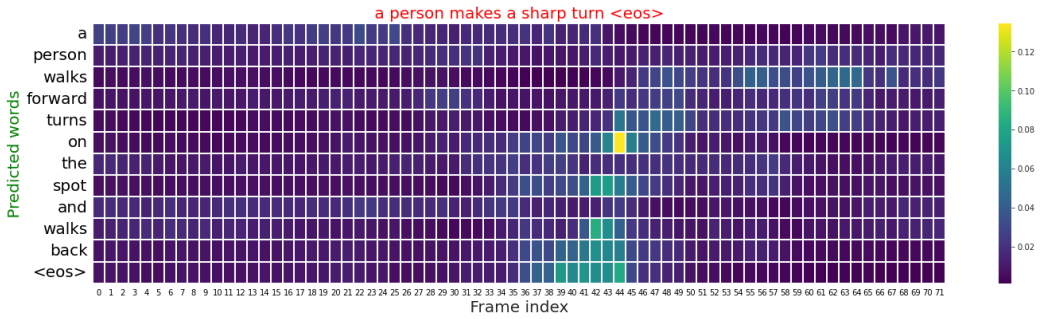


FIGURE 4.32: Transformer [H1L1] cross-attention visualization (*U-turn*).

	BLEU@4		Semantic Segmentation
Dataset	**Original**	**Augmented**	
Splits	RS=11	Standard	
T[H1,L1]	**32.2**	24.8	✗
T[H4,L3]	30.8	22.1	✗
Ours MLP+GRU	31.6	25.0	✓
Ours DeepMLP+GRU	**32.1**	**25.1**	✓

TABLE 4.10: Comparison with baselines on different splits. *RS* stand for the random state value for the original KIT-ML.

transformer's hyperparameters, the learnable attention weights by the transformer are not useful for motion segmentation and are not very interpretable.

Comparison vs SOTA. In Table 4.11, we report BLEU scores for different splits and dataset versions, comparing the Transformer and GRU-based generation. Our model based on the proposed local recurrent attention achieve better performance than the SOTA transformer based model TM2T (Guo et al., 2022b) and more importantly we preserve the semantic segmentation capability.

Dataset	Model	BLEU@1	BLEU@4	CIDEr	ROUGE-L	BERTScore	Segmentation
KIT-ML	TM2T (Guo et al., 2022b)	46.7	18.4	44.2	79.5	23.0	✗
	Ours MLP+GRU	**56.8**	**25.4**	**125.7**	**58.8**	**42.1**	✓
HumanML3D	TM2T (Guo et al., 2022b)	61.7	22.3	49.2	**72.5**	**37.8**	✗
	Ours MLP+GRU	**67.0**	**23.4**	**53.7**	53.8	37.2	✓

TABLE 4.11: Comparison with SOTA results on the augmented version of KIT-ML and HumanML3D (Radouane et al., 2023a).

4.6 Application: Synchronized Captioning

As a direct application of the proposed *local recurrent attention*, the alignment positions can naturally be depicted as time for words generation. This enables a synchronous generation of language alongside motion, referred to as *synchronized captioning*. Recalling that: $p_t = p_{t-1} + (T_x - 1 - p_{t-1}).\sigma(v_p^t \cdot W_p h_{t-1})$

In the following Figures, the visualizations are made using MLP-GRU with the config Local.rec.att [Mask True $D = 5$]. A word w_t can be generated at time frame p_t (cf. Figure 5.6). However, the importance of attention weights spans the range $[p_t - D, p_t + D]$. Consequently, using p_t could sometimes result in outputting a word at the middle or end of the motion primitive, and using $p_t - D$ could lead to generating the action word just before the start of the action. For instance, as shown in Figure 4.34a, the words *"takes a step to the left"* are generated near the end of the movement to the left, while the words *"back to the right"* are generated near the end of the movement to the right. Similar behavior can be observed in other compositional actions, as illustrated in Figure 4.34c.

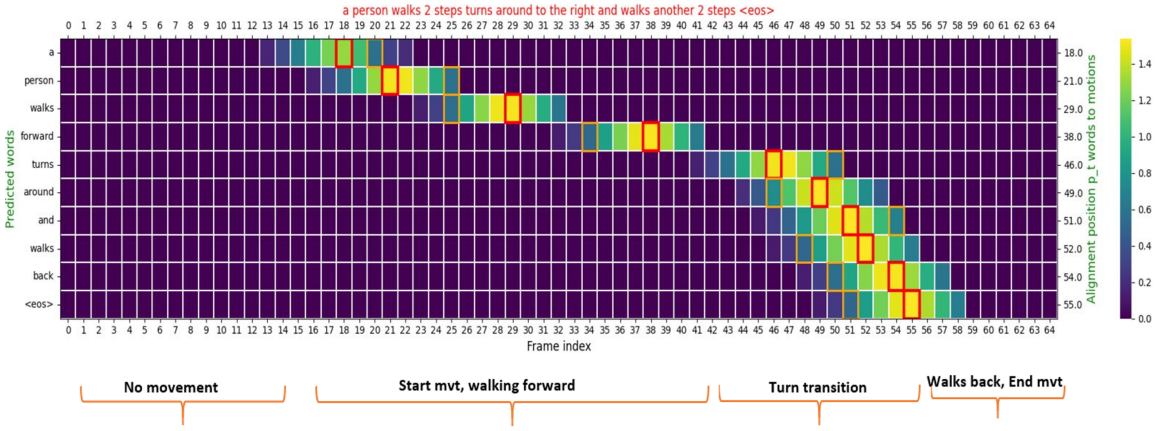
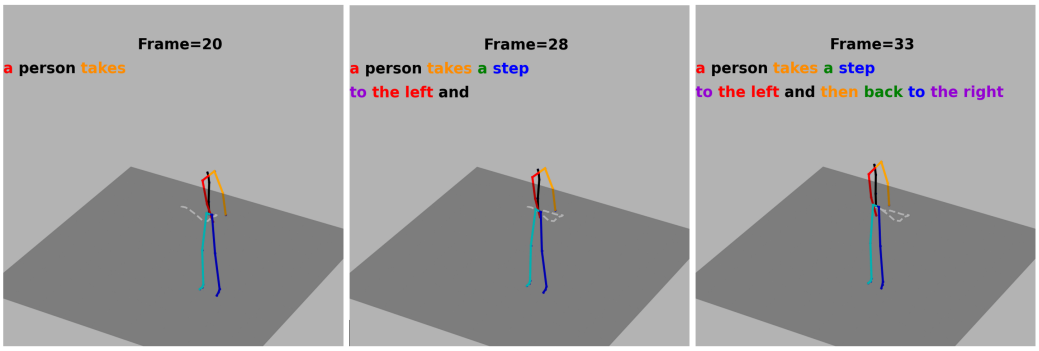
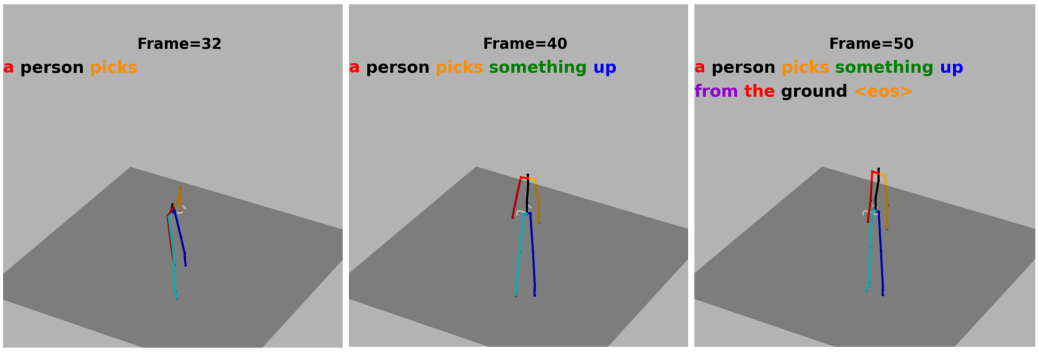


FIGURE 4.33: walk forward: $[\![15, 38]\!]$, turn at frame 40 , walk backward: $[\![42, 53]\!]$.

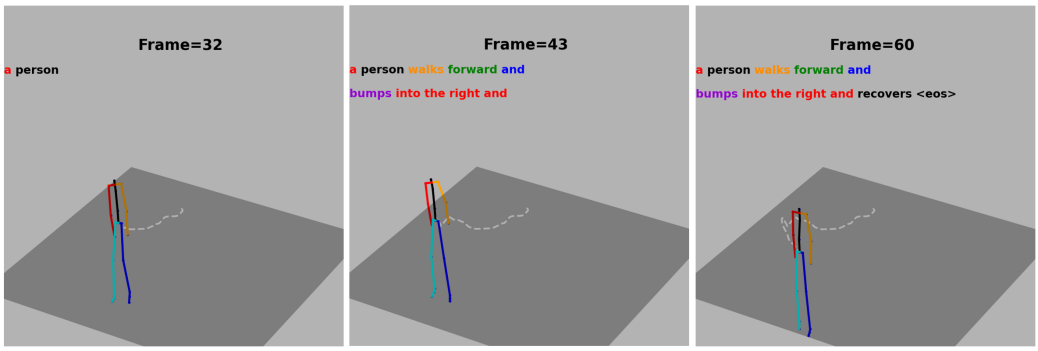
Although this behavior is still correct from a human analysis perspective, as the understanding of motion is accomplished at its end, for visual convenience, we could apply a static shifting of $D/2$ as a trade-off to achieve more visual synchronization. However, the generation is still adequate at this stage. Figure 4.34 show frozen frames. ***We note that each textual description displayed at a frame F describes the whole motion in the range*** $[\![0, F]\!]$. These samples and others illustrating synchronization between text and motion can be better visualized as animations on the project page.



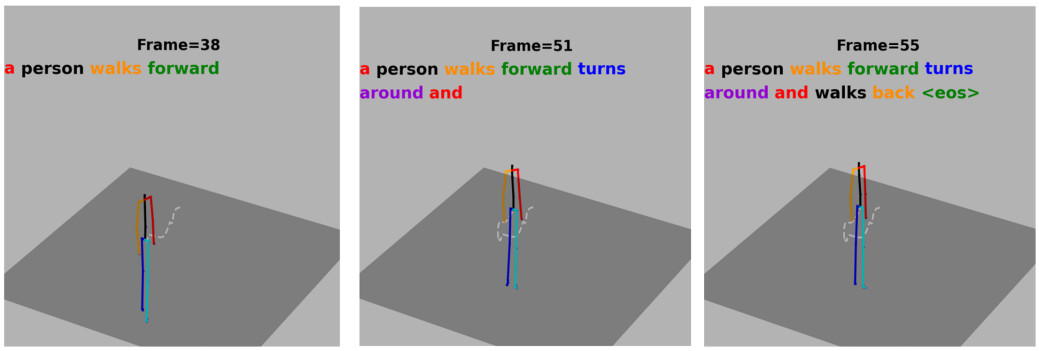
(a) Step to the left: ⟦15, 38⟧, step to the right: ⟦42, 53⟧.



(b) Pick at 32 and go up at 40.



(c) Walk forward ⟦13, 31⟧, bumps to the right: ⟧32, 43⟧ and recovers: ⟦44, 60⟧.



(d) Walk forward: ⟦15, 38⟧, turn at frame 40, walk backward: ⟦42, 53⟧.

FIGURE 4.34: Static visualization of progressive word generation along the motion.

4.7 Conclusion

In the conclusion of this chapter, we provided a comprehensive analysis of recent methods for motion encoding using discrete representations. We reviewed relevant techniques including transformer and GRU-based architectures for generating both language and motion. Subsequently, we presented our experiments, focusing on the mapping of motion to language, with the objective of achieving unsupervised segmentation and improved text generation. Our findings revealed qualitative limitations of recurrent neural network encoders and traditional attention mechanisms in terms of learning correct motion-language alignment. To overcome these limitations, we proposed a novel attention mechanism: *local recurrent attention* with frame-wise motion encoding that enables human motion segmentation, resulting in synchronized captioning as a byproduct. In the upcoming chapter, we will conduct a quantitative evaluation to assess the quality of the obtained segmentation and propose relevant metrics to analyze the attention visualization discussed earlier.

Chapter 5

Human Motion Segmentation and Dataset

5.1 Introduction

This chapter firstly focuses on the semantic segmentation of human motion in an unsupervised context and presents its theoretical formulation for quantitative evaluation as a continuation of Chapter 4. Secondly, it addresses the construction of a new dataset for the supervised learning of this segmentation. In general, semantic segmentation is a fundamental task in machine learning that involves dividing a given data into different parts or segments based on certain criteria. There are various types of segmentation techniques used in machine learning, including *image segmentation*, *semantic segmentation*, *video segmentation* or also *sign segmentation*. We are interested in *human motion segmentation* into atomic motions, and it's quantitative evaluation through *language semantic segmentation*. In this context, we highlight the motivations for human motion segmentation along with its associated tasks and applications (Section 5.2). Next, we establish formal definitions essential for the quantitative evaluation of segmentation and synchronization between motion and language (Section 5.3). Then, we detail our methods (Section 5.4) with relevant experiments (Section 5.5) and discuss the limitations of available datasets (Section 5.6). These analyses are based on our paper (Radouane et al., 2023a). Finally, since current available datasets do not enable supervised learning for segmentation, we construct the initial version of the **Euromov Motion Language Dataset** (EMLD), tailored for supervised motion-language alignment learning (Section 5.7).

5.2 Motivation via practical applications

In recent times, substantial advancements have been achieved in the domain of sign language research, focusing on various specific objectives, including *alignment* (Bull et al., 2021), *temporal localization* (Varol et al., 2021), and *sign spotting* (Momeni et al., 2020). In line with these efforts, a related approach in this field, proposed by (Varol et al., 2021), employs also attention scores to identify and segment signs in continuous video, as illustrated in Figure 5.1.

We discovered interesting visualizations of motion primitive time segments that emerged from another application aspect, namely *text-to-motion retrieval* (`TMR`). The authors (Petrovich et al., 2023) propose an architecture that performs motion retrieval based on natural language descriptions. For *Moment retrieval*, Figure 5.2

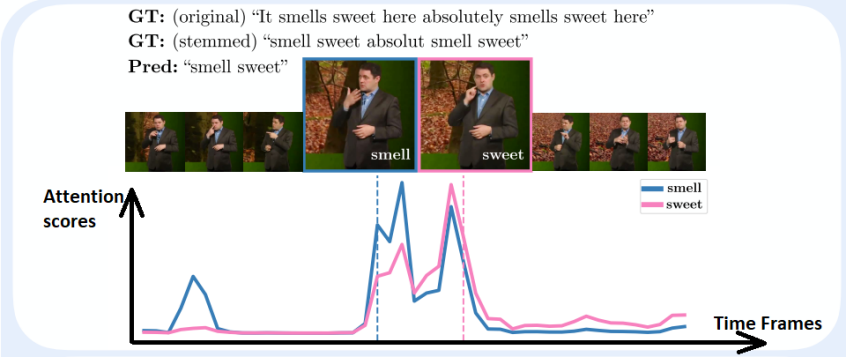


FIGURE 5.1: Attention scores evolution in frame time (Varol et al., 2021).

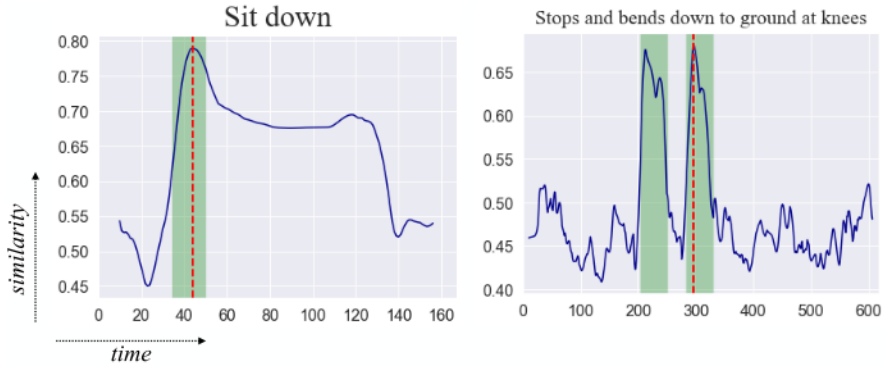


FIGURE 5.2: The ground-truth temporal span is denoted in green, the maximum similarity is marked with a dashed red line (Petrovich et al., 2023).

illustrates the similarity between temporally annotated BABEL text labels (Punnakkal et al., 2021) and motions using a sliding window approach, resulting in a 1D signal over time (depicted in blue). Remarkably, despite not being specifically trained for temporal localization, the TMR model demonstrates an inherent ability to localize relevant motions. While this *moment retrieval* process still requires a text language segment to be provided. In contrast to our **synchronized captioning** approach, which involves the automatic generation of text synchronized with the motion, enabling simultaneous ***language*** and ***motion segmentation*** that occur automatically. These two processes will be thoroughly examined and subjected to quantitative evaluation in the subsequent sections.

Sign language translation. The lack of communication between individuals with hearing or speaking impairments and the rest of the world can be frustrating and isolating for them. When spoken communication is impossible or undesirable,

sign language becomes essential, using bodily movements, particularly those of the hands and arms. Relying on manual sign language interpreters is not always ideal and can often intrude on the subject's right to privacy. To overcome these challenges, building an automated sign language translator can play a pivotal role. This process also involves the association between sign segment and their language segment.

Temporal action localization. Another domain of application that involves identifying and localizing actions in a video sequence with respect to their temporal boundaries. It is the process of detecting and localizing the start and end times of actions in a video, where each action is associated with a corresponding class label. The goal of temporal action localization is to accurately identify the time interval during which a particular action occurs in a video sequence. This task is useful in applications such as video surveillance, sports analysis, and human-computer interaction, where it is necessary to identify and track specific actions or events over time.

All these application areas could benefit from the use of the proposed *local recurrent attention* mechanism along with motion and language segmentation by making required adaptions.

5.3 Human motion semanticization

Definition. The human motion semanticization can be defined as the process of assigning meaningful labels or annotations to human motions in a video sequence. It involves analyzing the motion patterns of humans in the video and assigning semantic labels to the different types of motion or activities being performed. The goal of human motion semanticization is to identify and label human actions or activities in a way that is meaningful and interpretable to humans, such as *walking*, *running*, *jumping*, *sitting*, or *standing*. However, this process can go beyond simple action recognition. Indeed, the semanticization aims to provide a more fine-grained description level of human motion (e.g., *someone waving with right arm in high speed*).

Levels of semanticization. Fine-grained human motion segmentation involves measuring and analyzing human motions at a very detailed level, such as identifying the specific body parts or joints involved in each motion. Figure 5.3 provides

a representation of the motion segmentation point hierarchy in an exemplary motion recording of a person jumping, followed by walking three steps (Dreher et al., 2017). This level of segmentation provides a detailed understanding of human motion patterns and can be used for tasks such as biomechanical analysis, sports performance optimization/assessment and medical diagnosis. For example, fine-grained segmentation and analysis of gait patterns can be used to detect and diagnose musculoskeletal disorders or neurological conditions that affect walking patterns.

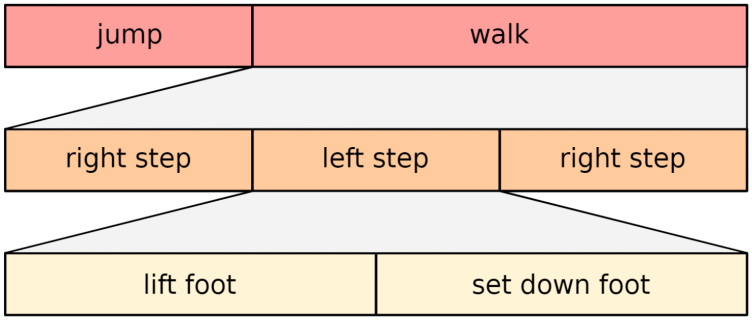


FIGURE 5.3: Motion segment with rough granularity (Red/top), medium granularity (Orange/middle) and fine granularity for a single step-segment (Dreher et al., 2017).

Human motion segmentation. The segmentation of motion was defined differently across authors, mainly using pose data, through the application of various techniques. In the study by (Lin and Kulic, 2012), authors aim to identify and segment movement repetitions using velocity features and stochastic modeling to select motion segment candidates. Then, a hidden Markov model (HMMs) is applied on these segment candidates to select the most relevant segment precisely. Similar techniques were applied in (Lin and Kulic, 2014). Others in (Kulić et al., 2009) learn to extract human motion primitives (*Lower arm raise*, *Bow Up/Down*,etc) based on HMMs from continuous time-series data converted to joint angles. The segmentation was treated differently in (Mei et al., 2021) as an unsupervised segmentation of human limb motion sequences, their proposed method used an autoregressive moving-average model (ARMA) to fit each limb bone angle, then the segment point is defined as the position where the current fitted ARMA models start to be not more convenient to describe the subsequence. In the context of segmenting motion into a sequence of actions, a technique based on the collaborative representation of 3D skeleton was proposed by (Li et al., 2018), also using the classical

Aligned Cluster Analysis (ACA) (Zhou et al., 2008) and Hierarchical one (HACA) (Zhou et al., 2013) algorithms. More recent works on skeleton-based action segmentation (Ma et al., 2021; Filtjens et al., 2022) relies on advanced techniques such as spatial temporal graph convolution networks. But to our knowledge, no previous work has been done with the aim of performing motion segmentation by synchronizing text generation to human skeleton motion, particularly using attention weights. In this work, we aim to infer this semantic segmentation of human motion automatically, using only Mocap data.

5.4 Methods

In this part, we present definitions and mathematical formulations (Section 5.4.1) for motion and language segmentation (Section 5.4.2). The methodology of the segmentation process and relevant metrics for the quantitative evaluation of segmentation performance are covered in Section 5.4.3.

5.4.1 Semantic motion segmentation

Before proceeding with the evaluation scores, it is essential to establish some key definitions necessary for quantitative motion segmentation. These definitions will serve as foundational concepts, providing a clear understanding of the subsequent discussions and analyses related to motion segmentation.

Word motion segment. A motion segment corresponding to the motion phase associated with a word w_i from a generated description, formally denoted S_t.

Primitive motion. A basic motion characterized by a unique combination of *action*, *direction*, and *speed*. A global change in the motion direction is considered as a *primitive transition motion*.

Language segment. A linguistic description segment which is the phrase associated with the word describing one specific action. As actions are almost always described by verbs, the linguistic description segments associated are often verbal phrases.

The *word motion segment* S_t is defined by masking the global sequence, as specified in Equation 5.1. Then, using the recurrence relation in Equation 4.23 and the formulation of S_t in Equation 5.1, we can derive intersection segment boundaries as concluded in Equation 5.2. The notation $[\![i, j[\![$ refers to the set of integers between i and j (i included, j excluded).

$$S_t = [\![p_t - D, p_t + D[\![\cap [\![0, T_x[\![= [\![\max(0, p_t - D), \min(T_x, p_t + D)[\![\tag{5.1}$$

$$S_t \cap S_{t-1} = [\![p_{t-1} - D + \epsilon + (T_x - 1 - p_{t-1} - \epsilon).\sigma(v_p^t \cdot W_p h_{t-1}), \min(T_x, p_{t-1} + D)[\![\tag{5.2}$$

We define $\boldsymbol{\epsilon} \in \mathbb{N}$ as a parameter that can be used to force a minimum shifting between two successive positions and also reduce overlapping between successive local attention windows, by setting $\boldsymbol{\epsilon = (1 - \alpha).2D}$, where $\alpha \in [0, 1]$ is an explicit parameter to control overlapping proportion between successive motion segments $S_t \cap S_{t-1}$.

Using Equation 5.2 we have $\alpha = 0 \Rightarrow S_t \cap S_{t-1} = \emptyset$. This can be convenient to ensure proper word ordering and fast training in motions with clearly separate motion segments. Of course, in more complex cases such as simultaneous events, this mechanism is likely insufficient on its own. In addition, as a result of the recurrence effect, $\boldsymbol{p_t}$ is mechanically incremented at least by $\boldsymbol{\epsilon}$, meaning that $\boldsymbol{p_t \geq \epsilon \times t}$. This can lead the network to miss critical time intervals and to quickly go towards the final positions, particularly in very short and quick motions. To allow generating words describing simultaneous actions at the same time, we set $\epsilon = 0$.

Hypothesis. Also we note that Equation 4.23 implicitly assumes that the motion and language description are *monotonically* related, meaning that the description of successive phases of a movement, will be in chronological order of the performance of the motion.

5.4.2 Motion and Language segmentation

For qualitative evaluation of primitive segmentation, we label some representative samples in the dataset for every action (*Kicking*, *Walking*, *Stomping*,etc). For evaluation, we needed an adapted alignment score, this process required to introduce precise definition of a motion segment. As detailed in (Lin et al., 2016), the definition is specific to the task. In our work, we want to evaluate synchronization between description words and motion primitives. To this end, we propose three adapted method to calculate a segmentation scores: (i) Intersection over Union, (ii) Intersection over Prediction and (iii) Alignment position.

The primary objective of this study is to quantitatively verify the accuracy of synchronization between correctly generated descriptions and motions. We exclude motions with completely incorrect predicted descriptions, as evaluating synchronicity for such cases would be meaningless. The assessment of synchronicity occurs after ensuring correct semantic predictions. The two processes given the *motion segment* and corresponding *language segment* will be referred respectively as motion and language segmentation. The details of these segmentations is formulated as follows:

•**Language segmentation.** To perform language segmentation, we first retrieve the words corresponding to the actions from the annotations and calculate their indexes k_m in the prediction. Let w_m be the *action word annotation* of a motion segment, we search for the index k_m of the word w_m in the predicted words then for the next word annotation w_{m+1} and so forth until the final primitive segment is reached, we denote the index of the end token as k_e. As result of this process, formally, a language segment L_m is given by $\boldsymbol{L_m = \{w_r, r \in [\![k_m, k_{m+1} - 1]\!]\}}$.

•**Motion segmentation.** We define P_m as the segment of a primitive motion. The calculation of P_m is given by the concatenation of words motion segments according to index in the set $[\![k_m, k_{m+1} - 1]\!]$ as given by the Equation 5.3. We recall that the word motion segment S_t was defined in Equation 5.1.

$$P_m = \bigcup_{i=k_m}^{k_{m+1}-1} S_i \tag{5.3}$$

5.4.3 Segmentation process

The two processes of language and motion segmentation are illustrated in Figure 5.4. Given a motion x and the predicted description y_x as "*a person walks forward then turn around and walks backs* <eos>". Let $\{S_i, i \in [\![0, 10]\!]\}$ be the set of motion segments associated to the eleven words forming the description y_x. Using the definitions above, we have three mappings :
$P_0 \rightarrow S_{k_0}, ..., S_{k_{0-1}} \rightarrow$ *walks forward then* ; $P_1 \rightarrow$ *turns around and* ; $P_2 \rightarrow$ *walks back* <eos>.
The indices k_m are found by searching the ground truth word describing the action in the predicted sentence, and then using the corresponding segments for the group of words describing the motion primitive (cf. Figure 5.4). For this specific example, $(k_0 = 2, k_1 = 5, k_2 = 8)$. Consequently, the motion sequence x is a sequential composition of three primitive motions P_0, P_1 and P_2. Ideally $G_i = P_i$,

but the subjectivity of human annotation introduces an inevitable variability in the start/end timestamps. Note that we always include the end token <eos> in the final language segment, because its motion segment is correlated to motion ending time.

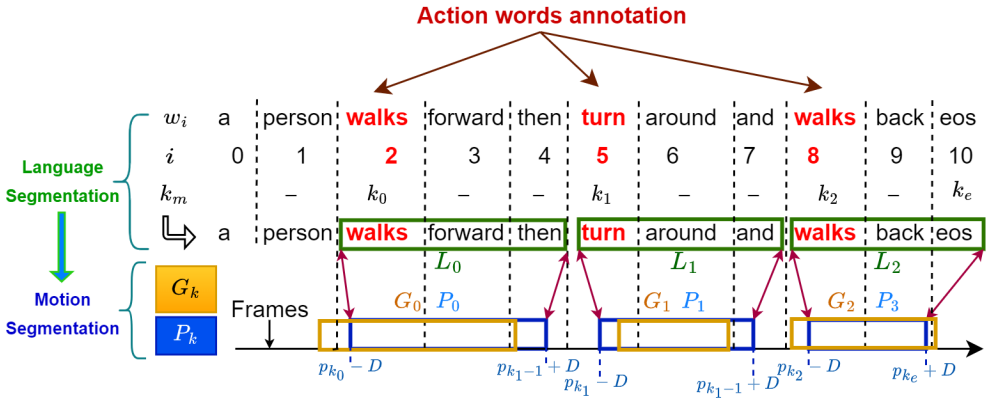


FIGURE 5.4: The language and segmentation process applied on an example of predicted words w_i with given action words and frame annotations G_k. The inferred motion segment is P_k and the language segment is L_k. The index k_e always refers to the end token (Radouane et al., 2023a).

In all following equations, we note S^e_{seg} the segmentation score of a sample e, and $\mid E \mid$ the cardinality of the set E. We note by $\mathbb{1}_A$ the indicator function ($\mathbb{1}_A = 1$ if condition A is true and 0 otherwise).

For a qualitative evaluation of motion and language synchronization, we propose to calculate the segmentation score using three different methods:

i) Intersection over Union (IoU). Measure the proportion of intersection between the primitive segment and a reference segment. Based on the *IoU* we calculate the segmentation score through Equation 5.4.

$$S^e_{seg} = \frac{1}{n_s} \sum_{m=0}^{n_s-1} \mathbb{1}_{(\mathrm{IoU_k} \geq \theta)}, \quad \underset{m \in [0, n_s[}{\mathrm{IoU_k}} = \frac{\mid P_m \bigcap G_m \mid}{\mid P_m \bigcup G_m \mid} \tag{5.4}$$

We note by n_s the number of motion segments, $m \in [\![0, n_s[\![$ is the motion primitive index, θ is the selected threshold. The reference motion segment for an *action word* w_k is the ground truth segment G_k, representing also the k^{th} primitive for a given motion sample. The measure *IoU* is sensitive to the absolute quality of the ground truth, which isn't always desirable due to annotation variability, which led us to also propose the following measures:

ii) Intersection over Prediction (IoP). We introduce the *IoP* as a measure for the inclusion proportion of P_k in G_k, so that we have $\boldsymbol{IoP_k = 1 \Leftrightarrow P_k \subset G_k}$, meaning that the segmentation is counted as completely correct when the prediction segment interval is in the range of the ground truth interval. This measure correlates well with the perceived visual quality of transcription generation and reduce the impact of annotation subjectivity (start/end primitive annotation uncertainty). In this case, S^e_{seg} is calculated as in equation (5.5).

$$S^e_{seg} = \frac{1}{n_s}\sum_{k=0}^{n_s-1} \mathbb{1}_{(\mathrm{IoP_k}\geq\theta)}, \underset{k\in[0,n_s[}{\mathrm{IoP_k}} = \frac{\mid P_k \bigcap G_k \mid}{\mid P_k \mid} \tag{5.5}$$

iii) Element of. Evaluate if the alignment position p_t is an element of the annotated segment interval. The score S^e_{seg} is directly defined by Equation 5.6. This is the least strict method, counting any amount of overlap as full correspondence.

$$S^e_{seg} = \frac{1}{n_s}\sum_{m=0}^{n_s-1}\sum_{i=k_m}^{k_{m+1}-1} \mathbb{1}_{(p_i\in G_i)} \tag{5.6}$$

5.5 Experiments

This section presents and discusses results from quantitative evaluation and contextualises the results through qualitative visual analysis.

5.5.1 Quantitative evaluation

We have manually annotated 68 samples from KIT-MLD with *motion segments* and their *action words*. For a compatible comparison between models, we select the samples with a semantically correct prediction in common across the models. This selection results in $N = 35$ common correct predicted representative samples for the calculation of scores. Each score is the mean value calculated on all samples $\frac{1}{N}\sum_{e=0}^{N-1} S^e_{seg}$.

The segmentation scores as defined previously are computed with a threshold, anything below is zero, anything above is one. We can also compute a continuous variant, where we sum the actual scores without threshold: $S^e_{seg} = \frac{1}{n_s}\sum_{k=0}^{n_s-1} m_k$. Figure 5.5 illustrates the threshold (solid lines) score for thresholds from 0 to 1 as well as the continuous scores (dashed lines). *Element of* is not included as it isn't compatible with such a visualization. As expected, the *IoU* is too strict, and the score is almost always zero even when the synchronization is perceived as

correct through visual inspection. We conclude that this measure is not adequate to give a fair assessment of how humans perceive synchronization. We also observe that the continuous scores remove the parameter that is the threshold and give an informative assessment of synchronization quality.

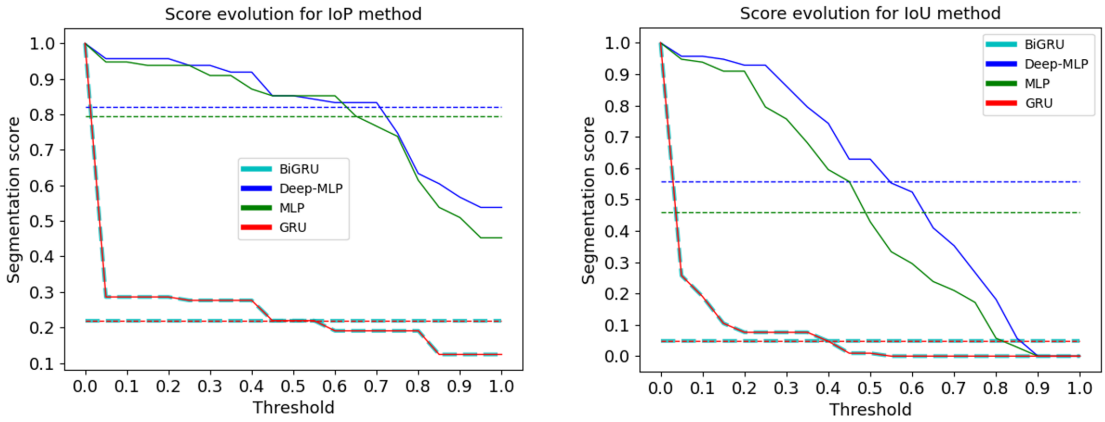


FIGURE 5.5: Comparison of IoP and IoU scoring methods. The legend specifies the encoder types, and the dashed lines represent the average segmentation scores (Radouane et al., 2023a).

In Table 5.1, we report all continuous scores, including *Element of* for a final comparison. We see that the segmentation performance of the GRU was low in comparison with the MLP encoder. The BiGRU has approximately the same performance as GRU. The two have a problem of a late detection of motion. The deeper MLP seems slightly better in segmentation than the shallow MLP regarding the continuous values *IoU* and *IoP*. Which is confirmed by the respective curves (cf. Figure 5.5). Noting also that the curves for GRU and BiGRU decrease exponentially w.r.t the threshold, confirming the low ability of recurrent encoders to perform semantic segmentation. Regarding the metric based on *Element of*, the MLP encoder is slightly better than Deep-MLP. This means that the MLP encoder localizes the main time when the action happens better, while Deep-MLP localizes the start/end of the complete action better.

5.5.2 Word-motion attention based mapping

In this part, we will discuss the perceived synchronization of generated words with motion through a qualitative error analysis. We illustrate this using the evolution of human skeleton motion and attention maps. Figure 5.6 shows a correct segmentation in the presence of multiple primitive motions (U-turn). This motion is

System	IoU	IoP	Element of
MLP-GRU	45.97	79.47	**89.05**
MLP-GRU (Deep)	**55.92**	**82.06**	88.09
GRU-GRU	5.01	21.90	6.67
BiGRU-GRU	5.01	21.90	6.67

TABLE 5.1: Continuous segmentation scores for four encoder type [MLP, Deep-MLP, GRU, BiGRU]. All with Cartesian coordinates and local recurrent attention [D=5,Mask=True] (Radouane et al., 2023a).

composed of three primitives "walks forward", "turns around" then "walks back", the primitive "turns around" represents the transition action.

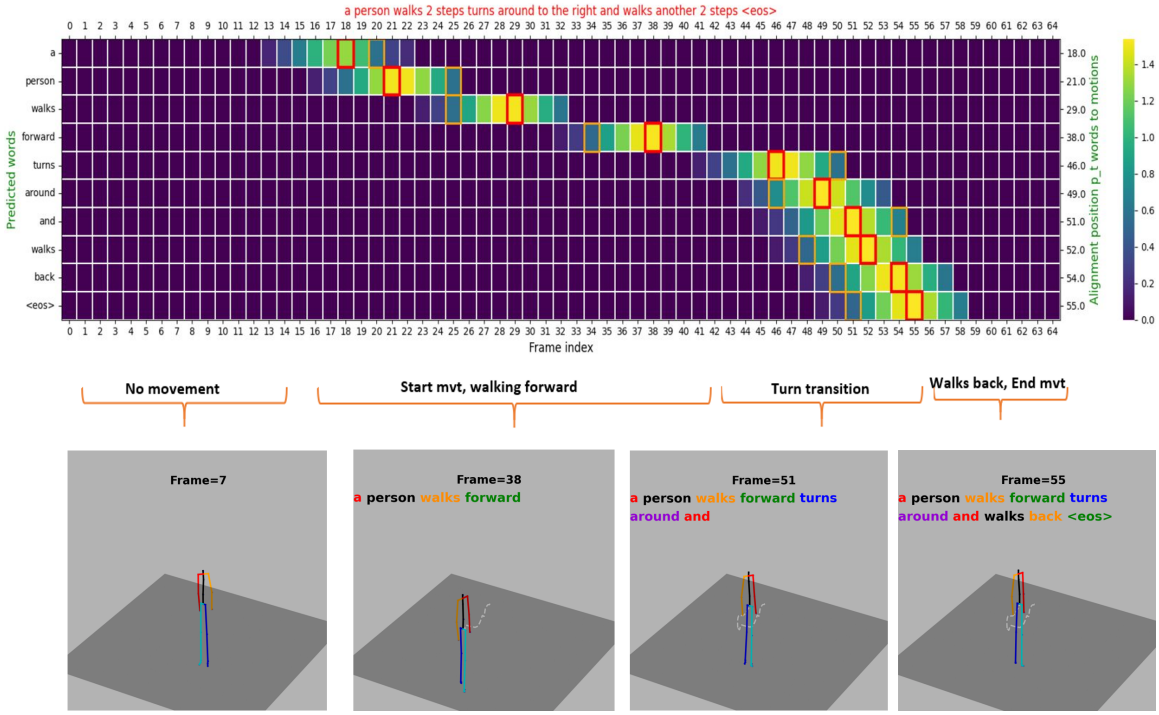


FIGURE 5.6: MLP-GRU [Local.rec.att Mask True $D = 5$]: walk forward in the range $[\![15, 38]\!]$, turn at frame 40 , walk backward in $[\![42, 53]\!]$

.

Figure 5.7 shows a stomping action, where the word *"stomping"* is associated to the frame interval $[\![22, 30]\!]$, which is the exact moment of execution of the stomping action, followed by an accurate recognition of the body part executing the motion *"left foot"*. Note that the reference and the generated descriptions convey the same meaning but use different words. The BLEU score would be incorrectly low in this case. Also note that the attention weight start to be higher when the human

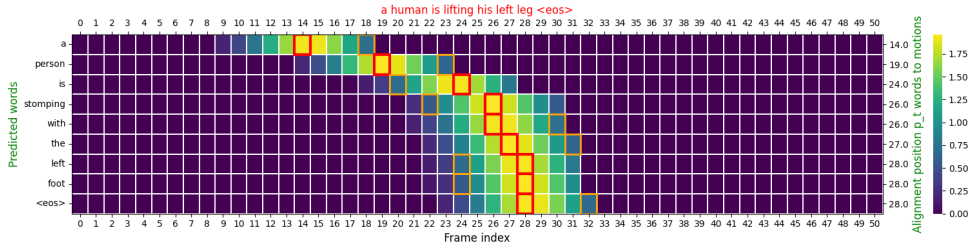


FIGURE 5.7: Truncated Gaussian MLP-GRU with $D = 5$, Stomping action at $[\![22, 30]\!]$.

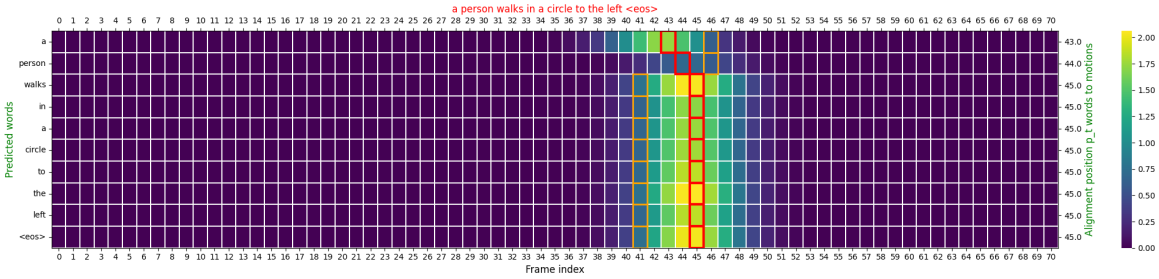


FIGURE 5.8: Mask false MLP-GRU [Turn left].

skeleton starts moving around the frames in the range $[\![13, 16]\!]$. Figure 5.8 gives an example of late detection and of a fixed alignment position p_t. The description was generated all at once around frame 45, too late compared to when the action is executed. Additionally, the words are not semantically coherent with successive actions in the motion evolution, with this model the alignment information is lost entirely.

5.5.3 Synchronization between motion and words

We propose a better visualization method for the synchronization by a parameterized skeleton transparency using attention weights. A sequence of motion frame $j \in [\![0, T_x - 1]\!]$ is generated with this transparency values for each predicted word w_i, where the transparency V_{ij} for a frame j given a word w_i is computed from the attention weights using Equation 5.7. The motion sequence is visualized for action words, as illustrated in Figure 5.9.

$$V_{ij} = \frac{1}{\max_{j<T_x}\{G_{ij}\}} \times G_{ij} \qquad G_{ij} = \frac{\exp\left(F \times \alpha_{ij}\right)}{\sum_{j=0}^{T_x-1} \exp\left(F \times \alpha_{ij}\right)} \tag{5.7}$$

The factor F is set to 100 for better visualization and controls the number of frames present per image. Note that none of these operations change the position

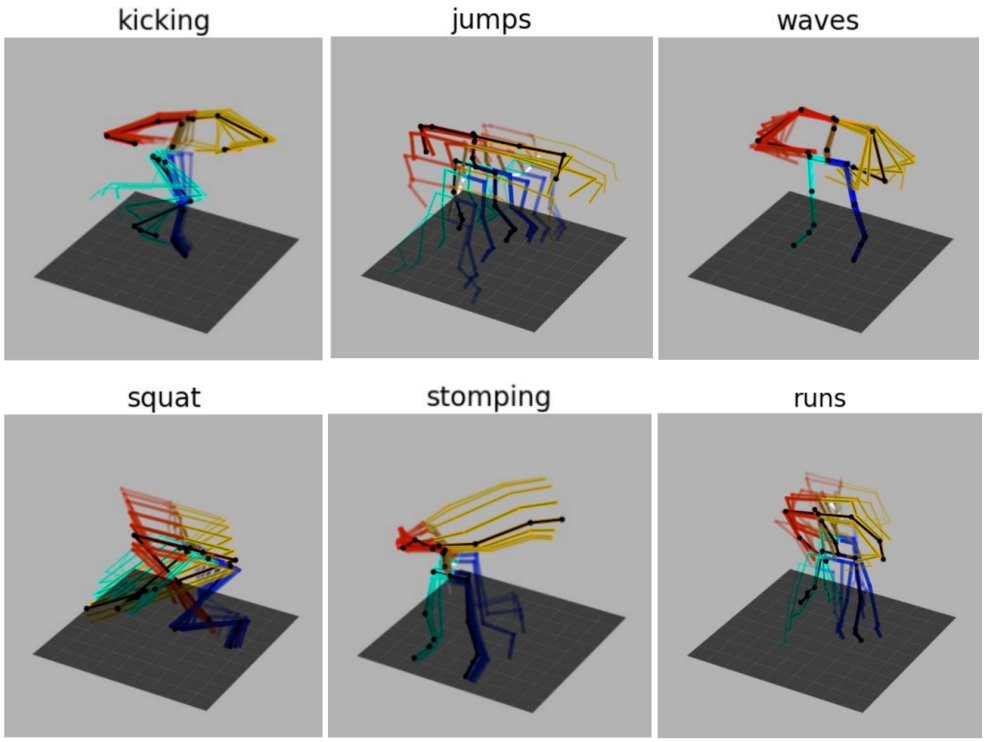


FIGURE 5.9: MLP-GRU [Local.rec.att Mask True $D = 5$]: action words and their high attention motion frames. Given a word w_i, the transparency of each frame j is set to value V_{ij} as described in Equation 5.7 .

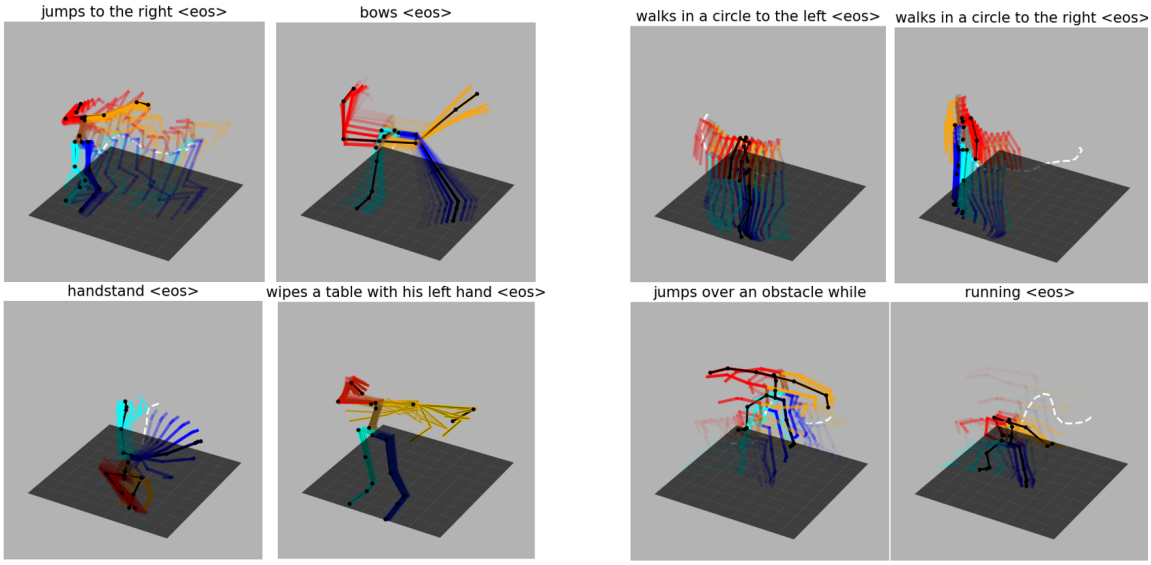
of alignment and maximum of attention, it only helps to have a better visualization of synchronization as a frozen sequence inside the interval $[\![p_t - D, p_t + D[\![$ (mask=True). 3D animation can be found in this repository[1] as a point of comparison to see the relevance of these static visualizations.

The visualization of sequence skeleton may be only relevant for words directly describing the motion. We have found that for the repeated words *"a person"* it is usually aligned with the start of movement, providing an initial position to predict the coming action, but are of no interest when visualizing the synchronization of language and motion segments.

5.5.4 Mapping language segments to motion primitives

As discussed before, it is more pertinent to associate a set of frames not only with a single action word but with the phrase describing the motion primitive with higher granularity, which differs from mere action recognition. Figure 5.10 shows

[1] https://github.com/rd20karim/M2T-Segmentation



(a) Single action. (b) Multiple actions.

FIGURE 5.10: Mapping $P_m \rightarrow L_m$: multiple actions vs. single action.

the mapping of the human pose sequence for the frames in the set P_m described by the language segment L_m. The transparency is calculated as before in Equation 5.7, but the coefficients α_{ij} are replaced by the coefficients γ_{ij} defined in Equation 5.8, computed for each language segment L_m.

$$\gamma_{k_m j} = \frac{1}{k_{m+1} - k_m} \sum_{i=k_m}^{k_{m+1}-1} \alpha_{ij} \qquad \forall m \in [\![0, n_s - 1]\!] \tag{5.8}$$

Specially when $m = n_s - 1$ we should have $k_{m+1} - 1 = k_e$. As described before, we always include the token `<eos>` in the final language segment, so by definition $\boldsymbol{k_{n_s} = k_e + 1}$.

5.6 Limitations of the recurrent attention and dataset

Counting motion. The KIT-ML dataset doesn't contain sufficient examples with a variable number of repetitions to learn the ability of counting repetitions (number of steps, waving hands, etc). Indeed, in the majority of cases, it includes only a few examples given fixed counts *(walking 4 steps)*. Instead, the architecture needs to see examples with a variable number of repetitions in the training data *(e.g., walks 5,4,2 steps forward)* to allow for a better generalization. However, this is not the only requirement, a strict limitation of visibility of the input in the encoder will

not allow the model to learn how to count repetitive motions, as a larger window width would be required. A partial solution can be to apply multiple Gaussian windows, as this could allow counting.

Dynamic motion measure. The information about the speed of the movements was seldom present in the dataset and consequently, the models are rarely able to generate descriptions regarding the speed of movements.

Body part identification. We have seen in some samples a wrong detection of the body part executing the motion (*e.g.*, *left leg* instead of *right leg*), which is the result of i) few descriptions rich enough to make the distinction in dataset, ii) the limitations of the extracted features that do not take skeleton geometry into account. Applying architectures allowing spatial feature extraction can be more pertinent to add this capability.

5.7 Building dataset for motion-language alignment

A system devised to learn associations between direct parameters and the description, is mainly able to capture perceptual or invariant properties of the physical movement, particularly the phases of the movement, its compositional structure (complex movements composed of motion primitives or invariant structures), qualifiers that specify the modes of movements (e.g., walking, running), the manner in which the movement is executed (e.g., quickly, stealthily, carefully), and the identification of discrete actions. The next step in the development of motion language applications involves adding more abstract layers of interpretation above these low-level associations pertaining to higher-level goals, grounded in a more profound understanding of the world and the surrounding context. In the following, we present the road map toward a more accurate synchronous captioning.

5.7.1 Road to supervised semantic segmentation

So far, we have presented experiments towards unsupervised learning of semantic motion segmentation. As results, we have designed a specific attention mechanism: *local recurrent attention* to infer motion segmentation and providing a quantitative evaluation by labeling a subset of Test set. The need for unsupervised learning stems from the lack of time annotation of atomic actions composing the global motion and their partial descriptions in KIT-MLD (Plappert et al., 2016) and

HumanML3D (Guo et al., 2022a). Furthermore, these datasets do not contain sufficient compositional motion. Considering all these constraints, in addition to the inherent limitations of unsupervised techniques, we have chosen to create a human motion language dataset with time annotations as a solution. In this section, we describe the construction of the initial portion of this dataset, including temporal and linguistic annotations for actions. Our dataset aims to open the path for supervised learning of human-motion alignment in the context of skeleton-based captioning.

Given the start and end time of each primitive motion and associated partial description, the previous proposed local recurrent attention could be supervised to be more accurate. Furthermore, we could advance the architecture design using dataset with more compositional motion and their annotations.

5.7.2 Euromov Motion Language Dataset-EMLD

In this first phase of EMLD acquisition, we focused on capturing a wide range of atomic and compositional actions. Each single motion action is annotated with text given a fine-grained description such as: speed of action, body part involved, trajectory and others.

Atomic motions. These primitive motions were recorded and annotated to provide a diverse and comprehensive dataset for our research. Table 5.2 illustrates the selected actions in the first phase and the differentiation within each action by the manner of execution, *specificity* and the subject *instruction*.

Compositional motions. Represented by two and three successive actions, sampled from the list of atomic actions with different specifications, to cover a representative set of possible combinations. Thus, our algorithm generate action following the sampling rules:

Sampling rules. Taking on all possible two and three sequence of actions with all specifications will result in a very important number of combinations. Consequently, the sampling process is constructed by the following rules to have a representative set among subject: i) The same action with different modifiers cannot be part of the same combination, ii) Two different actions involving the same body parts cannot be included in the same combination, iii) An atomic action can appear

Atomic motion	Specificity	Instruction
Turn	left | right	A simple turn to the left or right.
Walk	random	Basic walking without additional specifications.
Walk	quickly | slowly | steps	Walking at varying speeds for a random number of steps $N \sim U(2,5)$, uniformly sampled).
Kick	right | left	Executing a kick using the specified leg.
Punch	right | left	Delivering a punch with the specified arm.
Throw	right | left	Throwing an object using the specified hand.
Pickup	right | left	Bending down and picking up an object with the specified hand.
Clap		Performing a clapping motion.
Bend		Bending the upper body forward.
Squat		Executing a squatting motion.
Jump		A basic jumping action.
Jog		Performing a jogging motion.
U-turn		A complete turn or rotation around.
Wave	right | left | both	Waving using the specified hand(s).

TABLE 5.2: Details about defined atomic motions.

a maximum of 4 times in the set of 2-action combinations and 5 times in the set of 3-action sequences. Each experimental session consists of a total of 100 sequences, which include 22 atomic actions with modifier variations, 39 two-action sequences, 39 three-action sequences. Each configuration is deterministic and identified by a unique number K.

In this first phase, 9 **subjects** participated in the acquisition. Each participant performs 2 sessions of 100 sequences, with two unique configuration numbers assigned to each session. Therefore, no two participants perform identically the same samples. All these parameters are defined in a configuration files.

Subject instructions. Participants are provided with clear instructions through an automated program that outlines the sequence of actions they are required to perform (audio voice). Following this, participants receive an order to begin the execution at their convenience. Meanwhile, we record the transition times between consecutive actions.

5.8 Experimental setting and acquisition protocol

The acquisition has taken place in the AIHM motion capture experimental platform at *IMT Mines Alès* which has a Qualisys MIQUS motion capture system with 13 cameras, placed around an 18 square meters open area (cf. Figure 5.11).

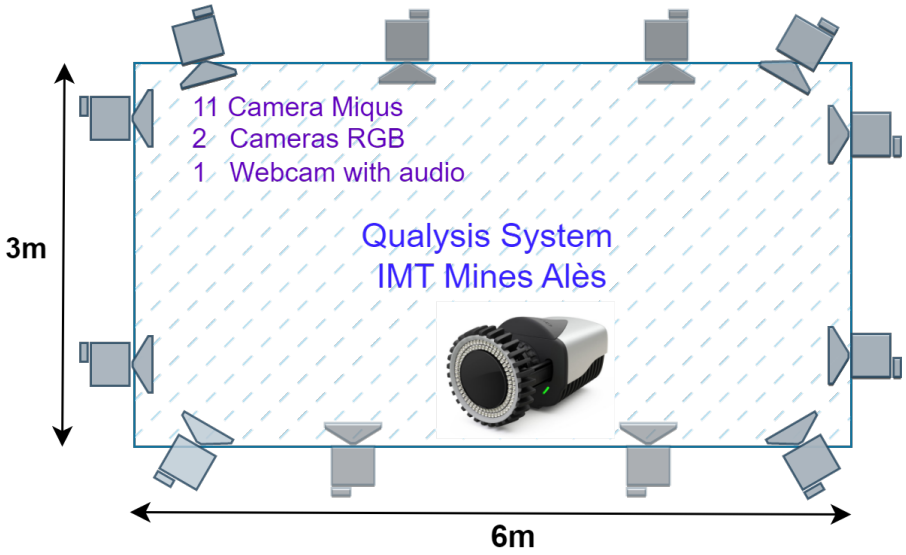


FIGURE 5.11: The Mocap acquisition area 6×3.

Additional synchronous acquisition modalities can be added through the MIQUS sync unit using TTL signals or through the QTM RT protocol. For EMLD, motion is captured with a Qualisys System using a sports marker-set, featuring 43 markers positioned as illustrated in Figure 5.13. The cameras are placed around a rectangular area, to obtain an effective and accurate capture volume of $6m \times 3m \times 2m$, as illustrated by Figure 5.12.

5.8.1 GUI Dashboard

The acquisition is driven by a python program that generates the experimental plan and executes it by giving synchronous instruction to subjects. An experimental dashboard is provided to experimenters. The program controls the Qualisys Track Manager acquisition software through Version 2.3 of the Qualisys RT protocol to start/end/save recording, but also to send time-indexed events corresponding to the successive phases of the experimental protocol in order to annotate said acquisition. The program offers a simple graphical user interface (`PyQT`) in the form of a dashboard that allows the experimenter to drive the acquisition. The

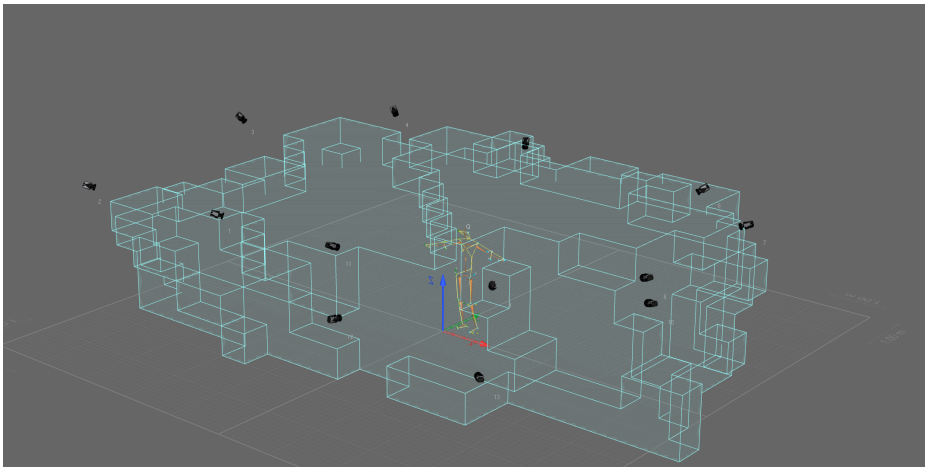


FIGURE 5.12: The Mocap acquisition volume $6 \times 3 \times 2$.

driver program is made openly available on GitHub [2] with detailed reproduction instructions.

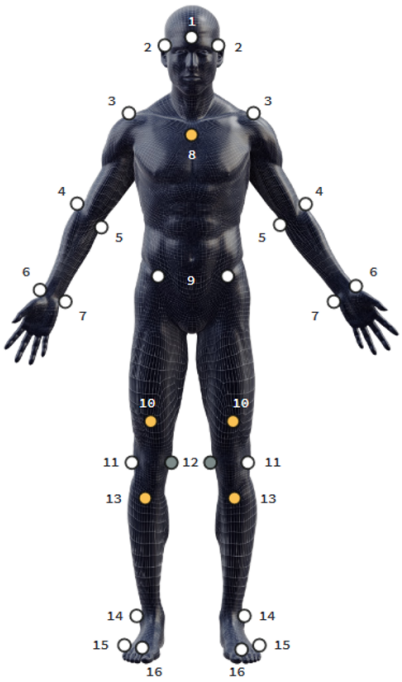
Generation Process. In the context of action description generation, we employ a systematic approach. Each atomic action is associated with a linguistic description template, featuring placeholders akin to Python format strings. For instance, a template might be structured as "walk{mod} for {# steps} steps" or simply "right kick."

Subsequently, we define modifiers that correspond to specific classes, allowing us to add variations to the actions. For example, the instruction *"Kick{mod} with the {body_parts}"* specifies that we are only interested in the *{left_leg, right_leg}* for this action. Another example is the instruction: *"Walk{mod} {steps} steps {speed}"* with available speed options of *{quickly, slowly}*.

During the generation process, we generate all modified variants by iterating over the predefined set of values for these modifier classes, replacing the placeholders with the corresponding modifier expressions using regular expressions. For combination descriptions, we define a combination type and a specific pattern with placeholders for the generated text related to the involved atomic actions. For example, `"First, {$A_1$}, then {$A_2$}, then {$A_3$}"`, where A_j stand for a given Action j.

This enables us to generate descriptions for various action combinations, while keeping in mind the potential for recursive generation up to an arbitrary depth, although we primarily focus on simpler cases.

[2] https://github.com/EuromovDHM-SemTaxM/EMBLD-acquisition



1	HeadFront
2	HeadL, HeadR
3	LShoulderTop, RShoulderTop
4	LElbowOut, RElbowOut
5	LElbowIn, RElbowIn
6	LWristIn, RWristIn
7	LWristOut, RWristOut
8	Chest
9	WaistLFront, WaistRFront
10	LThighFrontLow, RThighFrontLow
11	LKneeOut, RKneeOut
12	LKneeIn, RKneeIn STATIC
13	LShinFrontHigh, RShinFrontHigh
14	LAnkleOut, RAnkleOut
15	LForefoot5, RForefoot5
16	LForefoot2, RForefoot2
17	SpineThoracic2
18	LArm, RArm
19	LHand2, RHand2
20	SpineThoracic12
21	WaistBack
22	WaistL, WaistR
23	LHeelBack, RHeelBack

FIGURE 5.13: The sport Marker Set used in our setting.

5.8.2 Standard conversion: Keypoints and sub-captions

Sport marker-set to SMPL keypoints. As described earlier in Chapter 2, each MoCap system has its own definition of marker set, including the number of joint and location definition, making it challenging. Thus, a standardization step is required, which poses its own challenges. Up to now, few works have attempted to address solving optical marker-Based MoCap automatically such as SOMA (Ghorbani and Black, 2021). Experimenting with this framework on real-data, we noticed a limitation in SMPL mesh reconstruction, such as body deformation and unnatural human motions. This is likely due to the inevitable presence of occlusions and missing markers on some frames or even ranges of frames. Amelioration could be further performed by retraining this framework to handle these issues.

From sub-caption to full natural caption. This process consists in the conversion of partial descriptions or sub-captions into full descriptive sentences. This conversion could be performed using state-of-the-arts Large-Language-Models (LLMs), or simply Generative Pretrained Transformers (GPT) (Brown et al., 2020) that are readily available and widely used in various applications. For instance, the sequence of instructions: *"walks slowly"*, *"pick something with right hand"*, *"throw an*

object", will be transformed into a coherent sentence (`GPT output:` *"the person walks slowly, picks something with their right hand, and then throws the object"*).

5.9 Conclusion

We have proposed a supervised motion-to-language translation system that incorporates unsupervised alignment techniques based on attention weights that allow for synchronous text generation. Such a system, beyond the generation of motion descriptions in natural language, can be potentially adapted to other tasks where synchronous generation is important: subtitles of movies, sign language transcription, skeleton-based action segmentation, etc. For instance, for action segmentation, the decoder will generate a sequence of action labels instead of words, and the recurrent attention formulation will synchronize the action segments with minor adaptations. Our method can also be an alternative to classical clustering algorithms (Zhou et al., 2008, 2013). In general, there are multiple strategies that can be employed to drive the adaptation to adjacent tasks: adjusting the value of the D parameter, tuning ϵ to control the amount of overlap in the alignments. Regarding the tasks of motion-to-language translation and alignment, there are additional improvements that may be investigated to further improve alignment performance. Notably, by the use of the proposed EMLD dataset for a supervised approach towards better motion-language alignment.

Chapter 6

Interpretable Captioning With Guided Attention

6.1 Introduction

In this chapter, we investigate an alternative approach to motion semanticization, not solely relying on the temporal dimension as explored in the previous Chapters 4 and 5, but also incorporating spatial information. Our primary objective is to enhance the model interpretability and quality of textual descriptions of human motions. We aim to bridge the gap between attention mechanisms and motion captioning, leading to improved performance and a deeper understanding of how pose-based motion information and language generation interact in motion semantic analysis. First, we briefly review relevant captioning methods based on images and videos (Section 6.2). Drawing inspiration from these vision-based captioning approaches, we develop our contributed formulations for spatio-temporal and adaptive attention for motion captioning. Moreover, we introduce a novel supervision strategy for both attention modes. Globally, our architecture includes a part-based motion encoding, with attention guidance toward an interpretable and more accurate caption generation (Section 6.3). In the experimental phase (Section 6.4), we conduct a detailed analysis and ablation studies to validate the effectiveness of our proposed architecture. Furthermore, our architecture is interpretable by design, offering a model with a transparent reasoning process, in contrast to previous black box architectures. Thus, we propose effective tools for global evaluation of interpretability across the Test set as supported by qualitative illustrations (Section 6.5) and the potential application of proposed methodologies in other tasks (Section 6.6). We report state-of-the-art results on two challenging benchmarks, Human-ML3D and KIT-MLD (augmented version). The chapter's results and analyses are based on our paper (Radouane et al., 2023b)[1].

6.2 Vision-based captioning

Image and Video captioning has a wide range of applications, encompassing tasks such as video retrieval (Wu et al., 2023), and providing assistance to visually impaired individuals (Tiwary and Mahapatra, 2023). The challenges associated with this task are significant, as it requires the integration of visual and language comprehension to generate meaningful captions for images and videos. As a result, it has attracted significant attention in the fields of computer vision and natural language processing.

[1]Code: https://github.com/rd20karim/M2T-Interpretable

In this part, we will delve into a thorough review of the most relevant research that served as foundation and inspiration for our work. We aim to explore key studies, methodologies, and findings from various domains that have contributed to shaping our research objectives and approach for motion captioning.

6.2.1 Image captioning

In this context, the models are designed to learn the mapping from an input image to a sequence of words that effectively describe its contents. The primary objective is to accurately represent the objects, scenes, actions, and relationships present in the image. Typically, these architectures consist of two essential components: the Encoder, responsible for transforming raw input data into meaningful and informative representations, and the Decoder, responsible for generating textual descriptions based on the encoded information.

Encoder. A Convolutional Neural Network (CNN) encoder is commonly utilized for image encoding. The final nature of the encoded image depends on the method used. Regarding a recent survey proposed by (Stefanini et al., 2023) on multiple approaches, the distinction can be made based on the use or not of attention mechanisms. Figure 6.1 illustrates various methods for learning image representation. Approaches with *No attention* (cf. Figure 6.1.(a)), use a global feature vector extracted using a CNN architecture. Others apply an *Attention over grid*, where the features describing each region in the image are assigned different weights (cf. Figure 6.1.(b)). These features are usually obtained by the lower convolutional layer of the chosen pre-trained CNN. In (Fu et al., 2017), the system aligns the process of generating words with the visual perception experience by shifting attention among different visual regions. Where some methods rely on *attention over visual regions*, employing a decoder to extract relevant features for each image region. The attention model then utilizes these feature regions to compute the final representation, which the language model uses for generating a word at each time step (cf. Figure 6.1.(c)).

Attention model. Diverse methods were utilized to compute attention scores. Generally, the attention score is computed based on the similarity between the hidden state at time i and a vector representing a location j in the image. This similarity function can be a simple dot product with learnable weights, as suggested

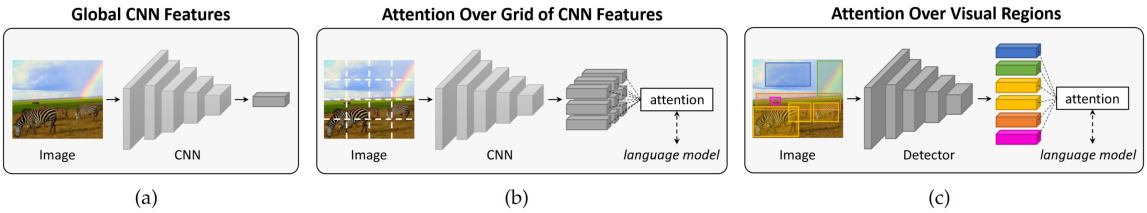


FIGURE 6.1: Different methods for image feature extraction (Stefanini et al., 2023).

in (Luong et al., 2015). Alternatively, an additive formulation, as exemplified in (Bahdanau et al., 2015), can be employed. More recent attention models are built upon the scaled-dot product of learned queries and keys (Vaswani et al., 2017).

Decoder. Decoder architectures considered were often recurrent, such as the GRUs or LSTMs. In the last years, Transformer (Vaswani et al., 2017) based decoders start to be extensively used in captioning tasks.

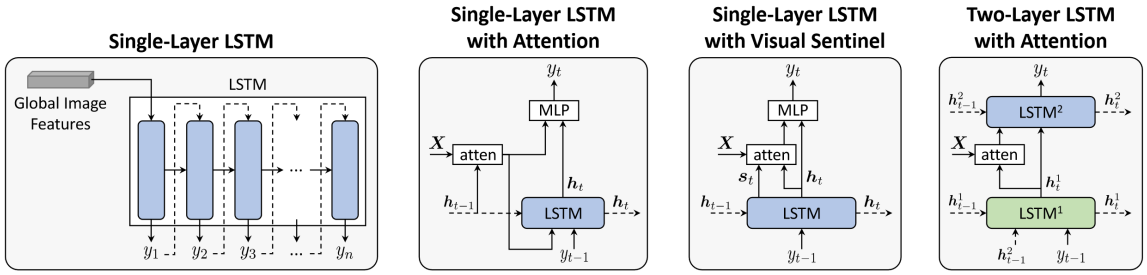


FIGURE 6.2: LSTM-based decoder. (a) No attention, (b)-(c)-(d) different methods for attention score computation (Stefanini et al., 2023).

Attention model & Decoder interaction. The Decoder learns how to *dynamically* attend to different parts of the image while generating the caption. The attention mechanism guides the language model to focus on the most relevant parts of the image while generating each word of the caption. The interaction between language and encoded input information can take various structures, as depicted in Figure 6.2 (b)-(c)-(d).

Non-adaptive attention. In a non-adaptive approach, the attention mechanism is used to compute the context vector for each word generation, even non-visual words that are not depending on the image such as language connection words.

Adaptive attention. Attending to images for generating non-visual words can be misleading and degrade performance. This degradation could result from using

visual information for non-visual words. To alleviate this problem, the authors (Lu et al., 2017) propose a formulation for a learnable gate variable β. The variable β is learned to choose either to rely on the image features or only on the context of language generation through the visual sentinel vector. Then, this idea was expanded, and learnable gate variables were extended over several branches of feature extraction. For example, (Guo et al., 2019) incorporate a module using context-gated attention, taking as input at each step the visual semantic unit embeddings, and aligning words hierarchically. Their approach first identifies the type of visual semantic unit related to the word (object, attribute, interaction) through the gate variables, then finds the most correlated unit within that type.

Guided attention. External information such as object proposals or attention maps are used to guide the focus of attention model. This allows the model to attend to specific regions of an image that are relevant to the description, instead of relying on a fixed or learned attention mechanism. The use of guided attention has been shown to improve performance and accuracy of image captioning models, particularly in scenarios where attention mechanism is not able to effectively capture the relevant image content on its own. In (Liu et al., 2017), the authors present a supervised model for attention and alignment annotation. The model learns a gating variable to distinguish between visual and non-visual words, while attention maps are supervised using bounding box associated with each target word in the caption.

6.2.2 Video captioning

Unlike image captioning, which focuses on describing the static content of an image, video captioning requires a more nuanced understanding of the underlying motion and dynamics in the video. The temporal dimension is involved, and architectures relies on spatio-temporal attention mechanisms. Different types of features were used in the context of video captioning.

Motion features. Capture the dynamics of the objects and actions in the video, such as their movement, velocity, and acceleration. These features can help the captioning model to understand the interactions between objects and the overall flow of the scene. Motion features are often extracted using optical flow, skeleton-based representations, or using a 3D Convolutional neural network (C3D) (Song et al., 2017).

Appearance features. Capture the visual content of the video, such as object categories, textures, and colors. These features help the captioning model understand the objects present in the video. Often extracted using convolutional neural networks (CNNs) pretrained on large image datasets.

Spatial attention. Focuses on the key objects or regions within a video frame. The attention mechanisms are generally designed as standalone components within the captioning model and can be trained to recognize the significance of various frame regions in influencing the generation of specific words in the caption.

Temporal attention. Selects the most pertinent frames for the generation of a specific word at a particular time step.

By combining both motion and appearance features, video captioning models can gain a more complete understanding of the video. While spatial and temporal attention allow the model to focus on the most relevant information in the video and to generate more accurate and descriptive captions. For instance, authors of (Song et al., 2017) propose a comparison of different feature extraction methods: *Motion/Appearance* features and *Spatial/Temporal* attention. Their architecture achieves superior performance in video captioning by leveraging spatial and temporal features, which are trained separately.

6.3 Methods

As discussed in the previous Chapters (4 and 5), motion captioning was tackled with a few methods based on RNNs and Transformers. The previous works have not considered the graph structure of the skeleton or focused on interpretability and understanding of the model's reasoning. To overcome these challenging limitations, we propose various methodologies for designing an interpretable and efficient architecture.

6.3.1 Captioning with spatio-temporal information

We first present the general proposed architecture for our captioning approach, giving a general overview, followed by a detailed presentation of the various components and finally a presentation of our training protocol. Our model, summarized in Figure 6.3, is composed of an encoder block, a spatio-temporal attention block and a text generation/decoder block.

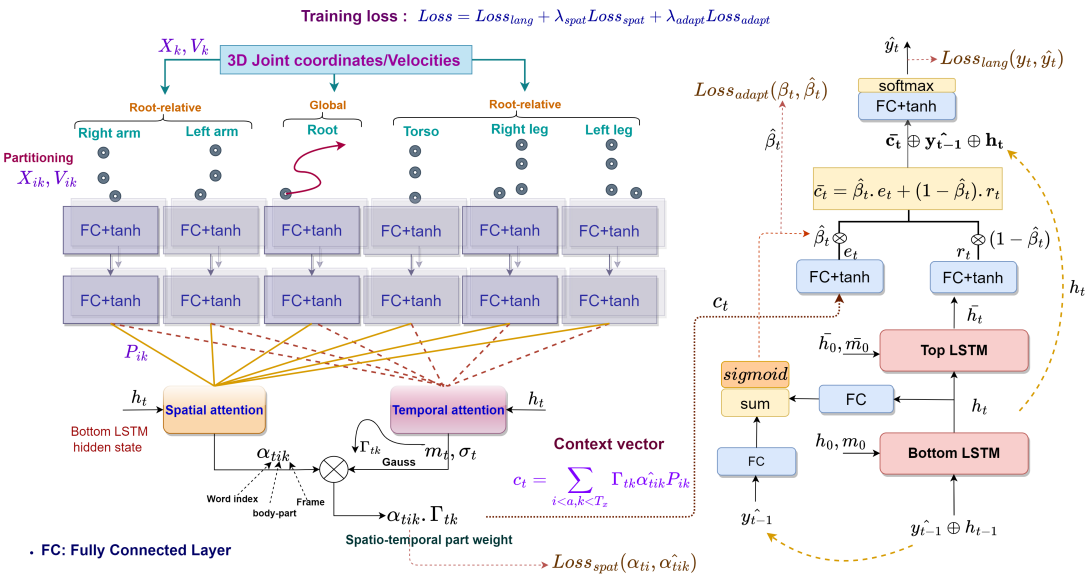


FIGURE 6.3: Our architecture design with an MLP-based encoder, and Two LSTM-based decoders (Radouane et al., 2023b).

Architecture Overview. Figure 6.3 fully illustrates our model design. The encoder branch encodes frame-wise part-based motion representations from joint positions (X_{ik}) and velocities (V_{ik}), while the decoder branch takes as input both the text sequence (previous tokens $\hat{y_{t-1}}$, hidden state h_{t-1}) and estimates the relative importance ($\hat{\beta}_t$ gate) of motion information for word prediction $\hat{y}_t$. Spatial ($\hat{\alpha_{tik}}$) and temporal attention Γ_{tk} are computed from encoded part embeddings of the motion encoder and the current hidden state (h_t) of the bottom LSTM. These spatio-temporal weights are utilized to compute the context vector c_t encoding attention-weighted part representations. Both the context vector and the output of the top LSTM pass through MLP layers with *tanh* activation, before being adaptively merged, yielding an adaptive context vector $\bar{c}_t$. This $\bar{c}_t$ is concatenated with the previously predicted tokens $\hat{y_{t-1}}$ and h_t to form the input to the final linear layer of the decoder. The overall loss is a weighted sum of losses optimizing different parts of the architecture. $Loss_{lang}$, the cross entropy between predicted, and target words is the main loss, to which we add a spatial loss, $Loss_{spat}$, guiding attention, as well as an adaptation loss, $Loss_{adapt}$, for the supervision of the $\hat{\beta}_t$ gate.

Formal notations. Let $\mathbf{X} \in \mathbb{R}^{T_x \times J \times D}$ be the input sequence of motion features of T_x time steps, where J is the number of joints in the skeleton and D is a number of spatial dimensions. We note X_k the 3D joints positions and V_k their corresponding

velocities at frame time k.

$$\mathbf{X}_k = [\mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \cdots, \mathbf{x}_{k,J}]$$

$$\mathbf{V}_k = [\mathbf{x}_{k+1,1} - \mathbf{x}_{k,1}, \cdots, \mathbf{x}_{k+1,J} - \mathbf{x}_{k,J}]$$

Body-part partitioning. We group the joints in 6 parts: Left Arm, Right Arm, Torso, Left Leg, Right Leg, Root. We convert the global coordinates to root-relative coordinates, except for the root itself, which describes the global trajectory of the motion. X_{ik} denotes the group of joints of part i for every frame k as described in Figure 6.3.

Encoder. Each of the six body parts is embedded by an MLP of two layers with *tanh* as activation function, as illustrated in Figure 6.3. The MLP encode positions X_{ik} and velocities V_{ik} separately. The final embedding P_{ik} for a given part i and frame k is the concatenation of the position and velocity embeddings. We note by P the frame-level motion features of all human body parts. $P \in \mathbb{R}^{T_x \times a \times h_{enc}}$ where h_{enc} stands for the dimension of the final output encoder and $a = 6$ is the number of body parts.

$$\mathbf{P} = \mathbf{Enc}(\mathbf{X})$$

Decoder. We adopt a two-LSTM decoder configuration, a *Bottom LSTM* for learning attention weights and language context, and a *Top LSTM* for final word generation based on the relevant information extracted from language and motion (Song et al., 2017). We note by $\mathbf{y} = (y_1, \ldots, y_{T_y})$, $y_i \in \mathbb{R}^{K_y}$ the sequence of words describing the motion. Let $h_t \in \mathbb{R}^{h_{dec}}$ be the decoder hidden state of the bottom LSTM for a word w_t in the sequence and $\bar{h}_t$ for the Top LSTM. We note by K_y the size of the target vocabulary, and T_x and T_y are respectively the lengths of the motion sequence and its description. The decoder *Dec* is used to predict the next word y_t given the adaptive context vector $\bar{c}_t$, the previous word y_{t-1}, and the bottom hidden state h_t.

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, \bar{c}_t) = Dec(y_{t-1}, h_t, \bar{c}_t) \tag{6.1}$$

The context vector c_t is computed by a spatio-temporal attention mechanism: temporal attention determines when to focus attention, and spatial attention determines where to focus in the part-based graph.

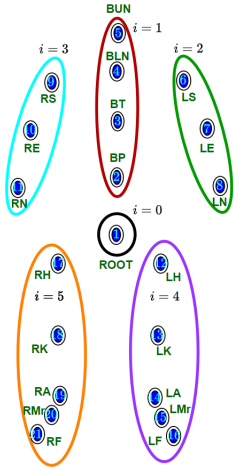


FIGURE 6.4: Partitioning of KIT skeleton of 21 joints into 6 body parts (22 joints for Human-ML3D) (Radouane et al., 2023b).

For both *Top LSTM* and *Bottom LSTM* we initialize the hidden and memory states by zero vectors. This forces the network to focus and get information only through the adaptive context vector, which is important for learning correct attention maps.

6.3.2 Proposition of attention mechanisms

In this part, we will provide detailed formal definitions we have proposed for each attention-based operation, as depicted in Figure 6.3. Staring with the two blocks *Spatial attention* and *Temporal attention*. Subsequently, we give details about formal definitions of each vector involved in the process of decoding motion into text description.

• **Spatio-temporal attention.** The spatial and temporal weights are computed using the extracted motion features P and the bottom lstm hidden state h_t. In the following, we denote by $P^* \in \mathbb{R}^{h_{enc} \times a \times T_x}$ the permutation of $P \in \mathbb{R}^{T_x \times a \times h_{enc}}$.

Temporal attention formulation. The temporal weights are computed from extracted motion features P^* and the current decoder hidden state h_t.

$$z_t = \boldsymbol{w}_h^T \tanh(\boldsymbol{W}_p \boldsymbol{P^*} + ep(\boldsymbol{W}_h \boldsymbol{h}_t)) \tag{6.2}$$

$$\gamma_t = \text{softmax}(z_t) \tag{6.3}$$

The notations $\boldsymbol{W}_p \in \mathbb{R}^{d \times h_{enc}}$,$\boldsymbol{W}_h \in \mathbb{R}^{d \times h_{dec}}$ and $\boldsymbol{w}_h \in \mathbb{R}^{d \times 1}$ refer to learnable parameters, ep is an expansion operator mapping to $d \times a \times T_x$, where a is the

number of body parts ($a = 6$). The vector of temporal attention weights, γ_t has the explicit form $\gamma_t = [\gamma_{t,1}, \gamma_{t,2}, \cdots, \gamma_{t,T_x}]$, with $\gamma_{t,k}$ being the attention score for the frame k for the word generated at time t.

Gaussian temporal attention. With the above formulation, we often encounter discontinuities in attention maps. However, these discontinuities are undesired, as the action occurs continuously within a given frame range. The distribution of attention weights for a given motion word can be modelled as a Gaussian distribution with a learnable mean and standard deviation. The mean m_t and standard deviation σ_t are computed from the previous temporal attention weights γ_{tk}, which are replaced by Γ_{tk} during training in this case (cf. Figure 6.3). Intuitively, the mean m_t will approximately represent the center time of action duration described by a motion word w_t, and the spread of the distribution approximately corresponds to the duration of the action.

$$\Gamma_{tj} = \exp\left(-\frac{(j - m_t)^2}{2{\sigma_t}^2}\right) \tag{6.4}$$

Spatial attention formulation. Spatial attention was widely explored in skeleton-based action recognition. Given our part-based design, attention weights are computed per human skeleton body parts (*e.g.*, Torso, left/right arm, left/right leg)

$$\boldsymbol{s}_t = \boldsymbol{w}_s^T \tanh\left(\boldsymbol{W}_{p_s}\boldsymbol{P}^* + ep(\boldsymbol{W}_{h_s}\boldsymbol{h}_t)\right) \tag{6.5}$$

$$\boldsymbol{\alpha}_t = \text{softmax}\left(\boldsymbol{s}_t\right) \tag{6.6}$$

Where $s_t \in \mathbb{R}^a$. The learnable parameters are $\boldsymbol{W}_{p_s} \in \mathbb{R}^{d \times h_{enc}}$,$\boldsymbol{W}_{h_s} \in \mathbb{R}^{d \times h_{dec}}$ and $\boldsymbol{w}_s \in \mathbb{R}^{d \times 1}$. We note by $\alpha_{t,m,k}$ the spatial attention score for part m of the skeleton graph at frame k for the word generated at time t. Thus, explicitly $\alpha_t = [\alpha_{t,1,1}, \alpha_{t,1,2}, \cdots, \alpha_{t,a,T_x}]$.

• **Adaptive attention.** As discussed before in the context of vision-based captioning, similarly attending to the inputs for all words is not necessary. Non-action words, particularly grammatical words, do not carry any information about the movement. Thus, we propose to learn a gating variable $\hat{\beta}_t$ to decide the proportion to which to use language context over motion features:

$$\hat{\beta}_t = sigmoid(W_b^h.h_t + W_e.(Ey_{t-1})) \tag{6.7}$$

Where $W_b^h \in \mathbb{R}^{1 \times h_{dec}}, W_e \in \mathbb{R}^{1 \times d_{emb}}$ are learnable matrices. $E \in \mathbb{R}^{d_{emb} \times K_y}$ refers

to embedding matrix of target vocabulary. The gating variable depends on the hidden state, which encodes residual information about generated words up to time step t, as well as on the embedding of the previous word, as detailed in Equation 6.7.

6.3.3 Language and motion encoding

As depicted in Figure 6.3, the final prediction relies on adaptive selection, where the learnable gate determines the "amount" of motion information to utilize for predicting a word at the current time step. Next, we formally define the context vector encoding this motion information and the adaptive selection process:

Context vector c_t. The context vector is derived by weighting the motion features with spatial and temporal attention weights as follows:

$$c_t = \sum_{k=1}^{T_x} \sum_{i=1}^{a} \Gamma_{tk} \alpha_{tik} P_{ik} \tag{6.8}$$

The motion information c_t and language information $\bar{h}_t$ are embedded into the same space through a linear layer with *tanh* activation (values in [-1,1]), giving e_t and r_t respectively.

Adaptive context vector $\bar{c}_t$. The adaptive context vector is given by Equation 6.9. When $\hat{\beta}_t = 1$ the model uses full motion information and when $\hat{\beta}_t$ is close to 0 the model relies more on language structure.

$$\bar{c}_t = \hat{\beta}_t . e_t + (1 - \hat{\beta}_t) . r_t \tag{6.9}$$

Final layer W_f. The probability outputs are computed using Equation 6.10, which is similar to previous work in video captioning (Song et al., 2017). Except that we include the bottom hidden state. This ensures that the language information of previously generated words is always retained, which is useful for maintaining correct syntax, even for motion-related words (*e.g.*, run, running, etc).

$$p(y_t \mid y_{1:t-1}, \hat{c}_t) = softmax(\tanh(W_f . concat([\bar{c}_t; y_{t-1}; h_t]))) \tag{6.10}$$

6.3.4 Spatial and adaptive attention supervision

Interpretability is one of the main focuses in our work. We propose guiding attention through the supervision of spatial weights and gate variables. This process

Category	Words	Body part
Trajectory	circle, circuit, clockwise, anticlockwise, forward, backward	Root
Local motion	open, waves, wipe, throw, punch, pick, boxing, clean, swipe, catch, handstand, draw	Arms
	kick, stomp, lift, kneel, squat, squad, stand, stumble, rotate	Legs
	bend, bow	Torso
Connection words	a,is, the, of, his, her, its, on, their	-
Subject	person, human, man	-

TABLE 6.1: Predefined dictionary for both datasets.

is intended to produce attention patterns that match human-like scene perception and analysis. However, the end goal remains to enhance the quality of text generation. Later, we investigate the effect of added interpretability on performance. To our knowledge, the supervision of the attention mechanisms with an adaptive gate and spatial attention have never been applied to captioning tasks, let alone motion captioning. Consequently, we first introduce the detailed definitions of our proposed methods for attention loss supervision and ground truth generation.

Ground truth generation for supervision. Both supervision methods are based on a predefined dictionary. We manually define a dictionary based on representative words in the dataset describing different motion characteristics. Intentionally, the dictionary doesn't cover all dataset actions with their synonyms, we want the model to be able to generalize to remaining unsupervised words for their spatial and gate attention. We will see later that the model effectively converges for this intended behavior. During training, the words in Table 6.1, and targets words, are stemmed to find correspondence for spatial weight supervision.

• **Spatial ground truth.** The ground truth spatial attention weights α_{ti} are generated based on the predefined dictionary and it's same for all frames (cf. Equation 6.11), the temporal attention performs temporal filtering.

$$\forall k \in [0, T_x - 1] : \alpha_{tik} = \alpha_{ti} \tag{6.11}$$

• **Gate ground truth.** In order to build a ground truth for adaptive attention, we define a mapping to distinguish motion words based on the Part Of Speech (POS) tagging. Then, ground truth $\beta_t = 1$ is assigned for categorized motion words and 0 for others.

For the remaining uncategorized motion words, we do not supervise attention scores, but let them be learned or inferred from supervised attention of synonym words. By this mechanism, we push the network to determine the relevant word to output based on the most attended body part and self generalize to the unsupervised words. Since the exact time of the action is unknown, we keep the ground truth weight α_{tik} frame time-independent (cf. Equation 6.11). The temporal attention block focuses on learning action time and performing temporal filtering of spatial weights through element-wise multiplication (cf. Figure 6.3).

Global loss definition. To define the global loss, we add the loss terms for spatial attention $loss_{spat}$, adaptive attention gate $loss_{adapt}$, respectively weighted by $\lambda_{spat}, \lambda_{adapt}$, to control their contributions.

$$Loss = loss_{lang} + \lambda_{spat}.loss_{spat} + \lambda_{adapt}.loss_{adapt} \tag{6.12}$$

In the following all losses are formulated sample-wise (sample x with source length T_x), we omit batch averaging in the notations for the sake of brevity.

• **Language loss.** The standard loss for motion-to-text generation is $loss_{lang}$ defined as the cross entropy between the target and predicted words:

$$Loss_{lang} = -\sum_{j=1}^{T_y} y_j \log(\hat{y}_j) \tag{6.13}$$

• **Adaptive loss.** We assign $\beta_t = 1$ for motion words and $\beta_t = 0$ for non-motion words, the loss is simply defined as:

$$Loss_{adapt} = -\sum_{y} \frac{1}{T_y} \sum_{t=0}^{T_y - 1} \beta_t \log(\hat{\beta}_t) + (1 - \beta_t) \log(1 - \hat{\beta}_t) \tag{6.14}$$

• **Spatial loss.** The predicted attention score is $\hat{\alpha_{tik}}$ for a given word w_t and part i of the source motion at the frame k. The loss is formulated in Equation 6.15, where N_y is a normalization factor that count the number of supervised words for a given target description y.

$$Loss_{spat} = -\frac{1}{N_y} \sum_{i,t,k} \alpha_{ti} \log(\hat{\alpha_{ti}}) + (1 - \alpha_{tik}) \log(1 - \hat{\alpha_{tik}}) \tag{6.15}$$

The spatial attention is guided by focusing attention on the human body parts through the ground-truth scores α_{tik}.

6.4 Experiments

This section presents the implementation details for interpretable motion captioning (Section 6.4.1) and quantitative results (Section 6.4.2). Then, we investigate the model interpretability (Section 6.5).

6.4.1 Training and hyperparameters

The architecture design is similar for both datasets, except for the input dimensions and the hyperparameters, which are defined as follows:

For KIT-ML, the word embedding size is set to $d_{emb} = 64$, the decoder hidden size to $h_{dec} = 128$, the dimension of each output of MLP_i for layer 1 is 128 and 64 for layer 2, for joint positions and for velocities. After concatenation, we obtain 128 joint-velocity features per frame.

For Human-ML3D, the word embedding size is set to $d_{emb} = 128$, the decoder hidden size to $h_{dec} = 256$, the dimension output of MLP_i for layer 1 to 256 and to 128 for layer 2, which is the final output dimension for joint positions and for velocities. After concatenation, we obtain 256 joint-velocity features per frame.

We use the AdamW (Loshchilov and Hutter, 2017) optimizer with a weight decay of $1e-4$ and $1e-5$ respectively for KIT-ML and Human-ML3D, both with a teacher forcing strategy of ratio 0.5. For loss supervision, we configure a search space for $(\lambda_{spat}, \lambda_{adapt})$ and run the search using WandB (Biewald, 2020).

6.4.2 Quantitative evaluation

In the following, all evaluations are conducted on the same dataset splits as other state-of-the-art (SOTA) systems, employing standard text generation metrics widely used for captioning tasks and motion-to-language models.

Loss weight ablations. We run experiments for different values of $(\lambda_{spat}, \lambda_{adapt})$. The quantitative results are reported in Table 6.2.

Dataset	λ_{spat}	λ_{adapt}	BLEU@1	BLEU@4	CIDEr	ROUGE$_L$	BERTScore ↑
KIT-ML	0	0	57.3	23.6	109.9	57.8	41.1
	0	3	56.3	22.5	108.4	56.5	39.8
	1	3	57.6	23.5	102.6	57.2	40.1
	2	**3**	**58.4**	**24.4**	**112.1**	**58.3**	**41.2**
	3	5	57.6	23.7	105.7	57.5	40.9
	5	5	56.5	22.0	99.4	56.8	39.9
HML3D	0	0	69.3	24.0	58.8	54.8	38.7
	0	**3**	**69.9**	**25.0**	61.6	**55.3**	**40.3**
	0.1	3	69.5	23.8	58.7	55.0	38.9
	0.25	3	68.7	23.8	59.7	54.7	39.3
	0.5	3	68.8	23.8	60.0	55.0	38.6
	1	3	68.7	23.7	58.2	54.6	39.0
	2	3	69.2	24.4	**61.7**	55.0	40.3
	3	3	68.3	23.2	56.5	54.5	37.1

TABLE 6.2: Spat+adapt supervision impact w.r.t each corresponding weights.

Impact of adaptive and spatial supervision. We train the model for both datasets with different combinations of (λ_{spat} and λ_{adapt}), where zero corresponds to the case without any supervision. Table 6.2 presents the performance comparison of our system for the considered combinations, using common text generation evaluation metrics. The gate (*adapt*) and spatial (*spat*) supervision perform well when used together on KIT-ML (small). We achieve a BLEU@4 score of 24.4 (+0.8%) compared to 23.6 obtained with no gate and attention guidance. For Human-ML3D, adaptive attention consistently showed benefits, with a increase of 1% on BLEU@4 and 0.5% on ROUGE scores. However, combining adaptive attention with guided spatial attention resulted in a slight degradation of exact matching scores (BLEU@4, ROUGE) compared to using adaptive attention alone. We can hypothesize that guiding spatial attention effectively results in the generation of semantically equivalent sentences with a slightly more diverse vocabulary. Consequently, this could lead to a decrease in the exact n-gram matching metrics.

Evaluation against SOTA. Table 6.3 presents a comparison with state-of-the-art (SOTA) systems for KIT-ML and Human-ML3D. Only (Guo et al., 2022b) utilize the updated version of KIT-ML augmented by language and motion adaptations. They replicate several SOTA architectures on the same split of the dataset, which we also report. For Human-ML3D, we include systems released after the dataset, as well as some of their older baselines replicated on Human-ML3D.

Dataset	Model	BLEU@1	BLEU@4	ROUGE_L	CIDEr	Bertscore
KIT-ML	RAEs (Yamada et al., 2018)	30.6	0.10	25.7	8.00	0.40
	Seq2Seq(Att)	34.3	9.30	36.3	37.3	5.30
	SeqGAN (Goutsu and Inamura, 2021)	3.12	5.20	32.4	29.5	2.20
	TM2T w/o MT	42.8	14.7	39.9	60.1	18.9
	TM2T (Guo et al., 2022b)	46.7	18.4	44.2	79.5	23.0
	Ours-[Spat+adapt] (2,3)	**58.4**	**24.4**	**58.3**	**112.1**	**41.2**
HML3D	RAEs (Yamada et al., 2018)	33.3	10.2	37.5	22.1	10.7
	Seq2Seq(Att)	51.8	17.9	46.4	58.4	29.1
	SeqGAN (Goutsu and Inamura, 2021)	47.8	13.5	39.2	50.2	23.4
	TM2T w/o MT	59.5	21.2	47.8	68.3	34.9
	TM2T (Guo et al., 2022b)	61.7	22.3	49.2	**72.5**	37.8
	Ours-[adapt] (0,3)	**69.9**	**25.0**	**55.3**	61.6	**40.3**

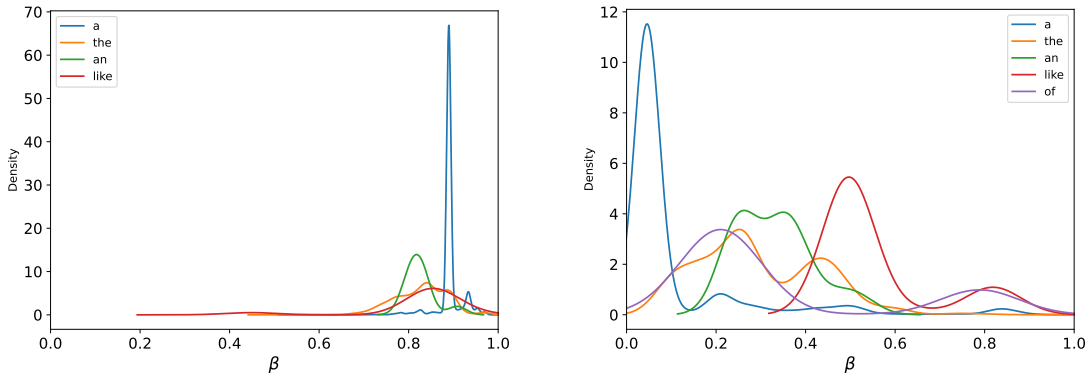
TABLE 6.3: Text generation performance compared with state-of-the-art approaches and baselines (Radouane et al., 2023b).

Our approach significantly outperforms other state-of-the-art approaches across all metrics on KIT-ML (with increases of +6% BLEU@4, +14.1% on ROUGE-L, +32.6% CIDEr, and +18.20% Bertscore). Similarly, it performs significantly better on Human-ML3D (with improvements of +2.7% BLEU@4, +6.1% ROUGE-L, and +2.5% Bertscore), except for CIDEr where there is a -10.9% difference. This includes comparison against the transformer-based TM2T. Moreover, our system demonstrates more consistent performance across datasets, with an order of magnitude fewer parameters, and yields interpretable outputs.

We hypothesize that considering the skeleton structure confers an advantage to our approach. This notion has been supported by its effectiveness in enhancing motion encoders for action recognition architectures, and it translates well to motion-to-language generation tasks. Additionally, our proposed method for attention guidance has resulted in improved metric scores, while providing interpretable outputs with greater transparency in the model's reasoning process. The qualitative and global analysis of interpretability will be further discussed in the following section.

6.4.3 Interpretability analysis and evaluation

In this section, we propose to conduct several analyses regarding the interpretability of the proposed architecture through a discussion on the effect of attention guiding on both the KIT-ML and the HumanML3D datasets.



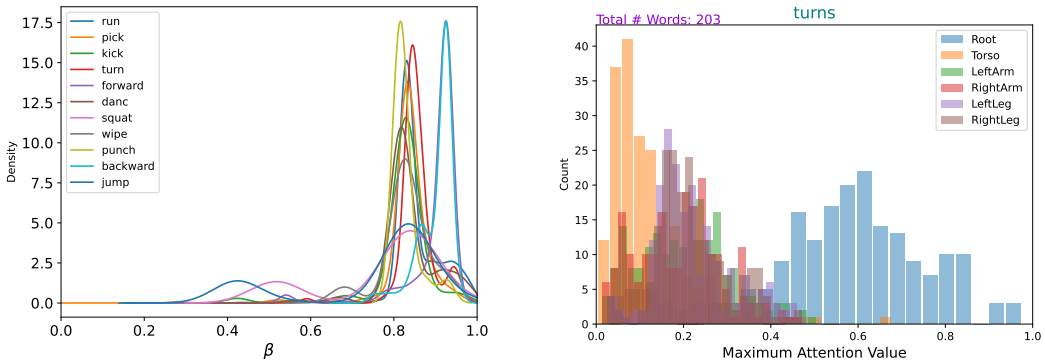
(a) Without gate supervision, decoder uses frequently motion information even for non-motion words (β frequently high).

(b) With gate supervision, the decoder uses correctly more language context for non-motion words (β frequently small).

FIGURE 6.5: β density distribution over test set for some non-motion words (stemmed) on Human-ML3D.

Adaptive attention effect. When training a model without guiding spatial attention, we observe that the $\hat{\beta}$ gate frequently takes higher values for non-motion words (a: 0.9, the: 0.8) as illustrated in Figure 6.5a. This behavior degrades performance, as seen in Table 6.2 for Human-ML3D. We hypothesize that this leads the weights of the spatio-temporal attention to receive more gradient updates for non-motion words through $\bar{c}_t$, which can make picking-out the more important motion words more challenging. However, when we introduce adaptive gate supervision (cf. Figure 6.5b), the model more frequently assigns a less weight $\hat{\beta}$ to non-motion words and begins to learn how to make decisions automatically. This allows the model to focus on using the context motion information only for motion words. Additionally, $\hat{\beta}$ values tend to be higher correctly for motion words describing: *trajectory*, *direction*, *action*, *body parts* as illustrated in Figure 6.6a. The adaptive attention supervision forces the model to prioritize the context vector for predicting motion words over non-motion ones, while guided spatial attention improves body part identification. We hypothesize that both types of supervision result in learning attention maps that better align with human visual perception.

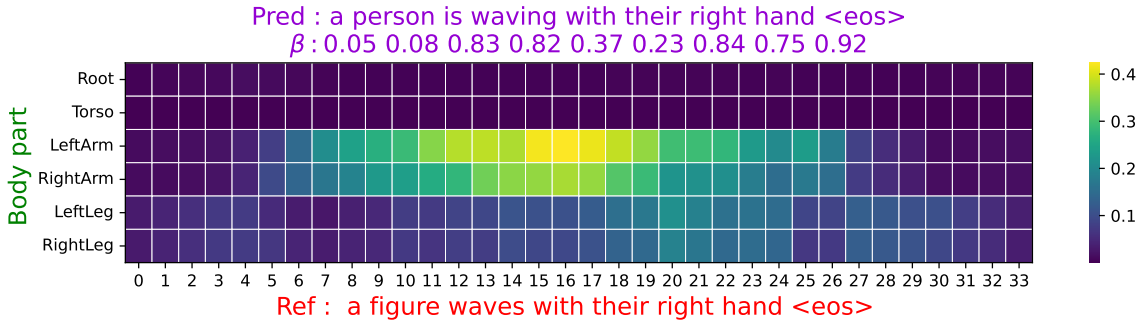
Spatio-temporal attention. The model's spatial attention effectively focuses on relevant parts for motion word prediction, as illustrated in Figure 6.7 for the motion words *"kicks"* and *"waves"*. Additionally, the temporal attention provides relevant information about actions. We will discuss these two aspects of body part identification and action localization in the following.



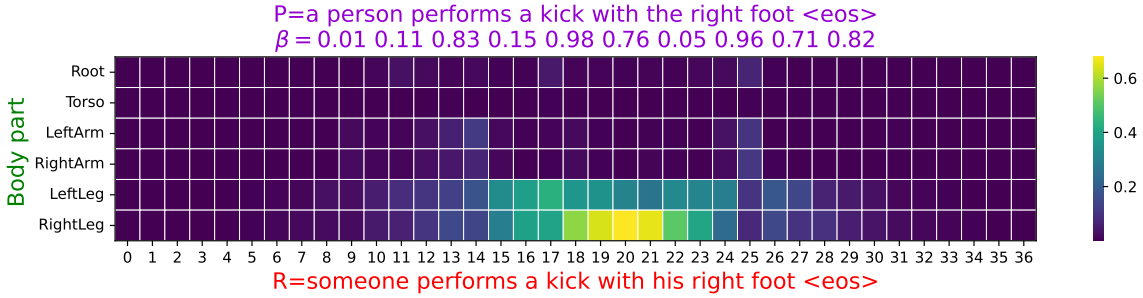
(a) With gate supervision, motion information is correctly used frequently for motion-words generation.

(b) Attention is frequently focused on relevant parts: e.g. on Root (global trajectory) for word "turns".

FIGURE 6.6: β test set density distribution for a few motion words stems on Human-ML3D and the temporal maximum body-parts attention histogram for word "turn".



(a) HML3D-(2,3), word *waving*



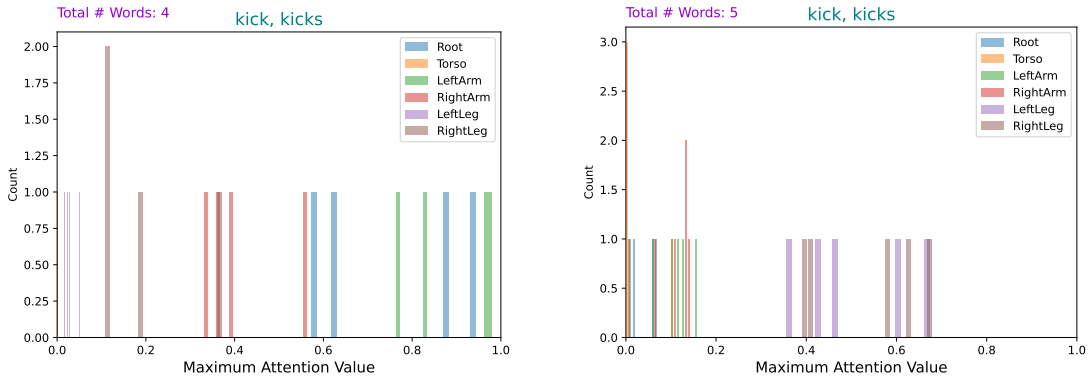
(b) KIT-(2,3), word *kick*

FIGURE 6.7: Spatial temporal attention map for different motion words. The vertical axis represents body parts, while the horizontal axis represents frames. The color scale represents the intensity of the attention score for the given body part at each frame.

• **Body part identification.** We can illustrate the effectiveness of our architecture in learning a correct body part association through spatio-temporal attention, by viewing the histogram for temporal maximum attention distribution for each body part given some motion words.

For KIT-ML, Figure 6.8 illustrates the compared spatial attention part weights with supervision (Figure 6.8b) versus without (Figure 6.8a) for the action *kick*. In the former, attention is concentrated on the legs, while in the latter it focuses on *Root* and *Left Arm*, which doesn't match the action. Quantitatively, this leads to an increase in performance, as previously seen (cf. Table 6.2).

For Human-ML3D, in Figure 6.9, the spatial attention correctly focuses on arms for *throw* in both cases (w and w/o supervision). We believe that with more training data, which includes more diverse descriptions, body part encoding alone may be sufficient to implicitly learn correct attention maps. However, for small-sized datasets, as seen for KIT-ML, spatial guidance is still required (cf. Table 6.2).

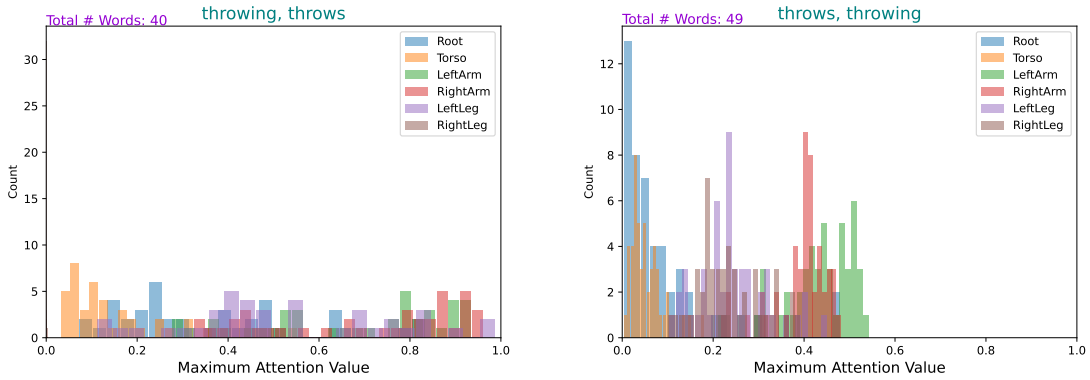


(a) Without spatial supervision, model incorrectly focuses on arms/root for the word "kicks".

(b) With spatial supervision, model focuses correctly on the relevant parts: e.g. legs for "kicks".

FIGURE 6.8: Effect of spatial supervision on KIT-ML.

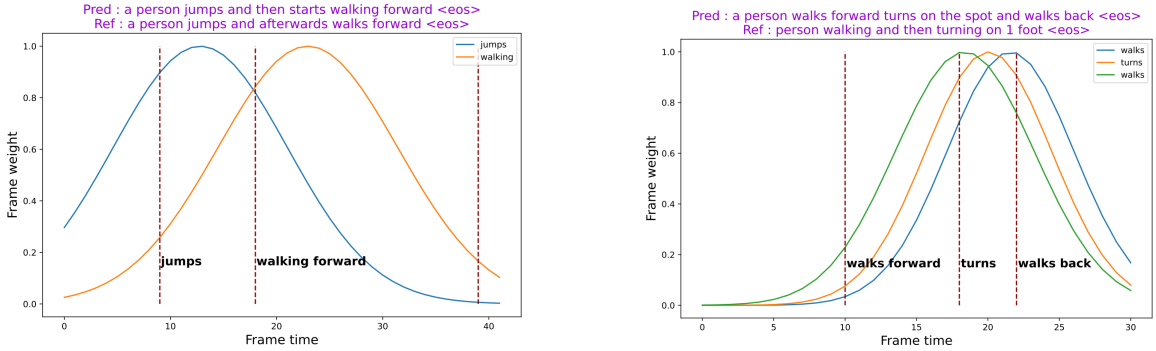
• **Action localization.** Another aspect that emerges from temporal gaussian attention weights is action localization. The architecture shows ability to identify motion onset without temporal supervision. We can derive the action onset from spatio-temporal attention maps as illustrated in Figure 6.10 where we also show their actual onset time (identified manually through 3D pose animation).



(a) Without spatial supervision.

(b) With spatial supervision.

FIGURE 6.9: Effect of spatial supervision on Human-ML3D.



(a) Multi-action.

(b) U-turn motion.

FIGURE 6.10: Temporal gaussian window displayed for different motion words given a prediction on KIT-ML.

6.5 Effectiveness of architecture components

In the following visualizations and discussions, we aim to demonstrate the global effectiveness of our architecture design for each component (cf. Figure 6.3) involved in learning an interpretable mapping between human motion and language:

- Functionality of gating mechanism.
- Attention mechanisms.
- Impact of Part based motion encoding with spatio-temporal attention blocks.

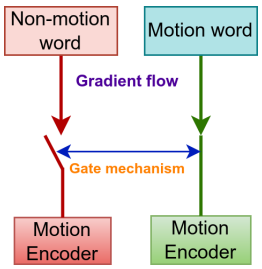


FIGURE 6.11: Illustration of our gating mechanism during training. The motion encoder is prevented from receiving important gradient updates for non-motion words.

6.5.1 Gating mechanism

The gate variable β allows the model to decide whether to use or not the motion information given by the word at the current time step. Figure 6.11 illustrates the effect of gating mechanism supervision in the optimization process. This mechanism prevents the decoder from attending to motion for non-motion word. Consequently, the motion encoder is prevented from receiving important gradients updates for non-motion words. To demonstrate the role of each of the context vectors c_t and LSTMs hidden states ($\bar{h}_t$, h_t) and visualize concretely this internal process of switching between motion and language during inference, we fix the $\hat{\beta}$ value at 1 and display predictions for the best model on Human-ML3D in Table 6.4 and on KIT-ML in Table 6.5, and show a representative examples compared to adaptive gate. The context vector ($\hat{\beta} = 1$) is successfully used for all words describing the motion characteristics: *action*, *speed*, *body parts*, *trajectory*, *direction*. While the hidden states provides the language structure and context. Particularly, we note that the end token `<eos>` is also motion related, as outputting this word depends on the end of the relevant human motion range. Interestingly synonym motion words are grouped (*e.g.*, *boxing, punching*). Our supervised gating mechanism succeeded in separating motion-words from other words effectively enhancing the generated text quality.

β (gate)	**Prediction**
1	***kicking*** kicks with right leg <eos>
adaptive	a person kicks with their right leg <eos>
1	***jumping jacks*** and jumping jacks <eos>
adaptive	a person does jumping jacks <eos>
1	***walking forward*** in a diagonal line <eos>
adaptive	a person walks forward in a straight line <eos>
1	***punching*** boxing and *moving hands around* <eos>
adaptive	a person is boxing with *both hands* <eos>
1	**jogs** in in place <eos>
adaptive	the person is jogging in place <eos>

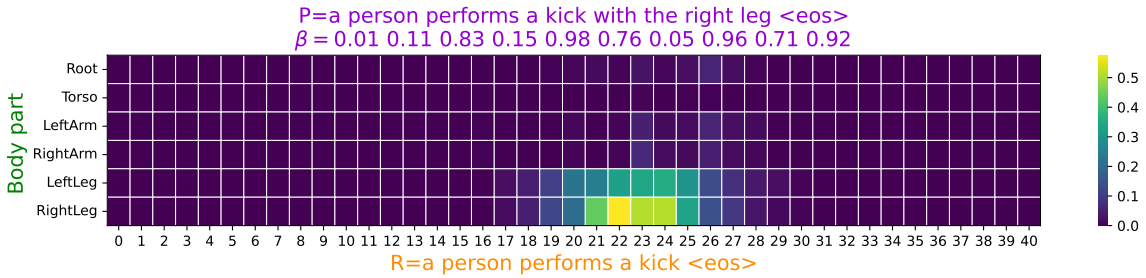
TABLE 6.4: Comparison of the predictions when setting $\hat{\beta} = 1$ and adaptive on Human-ML3D (adapt $(0, 3)$).

β **(gate)**	**Prediction**
1	waves waves waving with both hands <eos>
Adaptive	the person is waving both hands <eos>
REF	the person is waving both hands <eos>
1	walks walks slowly <eos>
Adaptive	a person walks slowly <eos>
REF	a person walks forwards quite slowly <eos>
1	kicking kicking kicking with left leg <eos>
Adaptive	a person kicks something with its left foot <eos>
REF	a human kicks something with his left foot <eos>
1	jumping jumps forward <eos>
Adaptive	a person jumps with both legs <eos>
REF	a person jumping 1 step <eos>
1	running running running <eos>
Adaptive	a person runs <eos>
REF	a person runs forward <eos>

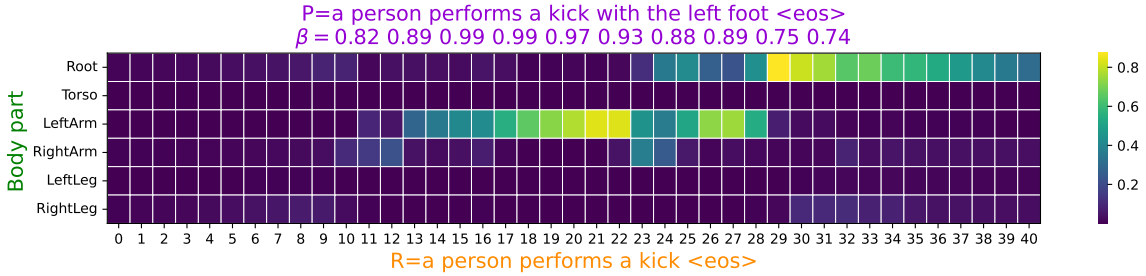
TABLE 6.5: Comparison of the prediction when setting $\hat{\beta} = 1$ and adaptive on KIT-ML (Spat+adapt $(2, 3)$).

6.5.2 Attention supervision

Spatial and adaptive supervision. We show comparison of spatio-temporal attention maps and text generated between the case with supervision and without supervision in Figure 6.12.



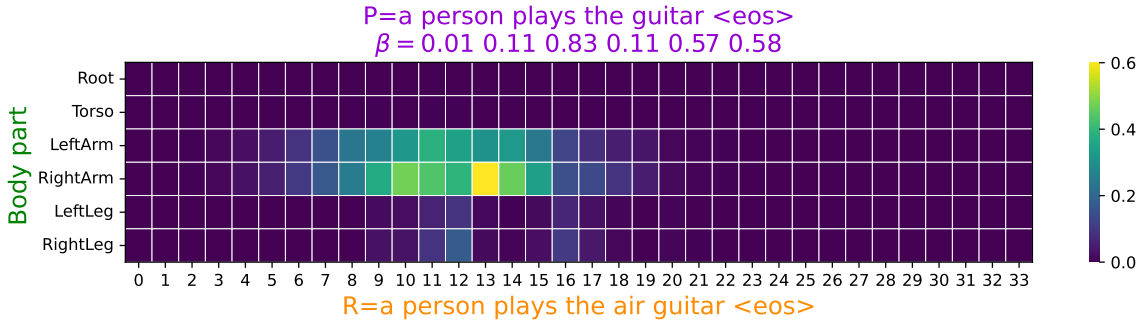
(a) With supervision KIT-(2,3) (action range [19,28]/right kick).



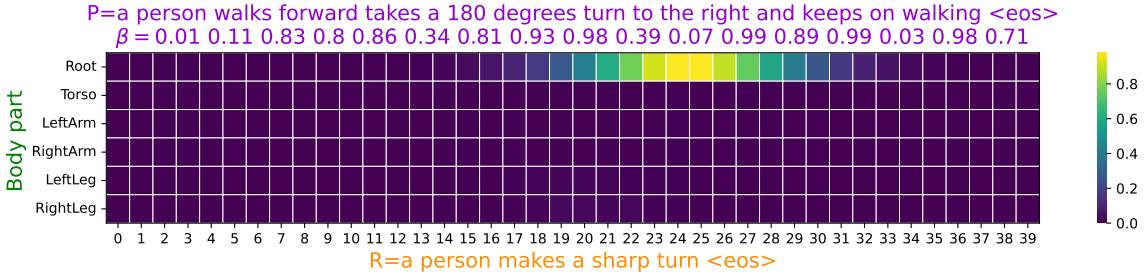
(b) Without supervision KIT-(0,0) (action range [19,27]/right kick).

FIGURE 6.12: Comparison between the case of guided attention and no supervision.

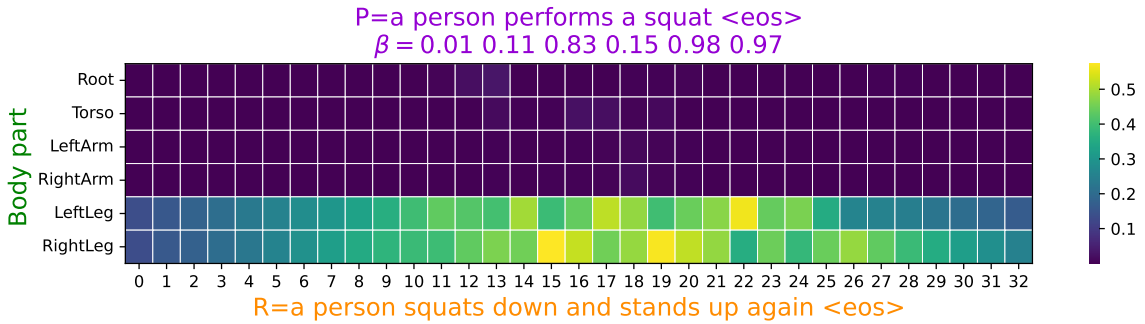
The case of supervision illustrated in Figure 6.12a shows that the relevant body parts were correctly identified, and corresponding action is perfectly localized in the range $[20, 26]$, which is very close to the manually identified range $[19, 28]$. Low predicted β values are associated with non-motion words. Without supervision (cf. Figure 6.12b), the model focuses on irrelevant parts, and consequently, the range of the action was not precisely localized. Additionally, the β values are unnecessarily high for all kinds of words. We visualize more samples (cf. Figure 6.13) with spatial and adaptive attention supervision. Temporal range is mentioned for comparison, even if action localization wasn't the main focus in the captioning task, the model was able to implicitly learn a temporal location through the temporal gaussian attention mechanism.



(a) Play (action range [10,20]).



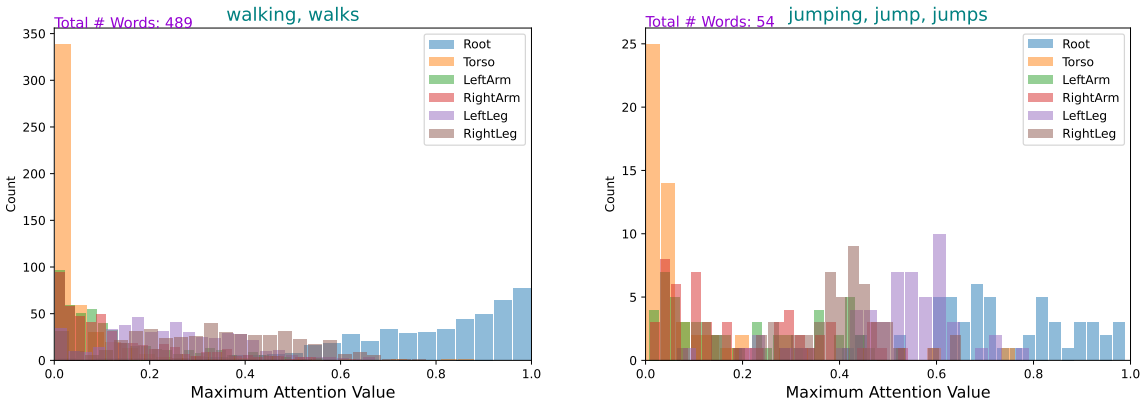
(b) Turn (action range [22,27]).



(c) Squat (action range [10,28]).

FIGURE 6.13: Spatio-temporal attention for different motion words on KIT-ML.

Trajectory and global motion. The attention was supervised only for words describing trajectory, but the model generalizes successfully to motion words that highly depend on a global trajectory. This results in maximum attention distributed toward the *Root* body part, as we see in Figure 6.14. For example, taking a non supervised motion word such as *walk* the maximum attention is assigned to Root body part, which effectively contains the global trajectory information.



(a) Walk.

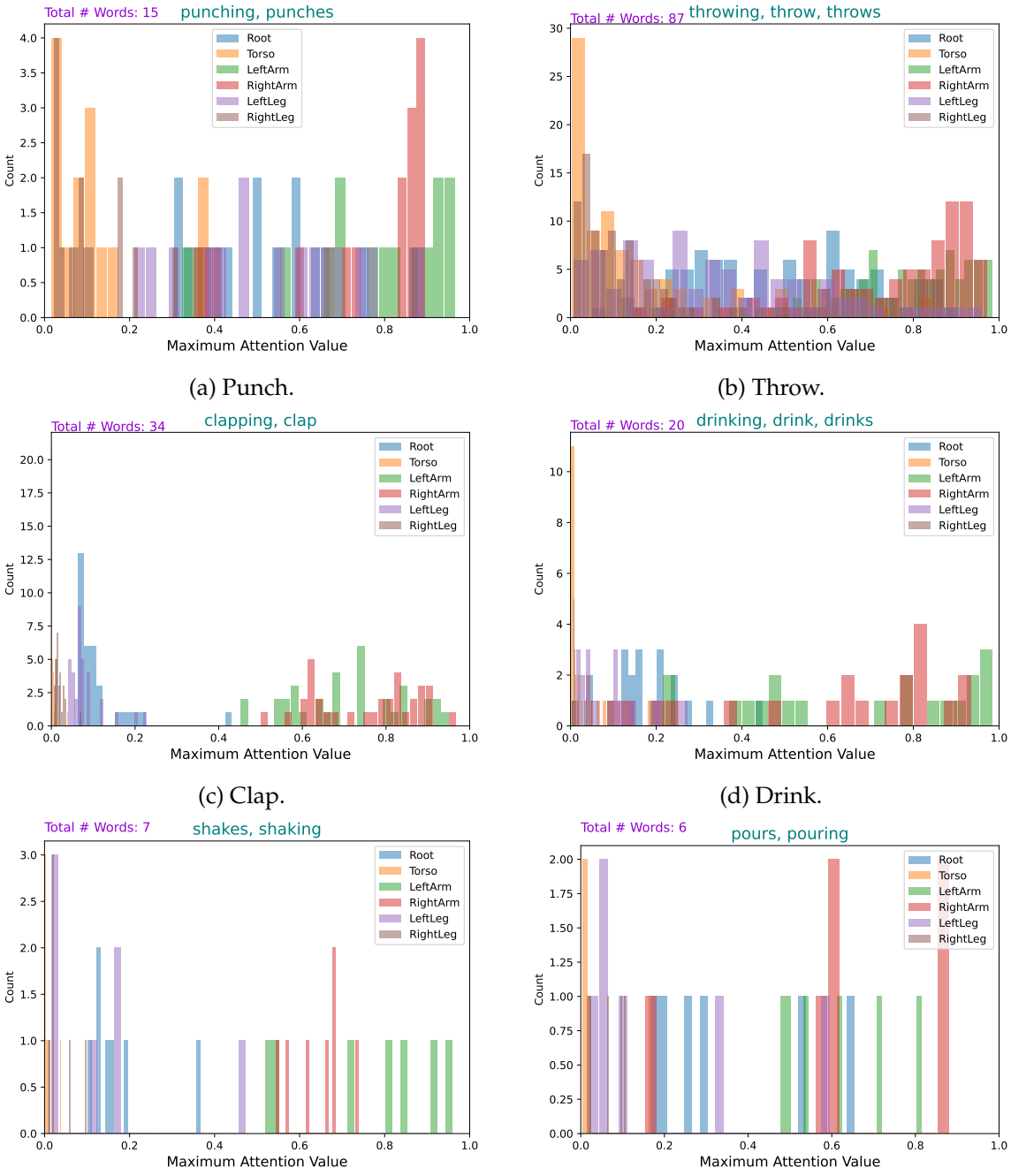
(b) Jump.

FIGURE 6.14: [KIT-(2,3)]: Body part distribution (spat+adapt).

6.5.3 Part based encoding & spatio-temporal attention

Our architecture design could be sufficient in learning correct spatial attention maps using a larger dataset with rich semantic descriptions. For the purpose of demonstration, we will use the model with *no spatial supervision*, to show that part-based encoding and spatio-temporal attention can work solely and correctly together for focusing on relevant body parts in relation to the associated generated motion words. To this purpose, we display the histogram distribution of temporal maximum attention weights for each body part over all the test set and given different motion words. This allows for an effective global evaluation of interpretability over the entire test set.

Histograms. In the following, we display the body parts histogram distribution across the test set for different motion words to demonstrate the effectiveness of part-based encoding along with spatio-temporal attention in finding relevant parts to focus on using the model ***with no spatial supervision***. This is only in the case of the larger dataset Human-ML3D. The KIT-ML small dataset still requires spatial supervision to help the architecture focus on relevant parts, as the vocabulary and its size are limited. This effectiveness is demonstrated in all the following Figures, depending on the motion word, arms-based/legs-based actions, and particularly some motions with an emphasis on the torso body part.



(a) Punch.

(b) Throw.

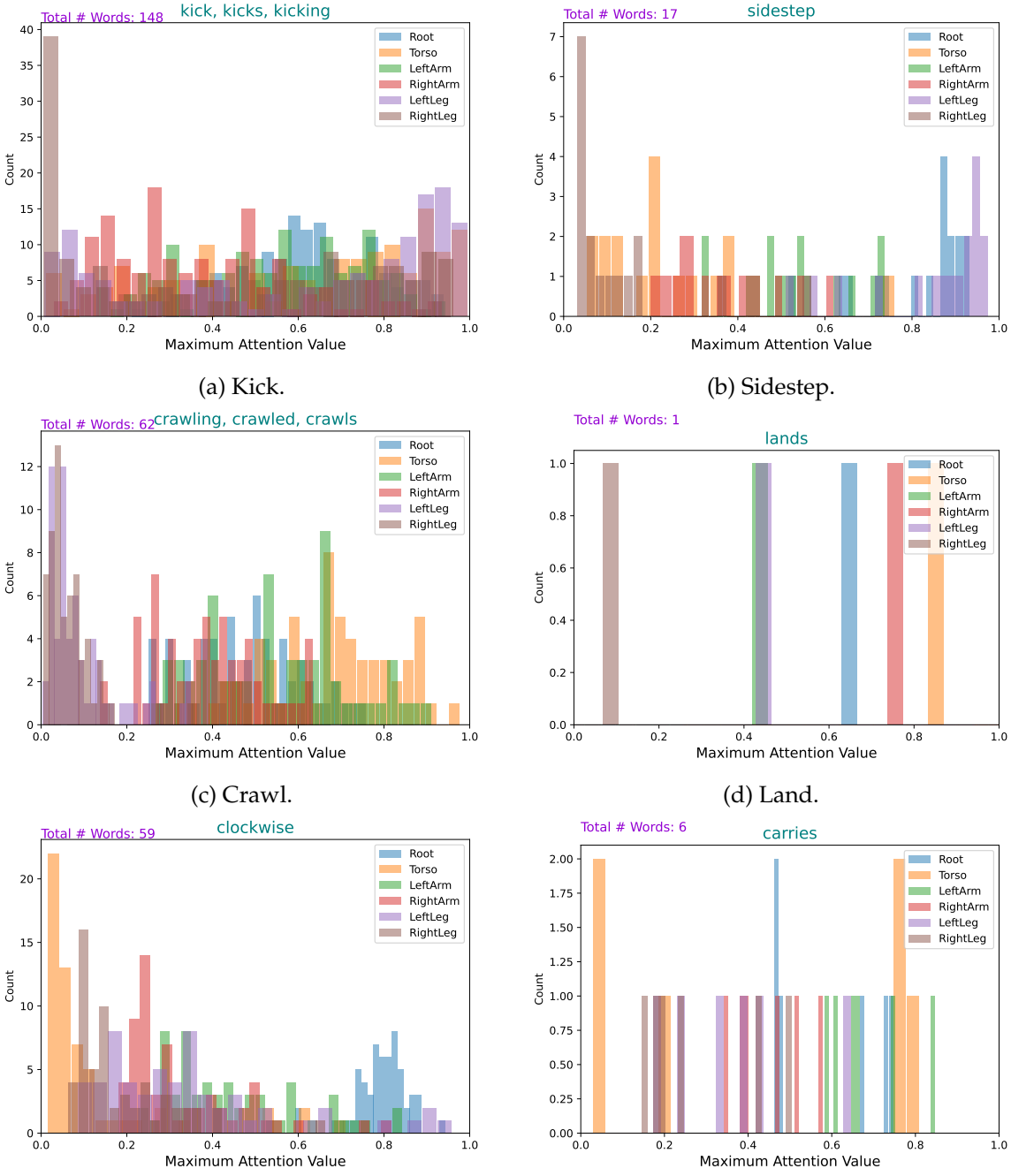
(c) Clap.

(d) Drink.

(e) Shake.

(f) Pour.

FIGURE 6.15: Histograms generated on the HML3D with the config (0,3).



(a) Kick.

(b) Sidestep.

(c) Crawl.

(d) Land.

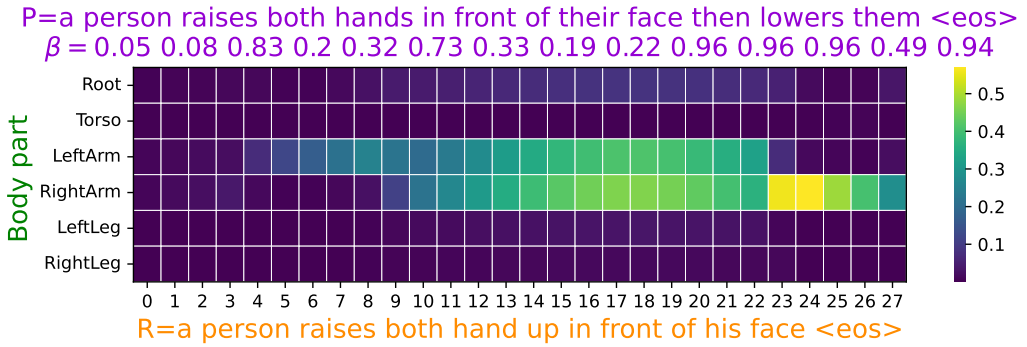
(e) Clockwise.

(f) Carries.

FIGURE 6.16: Histograms generated on HML3D with the config (0,3).

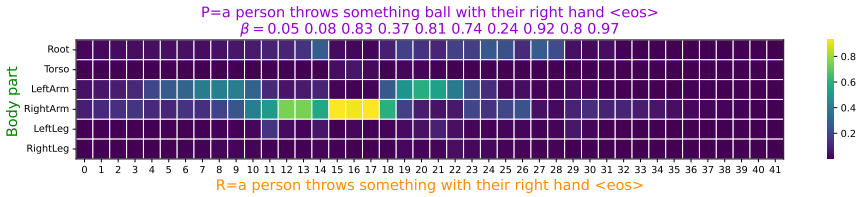
Spatio-temporal attention maps. In the case of the model without spatial supervision, we have found that the model performs a correct attention focusing. For actions performed with both arms or legs, the model focuses on both parts (cf. Figure 6.17). When an action is performed using right leg/arm, the model focuses correctly on the corresponding parts (cf. Figure 6.18). In all cases, body part words (left/right/both) are always accurately identified into the generated text, as shown by the prediction in Figure titles. These observations are common across different representative samples (from various actions).

(a) Stretches.

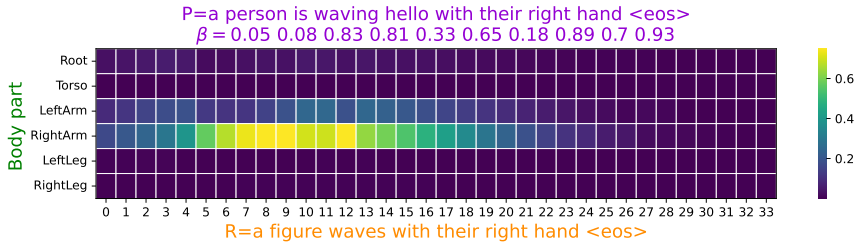


(b) Lowers.

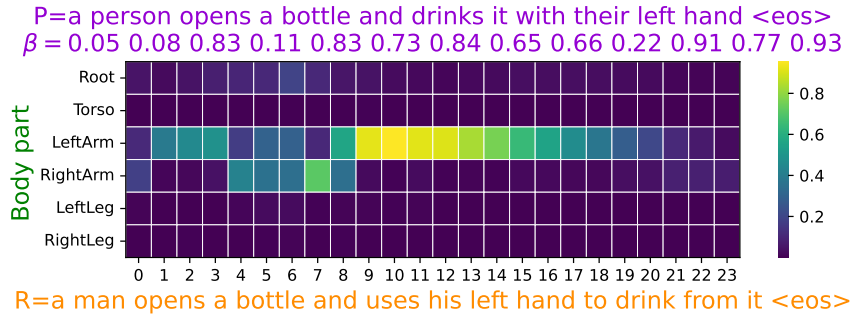
FIGURE 6.17: Bi-part based human motion (short).



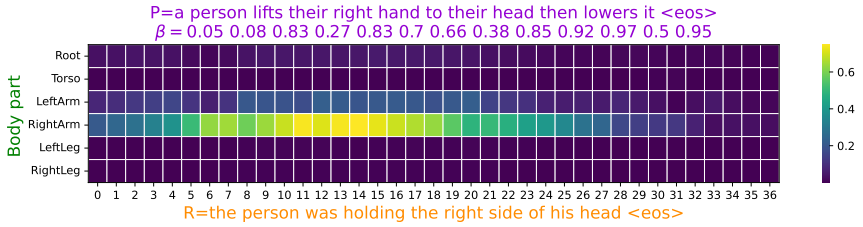
(a) Throws.



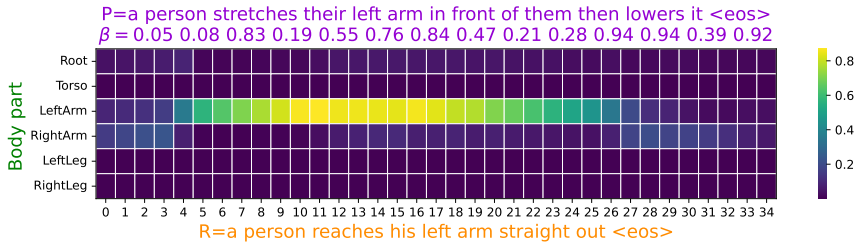
(b) Waving.



(c) Opens.



(d) Lifts.



(e) Stretches.

FIGURE 6.18: Single part-based human motion.

6.6 Transfer to adjacent tasks

Similar tasks such as action recognition and localization can benefit from the proposed formulations. For instance, our motion encoder and skeleton partitioning could be used for skeleton based action recognition, and the number of layers could be adapted based on the dataset's size and complexity.

For action localization in a continuous stream, this task could be approached as sequence-to-sequence learning. In this context, attention weights could be used to infer the start and end times of actions in an unsupervised manner without the need for action time labels. If the time annotations are available, they can be used to supervise the spread of temporal weights, further enhancing the accuracy of action localization and spatio-temporal attention maps.

Moreover, our formulation of spatial weights supervision could leverage prior knowledge about the body parts involved in a given action. In other scenarios, such as vision-based captioning, our spatial supervision could be adapted to maximize attention weights on spatial regions corresponding to the object described by each visual word in the caption. For image input, this spatial supervision could be applied directly, while for video input, a temporal attention block could be added. In both cases, the encoder could utilize a pretrained CNN network to extract spatial grid image features analogous to skeleton body parts. By incorporating the spatio-temporal formulation along with the supervision of adaptive learnable gates, emphasis can be placed on relevant visual words.

Finally, interpretability could be evaluated using the proposed density function for adaptive attention and histograms for attention distribution across spatial locations in various captioning contexts.

6.7 Conclusion

We have introduced guided attention with adaptive gate for motion captioning. After evaluating the influence of different weighting schemes for the main loss terms, we found that our approach leads to interpretable captioning while improving performance over the state of the art. Interpretability is important to consider when designing an architecture, as it provides insights into the model's capability to perform unbiased reasoning. This ensures the ability to generalize instead of memorizing. Our proposed model addresses these challenges by producing interpretable results with accurate semantic captions. Moreover, we have effectively

evaluated the global interpretability through the proposed density and histogram functions, which avoid sample-wise visualization, making it possible to see the effects across all test set samples. For language labeling, we utilized a rule-based language processing approach for word categorization. However, further performance improvement is likely achievable by proposing a more sophisticated and fine-grained semantic analysis of motion words in their context. The model and proposed methodology can be applied to other captioning tasks, as well as other contexts, such as supervising spatial attention weights in action recognition tasks.

Chapter 7

Conclusion and Perspectives

7.1 Summary of contributions

After the introduction, Chapters (2 and 3) focused on providing a recent literature review of human pose estimation and addressing practical challenges related to protective behavior based on human pose data (MoCap). In the rest, we addressed the problem of human motion captioning from various perspectives and for different objectives that went beyond merely generating text based on human motion.

- In Chapter 3, to address the challenge of detecting protective behaviors within a highly imbalanced dataset, we devised a modified architecture that outperformed the results achieved by other participants. Furthermore, we conducted an investigation into the explainability of the attention mechanism used in the binary classification process.

- In Chapter 4, our main concern has been on addressing motion captioning toward the generation of text aligned with motion, namely synchronized captioning. Thus, we conducted specific experiments given an additional and interesting challenge of motion and language alignment. The experiments were initially done on the original KIT-ML dataset, then after the release of an augmented version of KIT-ML and much larger dataset HumanML3D we did report our results on this new benchmark, outperforming the state-of-the-art approaches. Importantly, as a side product, we achieved semantic motion segmentation.

- In Chapter 5, we provide a detailed exploration of human motion segmentation through attention. We performed visual and quantitative assessments to evaluate the motion segmentation capabilities resulting from our proposed model in Chapter 4. Consequently, we introduced metrics for evaluating segmentation performance within the context of captioning, comparing performance both at the word and phrase level using attention weights aggregation.

- In Chapter 6, given the recent datasets, KIT-aug and HumanML3D, our primary objective was the development of a fully interpretable architectural design capable of performing spatial reasoning and temporal selection. Our design mirrors human-like analysis. To achieve this goal, we introduced a specialized formulation of spatial and temporal attention mechanisms. Moreover, we experimented with various attention guiding techniques aimed at

enhancing performance by increasing the model's interpretability. Furthermore, we provided tools to assess interpretability both qualitatively and quantitatively.

7.2 Perspectives

In this part, we discuss the potential use cases of the proposed methodologies and their relevance to real-world applications, while also highlighting the need for future research in the field.

7.2.1 Skeleton based action segmentation

Skeleton-based action segmentation employs both supervised and unsupervised techniques. Supervised methods use labeled data to learn and predict action boundaries, achieving good precision but requiring extensive annotations. Unsupervised approaches, on the other hand, rely on data clustering and temporal analysis, offering autonomy but potentially sacrificing precision. We can investigate the use of attention weights in both scenarios.

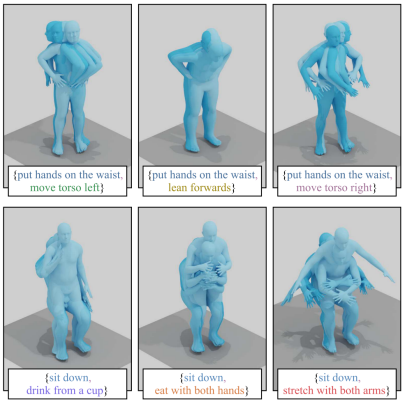
Supervised Segmentation. When the time annotation is available, it is possible to supervise attention weights to be distributed along the executed action for each action in the sequence. Furthermore, we could explore the recurrent aspect of our attention formulation and the temporal action segmentation could be treated as mapping a sequence of frames to a sequence of actions.

Unsupervised Segmentation. In this more challenging context, where there is no sufficient annotation information, the attention weights could be used to infer the action start and end. This may open a new path to novel approaches in the context of action segmentation that could be more accurate, replacing older methods based on clustering and other techniques.

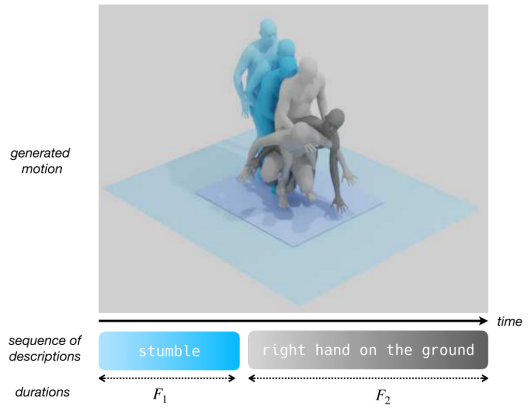
7.2.2 Synthetic motion generation with controlled annotation

Our proposed approach enables unsupervised learning for human motion segmentation. However, there is potential for the development of a more versatile and accurate architecture capable of handling scenarios beyond the training data

distribution. One promising avenue is to explore supervised learning for determining motion start and end times using automatically generated data. Several studies in the literature could be investigated to control the timing of human motion. Beginning with simple models that generate action-conditioned motion, as seen in (Guo et al., 2020). Drawing inspiration from Variational Autoencoders (VAE), more diverse generation techniques have been proposed, such as (Petrovich et al., 2021). Progressing further, models have been developed for more complex and fine-grained motion generation, including text-conditioned motion generation (Petrovich et al., 2022). Moreover, temporal action generation methods have been introduced TEACH (Athanasiou et al., 2022) (cf. Figure 7.1b). The same authors (Athanasiou et al., 2023) proposed another model SINC for spatial composition of 3D human motion (cf. Figure 7.1a). These recent advancements open the possibility of generating synthetic 3D human motion data with precise control over timing, allowing for automatic fine-grained annotation of actions and their localization. This, in turn, could facilitate pretraining models for motion segmentation through supervised learning.



(a) Spatial action composition (Athanasiou et al., 2023).



(b) Temporal action composition (Athanasiou et al., 2022).

FIGURE 7.1: Spatial and temporal human motion composition.

7.2.3 Sign Language

The primary objective of this task is to overcome communication barriers between hearing and Deaf individuals. Sign language research encompasses various intermediate tasks, including sign language recognition, localization, and alignment. Ultimately, the goal is to develop systems for Sign Language Production (SLP)

(Rastgoo et al., 2021) or Sign Language Translation (SLT) (Liang et al., 2023). However, a significant challenge lies in the limited availability of annotated data for building sign language translators with strong generalization capabilities. Consequently, when alignment annotation are weak or absent, unsupervised techniques become essential for learning sign and word alignment. In this perspective, one approach in line with proposed experiments, we could investigate more the possibility of controlling attention distribution with the proposed recurrent aspect to infer sign localization.

Real-time sign language description. Towards real time sign language description, a lot of research was conducted on a preliminary experiments such as *Isolated Sign Language Recognition* (Laines et al., 2023), *Alignment and Segmentation* (Bull et al., 2021; Varol et al., 2021), *Transcription* (Saunders et al., 2020; Camgoz et al., 2020). These tasks work under a simplified hypothesis, which may not fully correspond to the real scenario of continuous signing. However, it is still required as initial steps toward the challenging of bidirectional mapping between sign and text or voice. These preliminary experiments could form the basis for *real time sign language description*, which represents a very challenging task driven by the computational efficiency required for real-time generation and furthermore the complexity of architecture design that could progressively perform translation of continuous signing to natural language.

In our context of using a 3D human dataset, we did achieve this progressive aligned generation but under the assumption of a monotonic action order, which may not fully hold for sign language order with respect to spoken language order. This makes it more challenging to work with compared to the other considered motion datasets.

Sign Language Production. This could be seen as the inverse task of sign language translation, where it can take more inspiration from architectures developed for motion generation conditioned on text or sequences of actions and could benefit more from the approaches developed for motion generation conditioned on language.

7.3 Conclusion

We have presented work on motion captioning through various perspectives. We considered this task as intermediate to solve other derived tasks: action localization, alignment and identification of body parts involved in an action. In this holistic approach, our experiments were centered on unsupervised learning of semantic motion segmentation (Chapter 4). This process requires specific metrics for evaluation, thus we proposed formulations to perform quantitative evaluation. Furthermore, we initiated the construction of a motion-language dataset to open the path for supervised learning of motion segmentation (Chapter 5). In contrast to black box models, we designed an interpretable architecture through spatio-temporal attention. The new model design ameliorates our previous results and provides access to a solution for the derived tasks. According to each aspect, we proposed effective tools for interpretability evaluation (Chapter 6). Later, we discussed transferability of our architecture components to adjacent tasks. Finally, we conducted a deep analysis of potential applications: unsupervised action segmentation and recognition, sign language translation and impact in other scenarios.

French synopsis

Notre société actuelle est caractérisée par une forte dynamique de mouvement, que ce soit en tant que composante essentielle de la santé humaine ou dans le cadre d'une multitude de disciplines sportives où chaque athlète aspire à l'excellence et à la performance. Le mouvement est ainsi au cœur de nombreuses études scientifiques. Les travaux présentés dans ce manuscrit ont été menés au sein d'une équipe de recherche nommée EuroMov Digital Health in Motion, qui s'est concentrée sur l'étude de la plasticité sensorimotrice humaine. Cette équipe est composée d'experts en sciences du mouvement, de médecins et d'autres professionnels de la santé, ainsi que d'experts en informatique et en intelligence artificielle. L'objectif principal d'Euro-Mov DHM est de comprendre tous les aspects de la plasticité sensorimotrice humaine pour améliorer la qualité de vie et la santé au fil du temps, ainsi que pour récupérer des fonctionnalités physiques altérées par des maladies ou des accidents de la vie. Cette recherche vise également à développer des innovations et à valoriser les secteurs où les sciences du mouvement, de la santé et de l'informatique se chevauchent, avec pour ambition finale de comprendre les empreintes sensorimotrices de la santé physique et mentale chez l'homme grâce à des approches de calcul innovantes.

Dans ce contexte, la représentation du mouvement pourra revêtir différentes formes et impliquer de nombreux paramètres.

Qu'est-ce qu'on entend par représentation du mouvement ?

La représentation du mouvement correspond à la traduction d'un mouvement physique en une représentation graphique ou numérique qui permet de décrire ses caractéristiques quantitatives et qualitatives. Cette représentation peut prendre différentes formes, allant des dessins et des schémas sur papier aux modèles 3D animés sur ordinateur. Elle peut également utiliser différents types de capteurs, tels que des capteurs de mouvement, des caméras ou des accéléromètres, pour mesurer et enregistrer les mouvements. Cette représentation permet d'avoir plusieurs objectifs, tels que l'analyse et la compréhension des mouvements humains pour les disciplines sportives, la rééducation physique, la prévention des blessures et l'amélioration des performances sportives.

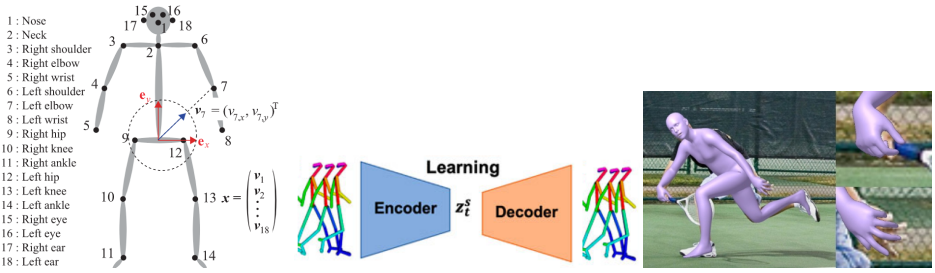
Quels sont les paramètres qui permettent de représenter le mouvement ? Il existe plusieurs méthodes pour représenter le mouvement. Une première méthode consiste à utiliser des capteurs déposés sur l'humain pour collecter des données cinématiques décrivant le mouvement, telles que la vitesse, l'accélération, l'angle. Le complément de ces données se focalise sur la description de l'objet en mouvement à travers sa position globale et encore la forme donnée par le squelette humain défini par le système Mocap. Les capteurs utilisés et ainsi les données nécessaires à collecter peuvent varier en fonction du type de mouvement étudié et du contexte d'utilisation.

Une deuxième méthode moins coûteuse repose sur l'utilisation de données vidéo pour l'estimation des paramètres du mouvement comme dans le cas de l'estimation de la pose 2D (Xu et al., 2022; Cao et al., 2019) et d'autres exemples de système d'estimation de la pose 3D comme détaillés par (Wang et al., 2021). Ces systèmes peuvent être mis en place pour la base de données étudiée et pour l'application choisie. À partir de cette estimation de positions spatiales, il est possible de dériver d'autres informations comme la vitesse et l'accélération.

Les représentations du mouvement peuvent également prendre la forme d'invariants calculés à partir des données de position des marqueurs. Les invariants sont définis par rapport à une mesure de référence, telle que le repère d'observation, et peuvent inclure des informations telles que les distances euclidiennes entre chaque paire de marqueurs, les vitesses, les coordonnées relatives et les angles. Ces invariants peuvent être définis manuellement comme proposé par (Takano and Lee, 2020), où des vecteurs sont calculés entre chaque paire de points d'articulations puis normalisés donnant ainsi des vecteurs unitaires invariants par rapport à la forme du corps et sa position globale (cf. Figure 7.2a).

Une autre solution consiste à apprendre automatiquement ces mesures invariantes. Plusieurs travaux (Yang et al., 2022; Li et al., 2020, 2018) ont proposé différentes architectures pour apprendre à produire des représentations invariantes par angle de vue. On note en particulier les travaux récents de (Yang et al., 2022), où les auteurs exploitent une technique de reciblage de mouvement qui permet de projeter les mouvements capturés sous différents angles de vue sur un plan commun, de manière à ce que les mouvements puissent être comparés de façon cohérente, quel que soit l'angle de vue. Ces représentations, incorporant des propriétées d'invariance, sont utiles pour réduire la variabilité entre les mouvements, par exemple représenter la même action, ce qui peut faciliter la reconnaissance d'action.

Cependant, ces représentations peuvent ne pas suffire dans certains cas tels que la reconnaissance de gestes complexes ou la prédiction de mouvements futurs. Ainsi des techniques avancées, comme proposé par (Bütepage et al., 2017; Zhu et al., 2023a) peuvent être mises en place pour apprendre des représentations plus robustes à partir de ces représentations de premier niveau. Observant la représentation squeletique de l'humain, son mouvement est perçu comme un graphe évoluant dans le temps, ce qui permet également de proposer des représetantions spatio-temporelles (Salih et al., 2016; Chi et al., 2022). Il est aussi possible d'exploiter une autre perspective du graphe du squelette en utilisant une modélisation par image en RGB (Pham et al., 2018), dont chaque pixel est défini par le triplet des coordonnées 3D de chaque point d'articulation après normalisation. La Figure 7.2 schématise des représentations simples (cf. Figure 7.2a) et avancées (cf. Figure 7.2b et 7.2c). Nous catégorisons sur la figure 7.3 les réprésentations les plus utilisées dans l'état de l'art pour la quantification du mouvement humain.



(a) Calcul manuel d'invariants.

(b) Apprentissage de représentation (espace latent).

(c) Représentation volumique en mesh.

FIGURE 7.2: Exemple de représentations du mouvement.

Pourquoi avons-nous besoin de représenter le mouvement ? La représentation du mouvement est essentielle pour de nombreuses applications, notamment la rééducation, la prévention des blessures et l'optimisation des performances sportives. En général, le processus d'analyse du mouvement humain fait appel une étape de quantification, strictement nécessaire pour l'analyse du mouvement par des méthodes utilisant l'apprentissage automatique. Ces dernières sont mises en oeuvre dans un grand nombre d'applications comme, par exemple, détecter des anomalies, surveiller la progression des patients en rééducation physique, faire de la recommandation pour l'amélioration de la performance sportifs et assurer le suivi des sportifs pour prévenir les blessures. Elles sont aussi utilisées en contexte de réalité virtuelle et en robotique pour développer des systèmes qui imitent les

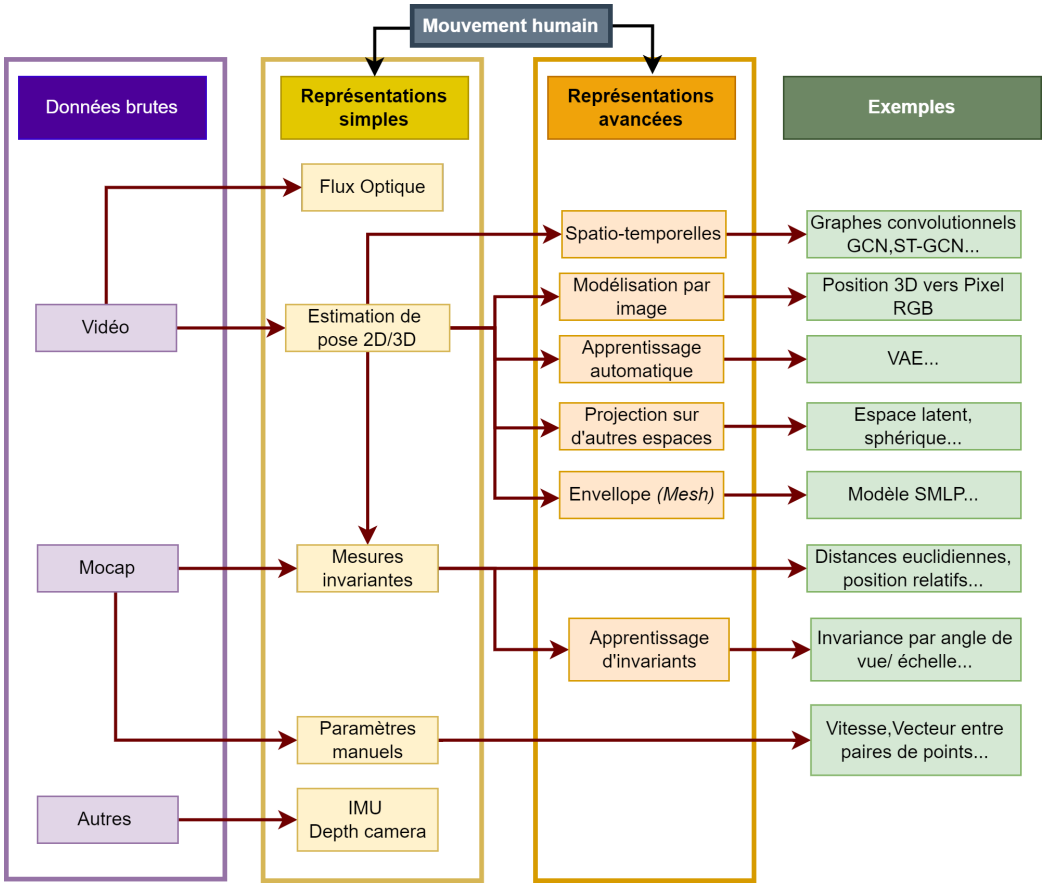


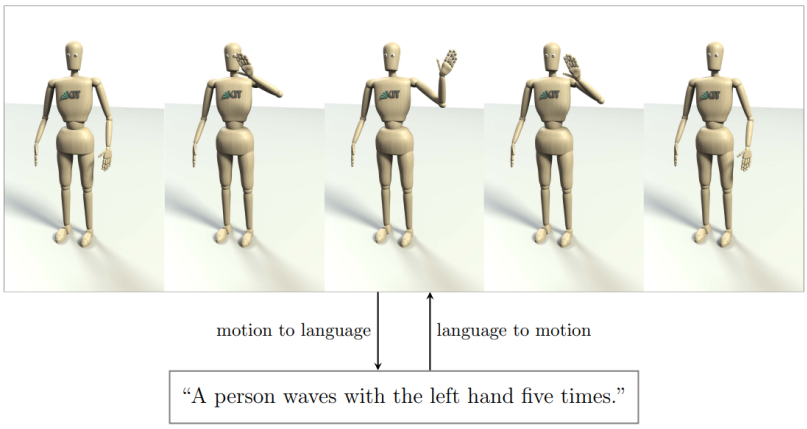
FIGURE 7.3: Taxonomie des représentations du mouvement.

mouvements humains. Dans ce contexte, l'utilisation de l'apprentissage automatique exploitent les coordonnées 2D ou/et 3D des articulations a été largement adoptée pour l'analyse du mouvement humain selon différents niveaux de granularité, allant de la reconnaissance simple d'action (Qin et al., 2022) jusqu'à la description détaillée du mouvement avec du langage naturel (Plappert et al., 2018). D'autres travaux proposent d'utiliser ces données intiales pour apprendre d'autres représentations plus robustes et profondes (Zhu et al., 2023a).

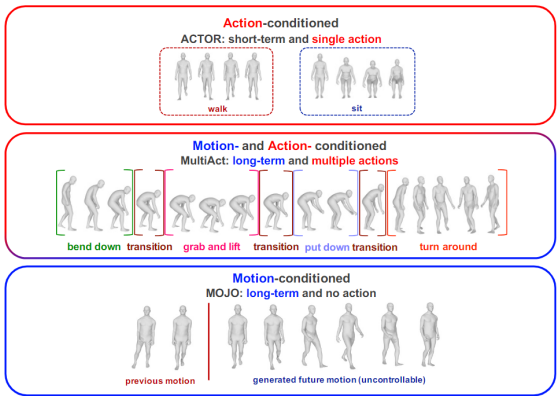
Où langage et mouvement se rencontrent La sémantisation du mouvement humain est au coeur de cette thèse. La littérature propose de nombreux exemples de travaux visant à ***générer du mouvement à partir de descriptions textuelles***. Un des premiers travaux dans ces champs disciplinaire a été proposé par (Plappert et al., 2018) qui étudie les deux *directions* de génération de mouvement et de texte (cf. Figure 7.4a) en se basant sur des modèles récurrents bidirectionnels. Plus récemment, les deux modes de génération ont été traités par (Toyoda

et al., 2022; Guo et al., 2022b). En se focalisant sur les travaux liés à la génération du mouvement, des techniques très diversiées ont été appliquées. (Ghosh et al., 2021) proposent la synthèse du mouvement (animation du texte) à partir d'un modèle basé sur des réseaux récurrents, spécifiquement des GRUs (gated recurrent units), avec un design hiéarchique qui encode les parties supérieures et inférieures du squelette humain. D'autres techniques plus avancées se basent sur l'utilisation de transformeurs (Vaswani et al., 2017). Par exemple, (Petrovich et al., 2022) encode le mouvement et le texte à travers un transformeur qui exploite le concept d'apprentissage du VAE (*Variationel AutoEncoder*) introduit par (Kingma and Welling, 2022). Ensuite, dans la phase d'inférence uniquement, l'encodeur du texte est utilisé pour la génération du mouvement par le biais d'un décodeur basé, lui aussi, sur un transformeur. (Guo et al., 2022a) proposent une architecture qui repose sur deux modules. *Text2length* permet d'échantillonner la longueur du mouvement et *Text2Motion*, un module architecturé par un VAE temporel, est utilisé pour la génération du mouvement. Pour l'encodage du mouvement ils introduisent une nouvelle représentation du mouvement interne nommée *motion snippet code*. Ainsi, tous ces modules mis en place, il est possible de générer des mouvements naturels et diverses. Plus récemment, (Zhang et al., 2023) ont proposé un design d'architecture incorporant un modèle génératif accompagné d'un modèle de type VQ-VAE (*Vector Quantised-Variational Autoencoder*) (van den Oord et al., 2017). Une autre technique plus sophistiquée a été utilisée par (Chen et al., 2023) qui se base sur un mécanisme de diffusion permettant de générer un mouvement conditionné par le texte ou l'action. Dans ces travaux, les chercheurs proposent d'apprendre une représentation interne du mouvement à travers le modèle VAE. Une autre forme de génération de mouvement à partir de texte est la génération conditionnée par une séquence d'actions continues (Lee et al., 2023) (cf. Figure 7.4b), ainsi que d'autres modes de génération du mouvement conditionnée par un texte dépendant de la scène étudiée (Wang et al., 2022) – cf. Figure 7.4c.

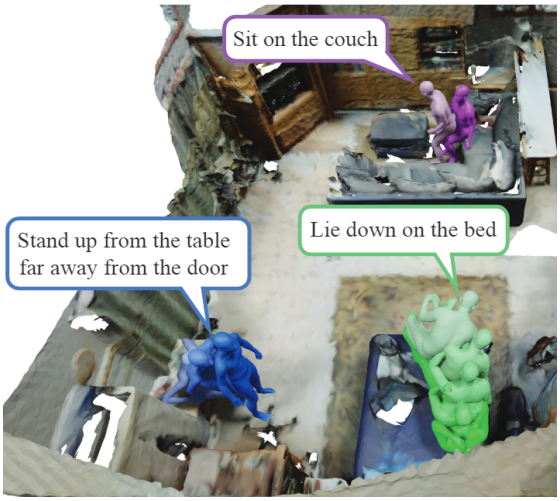
Si de nombreux travaux, on l'a vu, s'intéressent à la génération de mouvement à partir de texte, très peu de recherches s'intéressent à la génération inverse : ***produire une interprétation et une description sémantique à partir de la capture du mouvement***. Parmi les approches proposées dans la littérature, on note une large part des travaux de Takano, employant des méthodes qui, pour la plupart, s'articulent sur le modèle de markov caché (HMM) appliqué à des bases de données (BDD) de natures similaires (Takano et al., 2016; Takano and Lee, 2020; Takano et al., 2020). D'autres travaux restreints à la traduction mouvement vers



(a) Association mouvement-langage bidirectionnel (Plappert et al., 2018).



(b) Description comme succession d'actions (Lee et al., 2023).



(c) Description dépendant de la scène (Wang et al., 2022).

FIGURE 7.4: Trois différents contextes d'association langage-mouvement.

le texte ont été élaborés sur la base de données KIT-MLD, à la fois par les auteurs du jeu de données eux-mêmes (Plappert et al., 2018) et par d'autres équipes de recherche. Parmi ces dernières, on peut citer (Plappert et al., 2018) qui ont créé un système capable de générer des mouvements à partir d'une description textuelle et de générer du texte décrivant un mouvement en utilisant une architecture d'encodeur-décodeur GRU bidirectionnelle. Les auteurs évaluent leur système sur la génération de texte en utilisant la métrique BLEU-4 commune (Papineni et al., 2002), puis l'évaluent sur la génération de mouvement en faisant un aller-retour avec le mouvement généré pour générer à nouveau du texte qui peut alors être évalué par rapport à la description originale, en utilisant à nouveau la

métrique BLEU-4. La totalité des travaux d'association mouvement-langage traite cette tâche de manière globale et ne permet pas d'avoir une analyse sémantique fine du mouvement. C'est cette problématique qui sera abordée dans la suite de ce manuscrit: La génération du texte à travers des architectures interprétables en vue de l'analyse et la segmentation sémantique du mouvement humain.

> La thèse défendue dans ce contexte s'intéresse principalement à la génération du texte à travers des architectures interprétables en vue de l'analyse et la segmentation sémantique du mouvement humain. Le but est de permettre la résolution sans supervision des taches connexes: la synchronisation entre mouvement et texte, identification spatio-temporelle des éléments décrivant une primitive du mouvement humain et d'autres.

Pour répondre à cette problématique, plusieurs verrous doivent être surmontés.

Comment interpréter le mouvement ? Est-il possible d'expliquer le mouvement de façon automatisée comme le ferait un humain ?

La compréhension et l'interprétation du mouvement humain sont des défis importants pour les chercheurs. Les méthodes existantes pour interpréter le mouvement humain sont limitées, car elles ne permettent pas toujours une interprétation précise et complète du mouvement. Cela peut être dû à la complexité du mouvement humain et à la difficulté de traduire l'expérience subjective de l'interprétation du mouvement en un processus automatisé. Cette interprétation peut être graduée selon plusieurs niveaux de granularité, allant de la description globale du mouvement comme la reconnaissance d'action (*e.g.*, *sauter, marcher*), jusqu'à la description précise de chaque composante impliquée dans le mouvement : parties du corps humain impliquées dans le mouvement (*e.g.*, *pieds, jambe*), nature du mouvement (*e.g.*, *plier, poser, lever*) ainsi que manière de l'exécuter (*e.g.*, *vitesse*). La figure 7.5 donne un exemple de définition de granularité échelonnée sur trois niveaux : 0) reconnaissance du mouvement global ; 1) détection des parties impliquées dans l'exécution du mouvement ; 2) identification du mouvement relatif de chaque partie.

Comment comprendre et décomposer le mouvement au travers de ses décompositions temporelles et spatiales ?

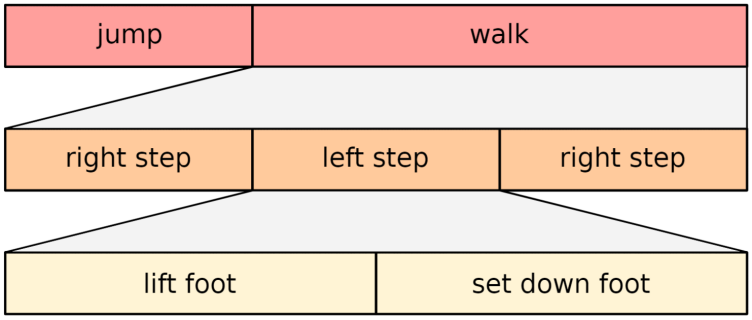


FIGURE 7.5: Exemple de niveaux de sémantisation d'un mouvement (Dreher et al., 2017).

La compréhension du mouvement humain nécessite une analyse en termes de ses composantes temporelles et spatiales. Les mouvements humains peuvent être décomposés en différentes parties, comme des segments corporels, des articulations, des mouvements linéaires ou circulaires, etc. La décomposition temporelle peut également inclure la durée du mouvement, la vitesse, l'accélération, etc. Pour comprendre et décomposer le mouvement humain, il est donc nécessaire d'abord de définir ce que considère une décomposition en lien avec l'analyse sémantique du mouvement. Les deux formes de décompositions peuvent être définies dans notre contexte comme suit :

Décomposition temporelle : Cette décomposition du mouvement sur l'axe temporel est assimilée à un processus de segmentation où chaque phase temporelle identifie un mouvement de base (primitive). L'ensemble des primitives compose alors le mouvement global.

Décompositon spatiale : Concernant la dimension spatiale elle consiste à identifier les parties impliquées dans la formation de chacun des mouvements de base et leur position dans l'espace.

Quelle architecture peut être proposée pour sémantiser le mouvement ?

La définition de l'architecture repose sur la définition de la sémantisation du mouvement. Dans ce manuscrit, la sémantisation du mouvement est définie comme le résultat de l'association des décompositions temporelle et spatiale appliquées au mouvement humain. Ainsi, on déduit que l'objectif de l'architecture est de permettre la représentation des différentes composantes du mouvement de manière à les rendre interprétables. Elle doit également permettre d'associer ces différentes composantes à des concepts sémantiques, comme des actions, etc. Plusieurs architectures sont en développement pour répondre à cette problématique, notamment des modèles basés sur l'apprentissage automatique et des approches inspirées du

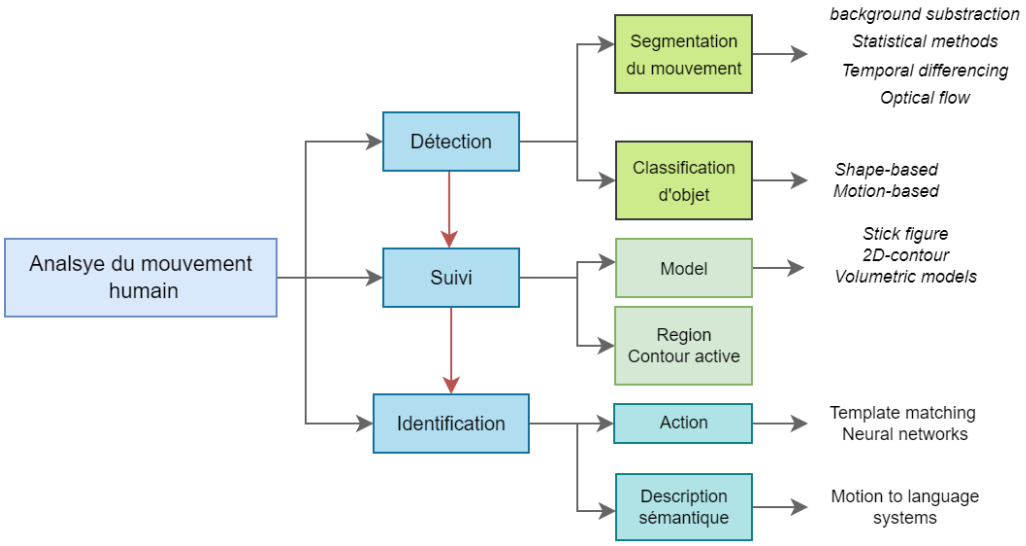


FIGURE 7.6: Taxonomie sur le processus d'analyse du mouvement humain et exemples d'approches classiques.

traitement du langage naturel. Dans le champ de l'analyse du mouvement humain (AMH), (Wang et al., 2003) proposent une revue approfondie de l'état de l'art qui recense les progrés établis dans l'AMH. Leur article présente les approches utilisées dans le processus de l'AMH qui implique la détection, puis le suivi et enfin l'identification de la nature du mouvement de la personne. Dans son sens large, l'AMH basée sur la vision adresse de la compréhension du comportement humain à partir d'une séquence d'images. La revue se focalise sur les contributions majeures dans les trois disciplines permettant d'appliquer l'AMH, ainsi que les applications qui en découlent pour la surveillance visuelle, les interfaces homme-machine et le diagnostic du mouvement. Dans la figure 7.6, on présente la taxonomie des méthodes d'AMH élaborée sur la base de cette revue.

D'autres revues plus récentes détaillent la progression dans l'AMH ainsi que la modélisation basée sur les sytèmes Mocap et les applications potentielles (Wang, 2016; Lim et al., 2015). Examinant ces états de l'art, on perçoit un manque dans les architectures permettant de répondre d'une manière efficace à la branche de description sémantique (cf. Figure 7.6), particulièrement l'association des deux décompositions temporelles et spatiales définies ci-dessus.

Pour répondre à ces différents enjeux, le manuscrit est organisé comme suit. Après une présentation du contexte et des objectifs de la thèse, le chapitre 2, consacré à l'estimation de pose, détaille les différentes techniques qui peuvent être utilisées pour quantifier et représenter le mouvement humain. L'objectif étant de

déterminer si les données estimées peuvent se substituer à la capture du mouvement à travers l'analyse des systèmes d'estimation de pose 2D et de reconstruction 3D. Le chapitre 3 fait une mise au point sur l'encodage du mouvement et les techniques de pointe qui peuvent être utilisées, avec une première contribution comme application d'une architecture *deep learning* dans le contexte de l'analyse du mouvement, spécifiquement pour la détection des comportements protectifs. Le chapitre 4 présente la deuxième contribution de cette thèse, qui consiste en une segmentation temporelle du mouvement, en utilisant des mécanismes d'attention. Le chapitre 5 constitue la troisième contribution, qui propose l'analyse quantitative et qualitative de la segmentation sémantique du mouvement humain inférée à partir des poids d'attention. Pour l'évaluation de la performance de cette segmentation on propose et on compare plusieurs métriques, puis, on détermine quelles sont les méthodes pertinentes pour la visualisation de la synchronisation entre mouvement et langage. Ensuite, on décrit notre construction de la première partie du jeu donnée *Euromov Motion Language Dataset* (EMLD). Le chapitre 6 détaille la quatrième contribution majeure, à savoir le développement d'une architecture interprétable basée sur des mécanismes guidés d'attention spatio-temporelle (Radouane et al., 2023b). Enfin, le chapitre 7 propose un résumé sur les contributions de la thèse, les applications potentielles des méthodologies proposées, ainsi que les perspectives d'amélioration et les retombées des travaux réalisés sur d'autres taches connexes.

Bibliography

Ahuja, C. and Morency, L.-P. (2019). Language2pose: Natural language grounded pose forecasting. *International Conference on 3D Vision (3DV)*. 112

Ali, A., Pinyoanuntapong, E., Wang, P., and Dorodchi, M. (2023). Skeleton-based human action recognition via convolutional neural networks (cnn). *ArXiv:2301.13360*. 56

Andriluka, M., Iqbal, U., Insafutdinov, E., Pishchulin, L., Milan, A., Gall, J., and Schiele, B. (2018). Posetrack: A benchmark for human pose estimation and tracking. In *Conference on Computer Vision and Pattern Recognition*, pages 5167–5176. 25, 27

Andriluka, M., Pishchulin, L., Gehler, P., and Schiele, B. (2014). 2d human pose estimation: New benchmark and state of the art analysis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 25, 27

Athanasiou, N., Petrovich, M., Black, M. J., and Varol, G. (2022). TEACH: Temporal action composition for 3D humans. In *International Conference on 3D Vision (3DV)*. 196

Athanasiou, N., Petrovich, M., Black, M. J., and Varol, G. (2023). SINC: Spatial composition of 3D human motions for simultaneous action generation. In *International Conference on Computer Vision (ICCV)*. 196

Aung, M. S. H., Kaltwang, S., Romera-Paredes, B., Martinez, B., Singh, A., Cella, M., Valstar, M., Meng, H., Kemp, A., Shafizadeh, M., Elkins, A. C., Kanakam, N., de Rothschild, A., Tyler, N., Watson, P. J., Williams, A. C. d. C., Pantic, M., and Bianchi-Berthouze, N. (2016). The automatic detection of chronic pain-related expression: Requirements, challenges and the multimodal emopain dataset. *IEEE Transactions on Affective Computing*, 7(4):435–451. 65

Badiola-Bengoa, A. and Mendez-Zorrilla, A. (2021). A systematic review of the application of camera-based human pose estimation in the field of sport and physical exercise. *Sensors*. 24

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*. 107, 108, 110, 111, 116, 164

Bai, B., Liang, J., Zhang, G., Li, H., Bai, K., and Wang, F. (2021). Why attentions may not be interpretable? In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 79

B'en'edict, G., Koops, V., Odijk, D., and de Rijke, M. (2021). sigmoidf1: A smooth f1 score surrogate loss for multilabel classification. *Transactions on Machine Learning Research*. 87

Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com. 174

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. 159

Bull, H., Afouras, T., Varol, G., Albanie, S., Momeni, L., and Zisserman, A. (2021). Aligning subtitles in sign language videos. In *International Conference on Computer Vision (ICCV)*. 131, 140, 197

Bull, H., Gouiffès, M., and Braffort, A. (2020). Automatic segmentation of sign language into subtitle-units. In *ECCV 2020 Workshops*. 131

Bütepage, J., Black, M. J., Kragic, D., and Kjellström, H. (2017). Deep representation learning for human motion prediction and classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1591–1599. 201

Camgoz, N. C., Koller, O., Hadfield, S., and Bowden, R. (2020). Sign language transformers: Joint end-to-end sign language recognition and translation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 197

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *Transactions on Pattern Analysis and Machine Intelligence*. 13, 32, 46, 200

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Neural Information Processing Systems*. 95

Chen, X., Jiang, B., Liu, W., Huang, Z., Fu, B., Chen, T., and Yu, G. (2023). Executing your commands via motion diffusion in latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14, 93, 102, 203

Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., and Sun, J. (2018). Cascaded Pyramid Network for Multi-Person Pose Estimation. In *Computer Vision and Pattern Recognition (CVPR)*. 29, 34, 46

Cheng, Y., Yang, B., Wang, B., and Tan, R. T. (2020). 3D Human Pose Estimation using Spatio-Temporal Networks with Explicit Occlusion Training. In *Association for the Advancement of Artificial Intelligence (AAAI)*. 35

Chi, H.-G., Ha, M. H., Chi, S., Lee, S. W., Huang, Q., and Ramani, K. (2022). Infogcn: Representation learning for human skeleton-based action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20154–20164. 13, 201

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (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)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics. 107

Chun, S., Park, S., and Chang, J. Y. (2023). Learnable human mesh triangulation for 3d human pose and shape estimation. *Winter Conference on Applications of Computer Vision (WACV)*, pages 2849–2858. 41

Cui, Y., Jia, M., Lin, T., Song, Y., and Belongie, S. (2019). Class-balanced loss based on effective number of samples. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9260–9269, Los Alamitos, CA, USA. IEEE Computer Society. 72

Dreher, C. R. G., Kulp, N., Mandery, C., Wächter, M., and Asfour, T. (2017). A framework for evaluating motion segmentation algorithms. *International Conference on Humanoid Robotics*, pages 83–90. 17, 143, 206

Filtjens, B., Vanrumste, B., and Slaets, P. (2022). Skeleton-based action segmentation with multi-stage spatial-temporal graph convolutional neural networks. *IEEE Transactions on Emerging Topics in Computing*, pages 1–11. 144

Fu, K., Jin, J., Cui, R., Sha, F., and Zhang, C. (2017). Aligning where to see and what to tell: Image captioning with region-based attention and scene-specific contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2321–2334. 163

Geng, Z., Sun, K., Xiao, B., Zhang, Z., and Wang, J. (2021). Bottom-Up Human Pose Estimation Via Disentangled Keypoint Regression. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14671–14681. 26, 33, 46

Ghorbani, N. and Black, M. J. (2021). SOMA: Solving optical marker-based mocap automatically. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*. 159

Ghosh, A., Cheema, N., Oguz, C., Theobalt, C., and Slusallek, P. (2021). Synthesis of compositional animations from textual descriptions. *International Conference on Computer Vision (ICCV)*. 14, 93, 101, 203

Goutsu, Y. and Inamura, T. (2021). Linguistic descriptions of human motion with generative adversarial seq2seq learning. In *International Conference on Robotics and Automation (ICRA)*. 93, 125, 176

Gu, Y., Pandit, S., Saraee, E., Nordahl, T., Ellis, T., and Betke, M. (2019). Home-based physical therapy with an interactive computer vision system. *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pages 2619–2628. 24

Guo, C., Zou, S., Zuo, X., Wang, S., Ji, W., Li, X., and Cheng, L. (2022a). Generating diverse and natural 3d human motions from text. In *Computer Vision and Pattern Recognition (CVPR)*, pages 5152–5161. 12, 14, 93, 94, 98, 99, 100, 101, 102, 103, 133, 134, 155, 203

Guo, C., Zuo, X., Wang, S., and Cheng, L. (2022b). Tm2t: Stochastic and tokenized modeling for the reciprocal generation of 3d human motions and texts. In *ECCV*, page 580–597. 12, 13, 14, 93, 97, 98, 101, 102, 103, 133, 135, 175, 176, 203

Guo, C., Zuo, X., Wang, S., Zou, S., Sun, Q., Deng, A., Gong, M., and Cheng, L. (2020). Action2motion: Conditioned generation of 3d human motions. *Proceedings of the 28th ACM International Conference on Multimedia*. 196

Guo, L., Liu, J., Tang, J., Li, J., Luo, W., and Lu, H. (2019). Aligning linguistic words and visual semantic units for image captioning. *MM 2019 - Proceedings of the 27th ACM International Conference on Multimedia*, pages 765–773. 165

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask r-cnn. In *International Conference on Computer Vision (ICCV)*, pages 2980–2988. 29, 30, 46

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. (2017). Learning discrete representations via information maximizing self-augmented training. In *International Conference on Machine Learning*. 95

Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014a). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339. 25

Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014b). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339. 27, 38, 44

Iskakov, K., Burkov, E., Lempitsky, V., and Malkov, Y. (2019). Learnable triangulation of human pose. In *International Conference on Computer Vision (ICCV)*. 34, 39, 41, 46, 47

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*. 95

Jiang, B., Chen, X., Liu, W., Yu, J., Yu, G., and Chen, T. (2024). Motiongpt: Human motion as a foreign language. *Neural Information Processing Systems*. 99, 103

Johnson, S. and Everingham, M. (2010). Clustered pose and nonlinear appearance models for human pose estimation. In *British Machine Vision Conference (BMVC)*, volume 2, page 5. 25, 27

Kaid, A. E., Brazey, D., Barra, V., and Baïna, K. (2022). Top-down system for multi-person 3d absolute pose estimation from monocular videos. *Sensors*, 22. 47

Kingma, D. P. and Welling, M. (2022). Auto-encoding variational bayes. In *arXiv:1312.6114*. 14, 101, 203

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907. 60, 65

Kulić, D., Takano, W., and Nakamura, Y. (2009). Online segmentation and clustering from continuous observation of whole body motions. *IEEE Transactions on Robotics*, 25. 143

Laines, D., Gonzalez-Mendoza, M., Ochoa-Ruiz, G., and Bejarano, G. (2023). Isolated sign language recognition based on tree structure skeleton images. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*. 197

Lee, T., Moon, G., and Lee, K. M. (2023). Multiact: Long-term 3d human motion generation from multiple action labels. In *AAAI*. 14, 15, 203, 204

Li, J., Wong, Y., Zhao, Q., and Kankanhalli, M. S. (2018). Unsupervised learning of view-invariant action representations. *Computer Vision and Pattern Recognition*. 13, 143, 200

Li, Y., Xia, R., and Liu, X. (2020). Learning shape and motion representations for view invariant skeleton-based action recognition. *Pattern Recognition*, 103:107293. 13, 200

Liang, Z., Li, H., and Chai, J. (2023). Sign language translation: A survey of approaches and techniques. *Electronics*, 12(12). 197

Lim, C. H., Vats, E., and Chan, C. S. (2015). Fuzzy human motion analysis: A review. *Pattern Recognition*, 48(5):1773–1796. 207

Lin, J. F.-S., Karg, M., and Kulić, D. (2016). Movement primitive segmentation for human motion modeling: A framework for analysis. *IEEE Transactions on Human-Machine Systems*, 46(3):325–339. 145

Lin, J. F.-S. and Kulic, D. (2012). Segmenting human motion for automated rehabilitation exercise analysis. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. 143

Lin, J. F. S. and Kulic, D. (2014). Online segmentation of human motion for automated rehabilitation exercise analysis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22. 143

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2014). Microsoft coco: Common objects in context. 25, 26, 32

Lipton, Z. C., Elkan, C., and Narayanaswamy, B. (2014). Optimal thresholding of classifiers to maximize f1 measure. In *Machine Learning and Knowledge Discovery in Databases*. 77

Liu, C., Mao, J., Sha, F., and Yuille, A. (2017). Attention correctness in neural image captioning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4176–4182. AAAI Press. 165

Liu, J., Rojas, J., Liang, Z., Li, Y., and Guan, Y. (2021). A graph attention spatio-temporal convolutional network for 3d human pose estimation in video. In *International Conference on Robotics and Automation (ICRA)*. 47

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. (2015). SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16. 47

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. In *International Conference on Learning Representations*. 174

Lu, J., Xiong, C., Parikh, D., and Socher, R. (2017). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3242–3250. 165

Lu, M., Poston, K., Pfefferbaum, A., Sullivan, E., Fei-Fei, L., Pohl, K., Niebles, J. C., and Adeli, E. (2020). Vision-based estimation of mds-updrs gait scores for assessing parkinson's disease motor severity. 12263:637–647. 24, 59

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 107, 109, 110, 116, 164

Ma, H., Yang, Z., and Liu, H. (2021). Fine-grained unsupervised temporal action segmentation and distributed representation for skeleton-based human motion analysis. *IEEE Transactions on Cybernetics*. 144

Markovitz, A., Sharir, G., Friedman, I., Zelnik-Manor, L., and Avidan, S. (2020). Graph embedded pose clustering for anomaly detection. *Proceedings of the IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition, pages 10536–10544. 24

Mehta, D., Sotnychenko, O., Mueller, F., Xu, W., Sridhar, S., Pons-Moll, G., and Theobalt, C. (2018). Single-shot multi-person 3d pose estimation from monocular rgb. In *International Conference on 3D Vision (3DV)*, pages 120–130. 47, 48

Mei, F., Hu, Q., Yang, C., and Liu, L. (2021). Arma-based segmentation of human limb motion sequences. *Sensors*, 21(16). 143

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. volume 2, pages 1045–1048. 107

Momeni, L., Varol, G., Albanie, S., Afouras, T., and Zisserman, A. (2020). Watch, read and lookup: learning to spot signs from multiple supervisors. In *ACCV*. 140

Moon, G., Chang, J. Y., and Lee, K. M. (2017). Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image. In *ICCV*. 47

Moon, G., Chang, J. Y., and Lee, K. M. (2018). V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. *Computer Vision and Pattern Recognition*. 41

Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*. 34

Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*. 63

Odena, A. (2016). Semi-supervised learning with generative adversarial networks. In *Workshop on Data-Efficient Machine Learning (ICML 2016)*. 95

Ofli, F., Chaudhry, R., Kurillo, G., Vidal, R., and Bajcsy, R. (2013). Berkeley mhad: A comprehensive multimodal human action database. In *Workshop on Applications of Computer Vision (WACV)*, pages 53–60. 49, 50, 51

Olugbade, T., Sagoleo, R., Ghisio, S., Gold, N., De C Williams, A. C., De Gelder, B., Camurri, A., Volpe, G., and Bianchi-Berthouze, N. (2021). The affectmove 2021

challenge - affect recognition from naturalistic movement data. In *9th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 01–05. 16, 56, 57, 65, 87

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics. 204

Pavllo, D., Feichtenhofer, C., Grangier, D., and Auli, M. (2019). 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Computer Vision and Pattern Recognition (CVPR)*. 34, 35, 46, 50

Petrovich, M., Black, M. J., and Varol, G. (2021). Action-conditioned 3D human motion synthesis with transformer VAE. In *International Conference on Computer Vision (ICCV)*. 196

Petrovich, M., Black, M. J., and Varol, G. (2022). TEMOS: Generating diverse human motions from textual descriptions. In *European Conference on Computer Vision (ECCV)*. 14, 93, 101, 102, 196, 203

Petrovich, M., Black, M. J., and Varol, G. (2023). TMR: Text-to-motion retrieval using contrastive 3D human motion synthesis. In *International Conference on Computer Vision (ICCV)*. 140, 141

Pham, H.-H., Khoudour, L., Crouzil, A., Zegers, P., and Velastin, S. A. (2018). Learning to recognise 3d human action from a new skeleton-based representation using deep convolutional neural networks. 13:319–328. 201

Plappert, M., Mandery, C., and Asfour, T. (2016). The KIT motion-language dataset. *Big Data*, 4(4):236–252. 12, 93, 94, 112, 124, 154

Plappert, M., Mandery, C., and Asfour, T. (2018). Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks. *Robotics and Autonomous Systems*. 12, 13, 14, 15, 93, 95, 100, 112, 125, 133, 202, 204

Punnakkal, A. R., Chandrasekaran, A., Athanasiou, N., Quiros-Ramirez, A., and Black, M. J. (2021). BABEL: Bodies, action and behavior with english labels. In *Computer Vision and Pattern Recognition (CVPR)*, pages 722–731. 141

Qin, Z., Liu, Y., Perera, M., Gedeon, T., Ji, P., Kim, D., and Anwar, S. (2022). Anubis: Skeleton action recognition dataset, review, and benchmark. *Computer Vision and Pattern Recognition*. 13, 202

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. 102

Radouane, K., Tchechmedjiev, A., Lagarde, J., and Ranwez, S. (2023a). Motion2language, unsupervised learning of synchronized semantic motion segmentation. *Neural Computing and Applications*. 18, 92, 122, 124, 125, 135, 140, 147, 149, 150

Radouane, K., Tchechmedjiev, A., Ranwez, S., and Lagarde, J. (2023b). Guided attention for interpretable motion captioning. *arXiv:2310.07324*. 18, 162, 167, 169, 176, 208

Radouane, K., Tchechmedjiev, A., Xu, B., and Harispe, S. (2021). Comparison of deep learning approaches for protective behaviour detection under class imbalance from mocap and emg data. In *9th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 01–08. 18, 56, 72, 74

Rastgoo, R., Kiani, K., Escalera, S., and Sabokrou, M. (2021). Sign language production: A review. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*. 197

Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Neural Information Processing Systems*. 97

Reddy, N. D., Guigues, L., Pishchulin, L., Eledath, J., and Narasimhan, S. G. (2021). Tessetrack: End-to-end learnable multi-person articulated 3d pose tracking. In *CVPR*. 41, 46

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Empirical Methods in Natural Language Processing (EMNLP)*. 119

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. 29, 30, 46

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101. 78

Salih, A., Alwani, A. L., and Chahir, Y. (2016). Spatiotemporal representation of 3d skeleton joints-based action recognition using modified spherical harmonics. *Pattern Recognition Letters*. 13, 201

Saunders, B., Camgoz, N. C., and Bowden, R. (2020). Progressive Transformers for End-to-End Sign Language Production. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 197

Shan, W., Liu, Z., Zhang, X., Wang, Z., Han, K., Wang, S., Ma, S., and Gao, W. (2023). Diffusion-based 3d human pose estimation with multi-hypothesis aggregation. *arXiv preprint arXiv:2303.11579*. 36, 46

Song, J., Gao, L., Guo, Z., Liu, W., Zhang, D., and Shen, H. T. (2017). Hierarchical lstm with adjusted temporal attention for video captioning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 2737–2743. AAAI Press. 165, 166, 168, 171

Song, Y., Sárosi, J., Cen, X., and Bíró, I. (2023). Human motion analysis and measurement techniques: current application and developing trend. *Analecta Technica Szegedinensia*. 13

Song, Y.-F., Zhang, Z., Shan, C., and Wang, L. (2020). Stronger, faster and more explainable: A graph convolutional baseline for skeleton-based action recognition. In *Proceedings of the 28th ACM International Conference on Multimedia*. 56, 57, 59, 68, 70, 71, 72, 73

Stefanini, M., Cornia, M., Baraldi, L., Cascianelli, S., Fiameni, G., and Cucchiara, R. (2023). From show to tell: A survey on deep learning-based image captioning. *Transactions on Pattern Analysis and Machine Intelligence*, (1). 163, 164

Sun, J., Wang, M., Zhao, X., and Zhang, D. (2020). Multi-View Pose Generator Based on Deep Learning for Monocular 3D Human Pose Estimation. *Symmetry*, page 1116. 36, 46

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*. 108

Takano, W., Kusajima, I., and Nakamura, Y. (2016). Generating action descriptions from statistically integrated representations of human motions and sentences. *Neural Networks*. 14, 93, 203

Takano, W. and Lee, H. (2020). Action description from 2d human postures in care facilities. *IEEE Robotics and Automation Letters*. 13, 14, 93, 95, 200, 203

Takano, W., Murakami, Y., and Nakamura, Y. (2020). Representation and classification of whole-body motion integrated with finger motion. *Robotics and Autonomous Systems*. 14, 95, 203

Takano, W., Yamada, Y., and Nakamura, Y. (2019). Linking human motions and objects to language for synthesizing action sentences. *Autonomous Robots*. 93

Tiwary, T. and Mahapatra, R. P. (2023). An accurate generation of image captions for blind people using extended convolutional atom neural network. *Multimedia Tools and Applications*, 82:3801–3830. 162

Tompson, J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. In *Neural Information Processing Systems*. 34

Toyoda, M., Suzuki, K., Hayashi, Y., and Ogata, T. (2022). Learning bidirectional translation between descriptions and actions with small paired data. *IEEE Robotics and Automation Letters*, 7(4):10930–10937. 14, 93, 101, 202

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. In *International Conference on Neural Information Processing Systems*. 14, 92, 95, 96, 203

Varol, G., Momeni, L., Albanie, S., Afouras, T., and Zisserman, A. (2021). Read and attend: Temporal localisation in sign language videos. In *Computer Vision and Pattern Recognition Conference (CVPR)*. 140, 141, 197

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Neural Information Processing Systems*. 14, 31, 36, 92, 94, 99, 103, 105, 108, 111, 133, 164, 203

Wang, C., Wang, Y., Lin, Z., Yuille, A. L., and Gao, W. (2014). Robust estimation of 3d human poses from a single image. *Computer Vision and Pattern Recognition(CVPR)*, pages 2369–2376. 35

Wang, J., Tan, S., Zhen, X., Xu, S., Zheng, F., He, Z., and Shao, L. (2021). Deep 3d human pose estimation: A review. *Computer Vision and Image Understanding*, 210. 13, 200

Wang, L., Hu, W., and Tan, T. (2003). Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585–601. 207

Wang, Q. (2016). A survey of visual analysis of human motion and its applications. In *International Conference on Visual Communications and Image Processing (VCIP)*. 207

Wang, Z., Chen, Y., Liu, T., Zhu, Y., Liang, W., and Huang, S. (2022). Humanise: Language-conditioned human motion generation in 3d scenes. In *Advances in Neural Information Processing Systems (NeurIPS)*. 14, 15, 203, 204

Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. In *Computer Vision and Pattern Recognition (CVPR)*. 31

Wu, W., Luo, H., Fang, B., Wang, J., and Ouyang, W. (2023). Cap4video: What can auxiliary captions do for text-video retrieval? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 162

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2019). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24. 61

Xu, Y., Zhang, J., Zhang, Q., and Tao, D. (2022). Vitpose: Simple vision transformer baselines for human pose estimation. In *NeurIPS*. 13, 32, 46, 200

Yamada, T., Matsunaga, H., and Ogata, T. (2018). Paired recurrent autoencoders for bidirectional translation between robot actions and linguistic descriptions. *Robotics and Automation Letters*. 176

Yan, S., Xiong, Y., and Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Association for the Advancement of Artificial Intelligence (AAAI)*. 56, 57, 62, 63, 64

Yan, W., Zhang, Y., Abbeel, P., and Srinivas, A. (2021). Videogpt: Video generation using vq-vae and transformers. *ArXiv:2104.10157*. 97

Yang, D., Wang, Y., Dantcheva, A., Garattoni, L., Francesca, G., and Brémond, F. (2022). Via: View-invariant skeleton action representation learning via motion retargeting. 13, 200

Yang, F., Sakti, S., Wu, Y., and Nakamura, S. (2019). Make skeleton-based action recognition model smaller, faster and better. *Proceedings of the ACM Multimedia Asia*. 57, 58

Zeng, A., Sun, X., Huang, F., Liu, M., Xu, Q., and Lin, S. C.-F. (2020). Srnet: Improving generalization in 3d human pose estimation with a split-and-recombine approach. In *ECCV*. 41, 42, 46

Zhang, J., Tu, Z., Yang, J., Chen, Y., and Yuan, J. (2022). Mixste: Seq2seq mixed spatio-temporal encoder for 3d human pose estimation in video. *Computer Vision and Pattern Recognition*. 36, 37, 46

Zhang, J., Zhang, Y., Cun, X., Huang, S., Zhang, Y., Zhao, H., Lu, H., and Shen, X. (2023). T2m-gpt: Generating human motion from textual descriptions with discrete representations. In *Computer Vision and Pattern Recognition (CVPR)*. 14, 93, 98, 101, 102, 103, 203

Zhou, B., Khosla, A., A., L., Oliva, A., and Torralba, A. (2016). Learning Deep Features for Discriminative Localization. In *Computer Vision and Pattern Recognition (CVPR)*. 82

Zhou, F., De la Torre, F., and Hodgins, J. K. (2008). Aligned cluster analysis for temporal segmentation of human motion. In *International Conference on Automatic Face and Gesture Recognition*. 144, 160

Zhou, F., De la Torre, F., and Hodgins, J. K. (2013). Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):582–596. 144, 160

Zhu, W., Ma, X., Liu, Z., Liu, L., Wu, W., and Wang, Y. (2023a). Motionbert: A unified perspective on learning human motion representations. In *International Conference on Computer Vision*. 13, 37, 39, 46, 47, 201, 202

Zhu, W., Ma, X., Ro, D., Ci, H., Zhang, J., Shi, J., Gao, F., Tian, Q., and Wang, Y. (2023b). Human motion generation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20. 12

Óscar G Hernández, Morell, V., Ramon, J. L., Jara, C. A., Nuovo, A. D., and Armada, M. (2021). Human pose detection for robotic-assisted and rehabilitation environments. *Applied Sciences*, 11(9). 24

Abstract

Captioning tasks mainly focus on images or videos, and seldom on human poses. Yet, poses concisely describe human activities. Beyond text generation quality, we consider the motion caption task as an intermediate step to solve other derived tasks. In this holistic approach, our experiments are centered on the unsupervised learning of semantic motion segmentation and interpretability. We first conduct an extensive literature review of recent methods for human pose estimation, as a central prerequisite for pose-based captioning. Then, we take an interest in pose-representation learning, with an emphasis on the use of spatiotemporal graph-based learning, which we apply and evaluate on a real-world application (protective behavior detection). As a result, we win the AffectMove challenge. Next, we delve into the core of our contributions in motion captioning, where: (i) We design local recurrent attention for synchronous text generation with motion. Each motion and its caption are decomposed into primitives and corresponding sub-captions. We also propose specific metrics to evaluate the synchronous mapping between motion and language segments. (ii) We initiate the construction of a motion-language dataset to enable supervised segmentation. (iii) We design an interpretable architecture with a transparent reasoning process through spatiotemporal attention, showing state-of-the-art results on the two reference datasets, KIT-ML and HumanML3D. Effective tools are proposed for interpretability evaluation and illustration. Finally, we conduct a thorough analysis of potential applications: unsupervised action segmentation, sign language translation, and impact in other scenarios.

Résumé

Dans l'état de l'art, les tâches de sous-titrage se concentrent souvent sur les images et les vidéos, mais rarement sur les poses humaines. Ces dernières offrent pourtant une représentation concise des activités humaines et, au-delà de la qualité de la génération de texte, la tâche de "légendage" de mouvement peut constituer un intermédiaire pour résoudre d'autres tâches dérivées. Les travaux présentés dans ce manuscrit sont centrés sur l'apprentissage non supervisé qui peut être utilisé pour la segmentation de mouvement et l'identification d'une sémantique associée, ainsi que son interprétabilité. Après une revue de la littérature des méthodes récentes pour l'estimation de poses humaines, un prérequis central pour le légendage basé sur la pose, nous nous intéressons à l'apprentissage de la représentation de pose, avec un accent sur la modélisation basée sur des graphes spatio-temporels. Notre modèle est évalué sur une application réelle de détection de comportement protecteur, pour laquelle nous avons gagné le défi AffectMove. Les contributions majeures concernant le légendage du mouvement sont ensuite détaillées en trois temps. (i) Un mécanisme d'attention récurrent local pour la génération de texte synchronisé avec le mouvement est proposé, où chaque mouvement et sa légende sont décomposés en primitives et sous-légendes correspondantes. Des métriques spécifiques sont proposées pour évaluer la correspondance entre les segments de mouvement et les segments de langage. (ii) Un jeu de données mouvement-langage est ensuite proposé pour permettre une segmentation supervisée. (iii) Enfin, une architecture interprétable avec un processus de raisonnement transparent à travers l'attention spatio-temporelle est proposée. Cette architecture montre des "résultats état-de-l'art" sur les deux jeux de données de référence, KIT-ML et HumanML3D. Des outils efficaces sont proposés pour l'évaluation et l'illustration de l'interprétabilité. Ces contributions ouvrent de nombreuses perspectives de recherche et le manuscrit se termine par une analyse approfondie des applications potentielles : la segmentation d'actions non supervisée, la traduction automatique de la langue des signes ou encore l'impact dans d'autres scénarios.