UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Danilo Barros Cardoso

Action recognition approaches with context and multi-scale motion awareness

Belo Horizonte
2022

Danilo Barros Cardoso

Action recognition approaches with context and multi-scale motion awareness

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Erickson Rangel do Nascimento

Belo Horizonte
2022

Cardoso,Danilo Barros.

C268c Action recognition approaches with context and multi-scale motion awareness [manuscrito] / Danilo Barros Cardoso — 2022.
65 f. il.; 29 cm.

Orientador: Erickson Rangel do Nascimento.
Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação
Referências: f. 60-65

1. Computação – Teses. 2. Reconhecimento de ações humanas –Teses. 3. Análise de movimento humano – Teses. I. Nascimento, Erickson Rangel do II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. III. Título.

CDU 519.6*82.10 (043)

Ficha Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa CRB 6/1510 Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

ACTION RECOGNITION APPROACHES WITH CONTEXT AND MULTI-SCALE MOTION AWARENESS

DANILO BARROS CARDOSO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Erickson Rangel do Nascimento - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Mario Fernando Montenegro Campos
Departamento de Ciência da Computação - UFMG

Prof. Renato José Martins
ImViA - Université de Bourgogne

Belo Horizonte, 29 de julho de 2022.

Acknowledgments

Agradeço a todos que, de perto ou de longe, contribuíram para que eu fosse capaz de trabalhar por este objetivo que é me tornar mestre. São efetivamente muitas pessoas entre familiares, amigos e colegas de estudo. Gostaria de agradecer especialmente a minha namorada, Ana Carolina, pela paciência ao longo deste período, pois foram muitas oportunidades de atividades juntos que tivemos que abrir mão para me dedicar aos estudos e às pesquisas. À minha mãe, não por esses últimos meses em que dediquei ao mestrado, mas por uma vida inteira de oportunidades que pude aproveitar em consequência do seu esforço e sacrifício que me permitiram ter uma boa educação. À FAPEMIG e CNPq por seguir firme no desafiador trabalho que é fomentar pesquisa, e que permite ao VeRLab possuir equipamentos para a realização de seus trabalhos. Aos colegas do VeRLab pelo companheirismo e disponibilidade em ajudar, especialmente aqueles que contribuem para manter ativa a invejável estrutura do laboratório. E por fim ao Professor Erickson pela dedicada orientação que acabou por tornar a experiência acadêmica muito mais enriquecedora do que o esperado.

“*Civilization is the limitless multiplication of unnecessary necessities.*”

(Mark Twain)

Resumo

Embora tenhamos testemunhado um progresso substancial feito por abordagens de visão computacional na solução de problemas de classificação de imagens, detecção de objetos e estimativa de pose, para citar alguns, o reconhecimento de ação continua sendo um dos seus principais desafios em visão computacional e reconhecimento de padrões. Um método abrangente deve lidar com uma série de desafios, como ruidos no plano de fundo, oclusões, variações de escala, iluminação e aspecto. Além disso, quando consideramos métodos baseados em aprendizagem de máquina, a construção de conjuntos de dados tende a ser cara e complexa, incentivando o aproveitamento de sequências capturadas em situações naturais que trazem, por sí mesmas, novos desafios como o desbalanceamento entre as atividades observadas e ambiguidade na classificação. Esta dissertação propõe uma estrutura de aprendizagem para endereçar o problema de reconhecimento de atividades quando exposta a dois destes desafios: desbalanceamento e ambiguidade. Nossa abordagem utiliza para análise de poses de agentes uma arquitetura que combina camadas de convolução em grafos acrescida de um mecanismo para captura de características multiescala espaço-temporais e camadas de Transformers para captura de contexto. Embora diversos métodos da literatura tenham alcançado elevados níveis de precisão quando testados em conjuntos de dados de referência como NTU, seu desempenho diminui significativamente quando testados em um conjunto de dados com alto grau de ambiguidade entre as atividades e um número desequilibrado de amostras para cada classe. Avaliamos nossa arquitetura no desafiador conjunto de dados BABEL, onde alcançamos o estado da arte em termos de precisão (65,4%) na classificação de ações em métrica que considera tanto a ambiguidade quanto o desequilíbrio na representação entre classes. Além disso, por meio da observação dos perfis de ativação obtidos por diferentes modelos, realizamos uma análise qualitativa de como aspectos da nossa abordagem contribuiram para o resultado obtido.

Palavras-chave: reconhecimento de ações, representação de ação, análise de movimento multiescala, análise de movimentos baseado em esqueletos.

Abstract

Although computer vision approaches have provided remarkable advances in solving image classification, object detection, and pose estimation, to name a few, activity recognition still remains one of the key challenges. A comprehensive method has to deal with several challenges such as background noise, occlusions, variations in scale, lighting, and aspect. Furthermore, when we consider learning-based methods, the construction of datasets tends to be expensive and complex, inducing the use of sequences captured in natural situations that brings new challenges such as imbalance between observed activities and labeling ambiguity. This dissertation proposes a learning framework to address the problem of recognizing activities when exposed to two of these challenges: imbalance and ambiguity. Our approach is based on an architecture that combines graph convolution layers for Spatio-temporal agent poses analysis through a multi-scale approach and Transformers layers for context capture. Even though several methods have achieved high accuracy in benchmark datasets like NTU, their performance significantly decreases when tested in datasets with a high level of ambiguity among activities and an unbalanced number of samples for each class. We evaluated our architecture in the challenging BABEL dataset, where we achieved state of the art in terms of accuracy (65.4%) in action classification when considering both ambiguity and class unbalance. Furthermore, by observing activation profiles obtained by different models, we performed a qualitative analysis of how aspects of our approach contributed to the result obtained.

Keywords: action recognition, action embedding, multi-scale motion analysis, skeleton based motion analysis.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Humans are very good at making sense of what is going on with movements and assigning labels to the observed action. We have an innate ability to understand human behavior by analyzing a small set of human poses. Although in the past decade, we have witnessed the performance gap between humans and computers becoming smaller for many tasks such as image classification [19], scene recognition [55], and object detection [57], the same does not hold true for action recognition.

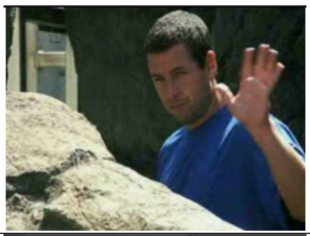
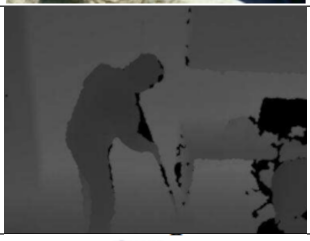
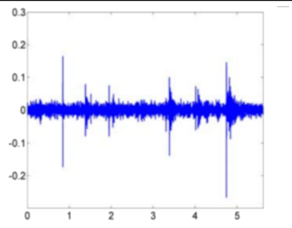
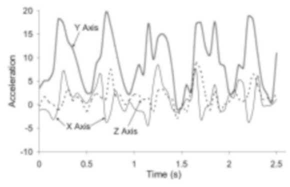
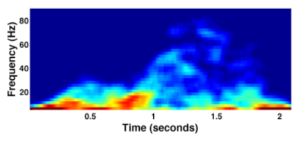
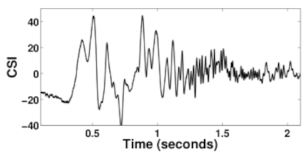
A comprehensive action recognition method has to deal with several practical challenges such as background noise, occlusions, variations in scale, lighting, and aspect. Furthermore, when we consider learning-based methods, the construction of datasets tends to be expensive and complex, leading to the use of sequences captured in natural situations. Using sequences captured in a natural environment implicitly brings the difficulty of maintaining a balanced number of samples that cover all the variations described. Even the balance among classes is a complex variable to control. Furthermore, sample labeling can be difficult to fit into a well-separated set of classes, leading to ambiguous classification.

Human Action Recognition (HAR) is a relevant problem in computer vision since it can have a significant impact in a wide range of areas such as sports development [44], health and safety [32], and surveillance [22]. Examples of practical problems that HAR can address are:

- Health and safety monitoring, by anomaly detection of potentially hazardous scenarios in factories;
- Security improvement, by detecting dangerous or aggressive actions;
- Sports performance, through the generation of team and athletes statistics and biomechanical metrics.

The first works in the field usually focused on using gray-scale or RGB videos as input due to their popularity and easy access [38]. Due to the development of different kinds of accurate, affordable, and wide available sensors, recent years have witnessed an emergence of works using other data modalities, such as skeleton [31, 30], depth [50], infrared signal [20], event stream filtering [16], audio processing [28], accelerometer data [56], radar [24], and even WiFi [48]. Those modalities are exemplified in Table 1.1.

Table 1.1: Action recognition data modalities

Mode / Example		Mode / Example	
RGB [26]		Skeleton [31]	
Depth [50]		Infrared [20]	
Point cloud [6]		Event stream [16]	
Audio [28]		Acceleration [56]	
Radar [24]		Wifi [48]	

Among the methods listed, action recognition based on skeleton data has received increasing attention due to its compactness and specialized ability to capture pose dynamics. In this approach the human skeletons are mainly represented as a sequence of joint coordinate matrices, where the coordinates are extracted by pose estimators like OpenPose [3]. Since it only includes the sequence of poses, the information provided is concentrated on the description of the movements performed, making it easy to follow the movement of the limbs of the person of interest. An example of poses extracted from an image can be seen in Figure 1.1. The data is focused on human motion information while being immune to contextual noise, such as background variation and lighting changes.

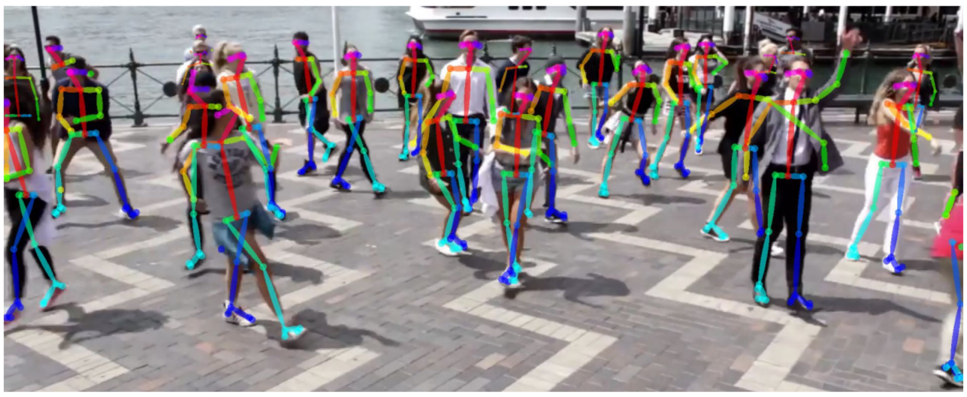


Figure 1.1: Example of skeletons extracted from an image. After extraction, the pose information is condensed in the graphs immune to background noise and lighting. Image extracted from [3].

1.1 Problem Definition

Although high accuracy have been achieved by classical action recognition skeleton-based datasets with simple actions (*e.g.*, NTU [30, 41] and Kinetics [21]), with the proposal of datasets with more natural and complex human motions, those methods have shown a significant drop in performance. Particularly, datasets such as BABEL, proposed by [39], highlight the challenge faced by action recognition tasks as they show low levels of accuracy when evaluating state-of-the-art methods on this dataset.

Human action recognition techniques face several challenges. One of them is the high level of motion ambiguity when classifying natural and common human movements. For instance, more general actions such as interacting with or using an object are easily misclassified with specific actions with finer and low-scale motions like taking or picking up something. Running and jogging actions, for example, show very similar motions for the joints and links in a human skeleton, albeit easily separable when considering the spatiotemporal relation among joints. Recent advances in human pose estimation [3, 2, 53], self-attention layers [45] and graph architectures [54, 7, 5] have shed new light on action recognition approaches. However, modeling the multiscale nature and spatiotemporal relation of the skeleton joints remain a key challenge in the action recognition field.

The ambiguity problem is also observed in datasets generated from descriptive data based on video sequences not specifically generated to support research in the field of action recognition. Datasets forged for research have clear and easily discriminated classes. In the case of Datasets generated from the description of sequences, the set of available class labels ends up having overlaps, which means that a certain observed action can receive more than one classification. In this case, the most likely classification is not enough to analyze the network performance; being necessary to observe all the most activated classes to evaluate the network performance. For example, the dataset may have ‘throw’ and ‘play sport’ labels. In this scenario, a video sequence showing a baseball player throwing a ball could be correctly classified with either of these labels.

The same holds true for the mentioned approaches to datasets with balancing problems. Given that data in the wild are naturally unbalanced, the class imbalance problem must be addressed. This imposes the need to find approaches to the situation not so oriented to finding the best result for a dataset but finding ways to represent the movement robustly and generically in such a way that it enables better performances even if identifying less represented classes.

This work proposes ways to tackle these problems by developing a model that combines a multiscale mechanism to capture fine and coarse motion and Transformers layers to capture context. The model is pre-trained in two different methods that target the creation of representations in the hidden layers of the network that allow robust description of the movement. Then, in a second stage, the model is fine-tuned to recognize actions from an unbalanced dataset.

1.2 Contributions

This thesis proposes a new learning framework based on two components: a multiscale spatiotemporal graph convolution layer and a Transformer module. Our model is trained using a stage-wise strategy. In the first stage, we optimize the network using one of the following approaches: (a) an encoder-decoder model that is trained to reconstruct the action through self-supervision in an auto-regressive task or (b) an encoder that is trained using contrastive learning to force the output vector to be able to separate different actions.

In both cases, the input is an action represented by a set of poses encoded as skeleton graphs, and the goal is to create a rich embedding of the action in the internal layers of the model. Then, the feature vectors extracted from our encoder are used to feed the classifier. Our encoder is built upon a Transformer architecture, which enables it to enrich the pose representation with context information. The experimental results show that our architecture is superior to the state-of-the-art methods in the challenging BABEL dataset in the most relevant indicators.

Our contributions are as follows:

- An architecture that uses a proposed multiscale component and Transformers layers to combine multiscale motion and context into an action embedding;

- A two stage-wise pre-training strategies in an action recognition problem: An auto-regressive approach, and a contrastive learning approach;
- The analysis of the relevance of focal loss to optimize models for unbalanced and ambiguous datasets.

Part of this work was submitted to 35th Conference on Graphics, Patterns and Images (SIBGRAPI), 2022.

1.3 Document Structure

This document is structured as follows. In Chapter 2, we discuss the works related to neural networks on graphs, skeleton-based action recognition, the usage of Transformers in computer vision problems, and contrastive learning. In Chapter 3, the model components are formalized, including the multiscale downsampling component. It also details the two proposed pre-training methods: auto-regressive pre-training and contrastive learning pre-training. The dataset, experimental setup, and results are discussed in Chapter 4. Finally, we conclude and present future works in Chapter 5.

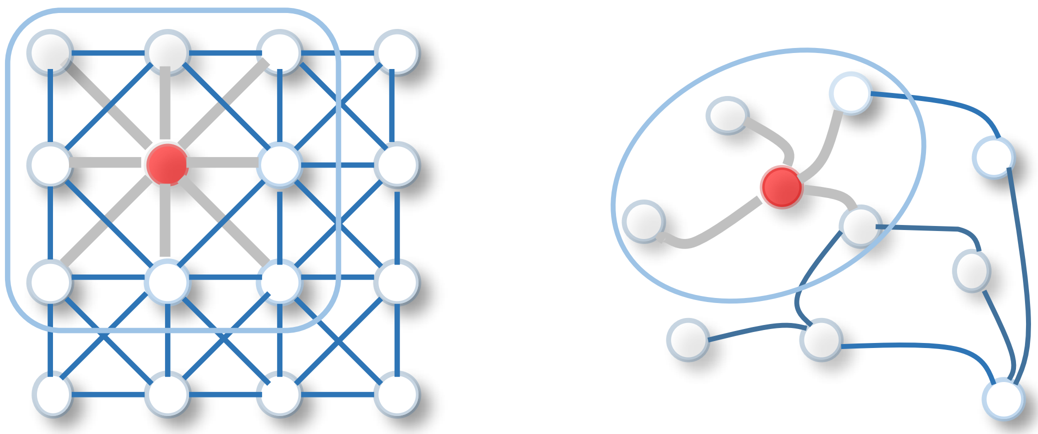
Chapter 2

Related Work

2.1 Neural Networks on Graphs

A graph is a highly flexible data structure that can be used to model a wide variety of problems [40] as it can, simultaneously, represent instances (nodes) and relationships (edges). This feature enables its usage in a large number of fields, including social sciences, chemistry, and logistics. Its representation power makes it especially interesting for applications in neural network architectures. However, its flexibility prevents its usage with typical neural network layers such as Convolution Neural Networks (CNNs) and Multilayer Perceptrons (MLPs).

Works developed by [14] and [25] represented a breakthrough in the area due to the ability shown by their methods to process graphs of arbitrary topology by an aggregation function that passes messages to a node from its neighbors. [18] propose to



(a) Convolution operation in a 2D Euclidean grid.

(b) Convolution operation in a graph. No regular structure.

Figure 2.1: Convolutions in images versus convolutions in graphs. a) Euclidean space with well defined structure. Operations on the surroundings are naturally defined for each position in space. b) Non-Euclidean space where the neighborhood is not defined by space structure, but by connections. Image extracted from [52].

make the aggregation function more generic, partitioning the adjacency matrix into two parts: loops and neighbors, and learning different weights for each partition. Finally, [46] added attention mechanisms to the message passing process, enabling to weight a set of neighbors based on their features. A comparison between a convolution in a typical Euclidean space and a convolution in a graph can be seen in Figure 2.1.

2.2 Skeleton Based Action Recognition

The problem of recognizing actions has been an object of study in the computer vision community for a long time. Before the dominance of methods based on deep learning, techniques for the generation and processing of handcrafted features were usual. Examples of classic ways to represent an action are:

- Spatiotemporal volume-based action representation;
- Spatiotemporal interest point (STIP), and
- Joints trajectory representation.

An example of a skeleton-based motion analysis without deep learning was presented by [47], who proposed a descriptor called Space–Time Occupancy Pattern (STOP), where the motion is represented by the occupancy level of a 4D space-time grid. From the grid cells, feature vectors are obtained, helping to classify the action. Still using classical approaches to skeletons, [9] combined a filtering strategy to select discriminative poses with a Latent-Dynamic Conditional Random Fields (LDCRF) model to discriminate actions in a video sequence.

In the realm of deep-learning-based methods, [54] work was quite influential because it presented a generic graph-based way of modeling the dynamics of a skeleton's movements. Using their model, the pose graphs extracted from each time frame are temporally connected, forming a single graph representing the entire action, as shown in Figure 2.2. This way of modeling the movement allowed the application of neural networks based on graphs directly on the input data for the action recognition task.

To capture richer dependencies between joints, [27] introduced two architectures. The first is called A-links, or actional links, which are responsible for learning action-specific dependencies directly from the data. The second structure, called S-Links, extends the existing skeleton graphs, adding higher-order connections to be able to capture more complex movements. The combination of these two structures, named Actional-Structural

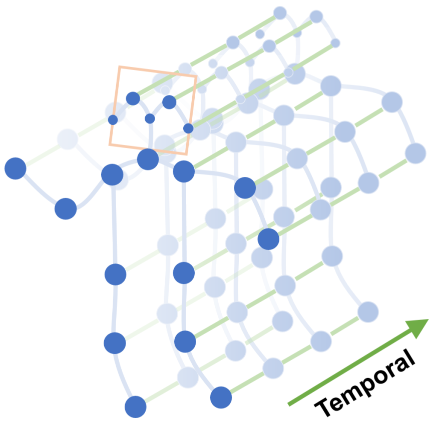
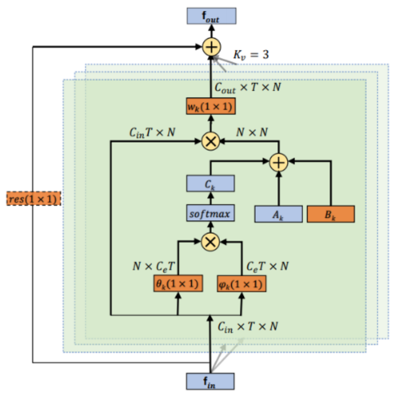


Figure 2.2: Human motion represented by temporally interconnected graphs. Note that a vertex is only connected to its neighbors in the same time frame and its respective vertex in adjacent time frames.

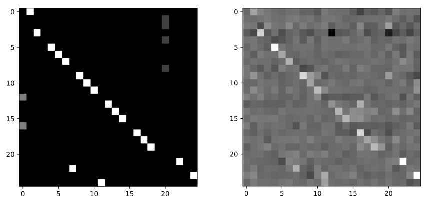
Graph Convolution Network (AS-GCN), can be used as a basic building block for learning both spatial and temporal features for action recognition.

Another extension intended to capture richer dependencies among joints was proposed by [42], where structural information,(*i.e.*, the adjacency matrix), is complemented by a set of learned weights capable of establishing higher-order topology connections. They proposed the adaptive graph convolutional layer, where the graph structure used during the convolution is a composition of three parts. The first part, A_k, is the same as the original $N \times N$ normalized adjacency matrix and represents the physical structure of the human body. The second part, B_k is also an $N \times N$ adjacency matrix, but, in contrast to A_k, the elements are optimized together with the other parameters during the training process. The third matrix C_k represents a data-dependent graph that is different for each sample. To determine whether there is a connection between two vertexes and how strong given connection is, a similarity measurement of the two vertexes is calculated. The layer topology and a comparison between the structural adjacency matrix and the learned adjacency matrix can be seen in Figure 2.3.

[7] adapted the shift convolution operation from CNN architectures to graphs, improving results when compared to previous works using fewer parameters. [13] propose a different method that relies on a 3D heat map stack instead of a graph sequence as the base representation of human skeletons. More recently, [5] propose the Multi-Scale Spatial Graph Convolution Module. Inspired by Res2Net [15], it enriches the receptive field of the model in spatial and temporal dimensions partitioning the feature space and applying successive convolutions, each time adding a new partition and increasing the receptive field.



(a) Illustration of the adaptive graph convolutional layer.



(b) A comparison between the structural adjacency matrix (left) and the learned adjacency matrix (right).

Figure 2.3: Adaptive graph convolutional layer. a) There is a total of three types of graphs in each layer, *i.e.*, A_k represents the original adjacency matrix, B_k contains an adjacency matrix learned by the model and C_k is a matrix that measures the similarity between each vertex pair. b) The left matrix is the original adjacency matrix for the second subset in the NTU-RGBD dataset. The right matrix is an example of the corresponding adaptive adjacency matrix learned by the model. Image extracted from [42].

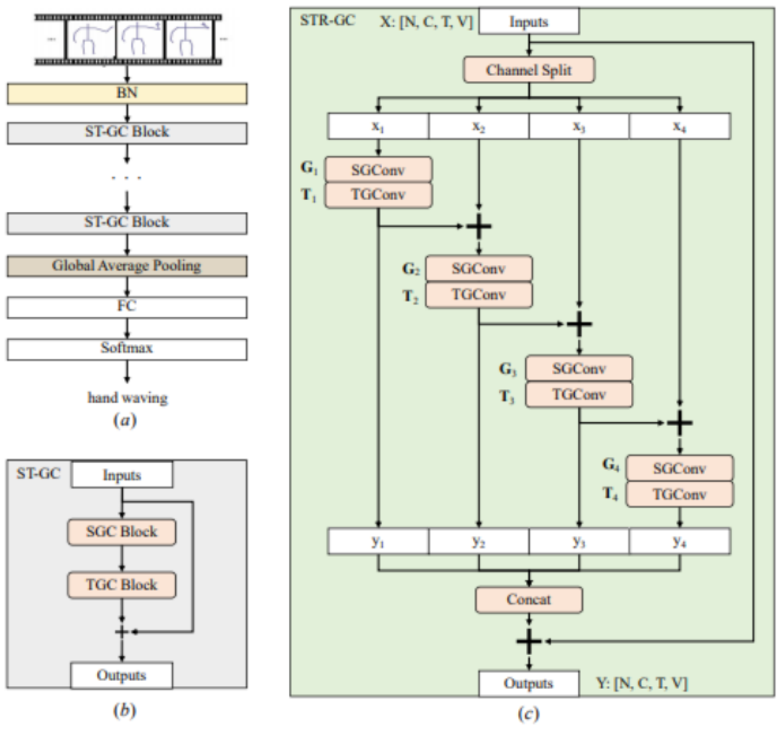


Figure 2.4: The MST-GCN block topology. Each key-point feature vector is partitioned. The convolution operations are applied in cascade, one partition per step. Image extracted from [5].

2.3 Transformers in Computer Vision

The Transformer, proposed by [45], is currently the basic building block for Natural Language Processing (NLP) models due to its power of linking context between words in a sentence, even in situations in which two related words are separated by many others. The architecture blocks are illustrated in Figure 2.5. The model proposed in their paper is composed of an encoder and a decoder.

The encoder's role is to convert the tokens of the input (*i.e.* a sentence) into a memory state. Then the state is passed into the decoder responsible for generating the model outputs. The encoder is composed of a stack of self-attention and feed-forward layers. In a self-attention layer, each token of the input sequence is converted into key, value, and query vectors. With these vectors, the layer can calculate how one token of the sequence is important for another. This allows a token in a given position to capture information from any other token, regardless of the distance separating them. The decoder has a similar topology, having an additional attention layer to enable the connection with the encoder output. The operation in the encoder-decoder connection layer is similar to a self-attention layer. The difference is that the key and query vectors are not calculated from the previous decoder layer but from the output of the last encoder layer.

Since the position of a word in a sentence plays a determining role in understanding the sequence, the Transformer also uses a mechanism to generate vectors that encode position. Positional encodings are vectors that describe the location of an entity in a sequence so that each position is assigned a unique representation. Each vector value is calculated from a set of sines and cosines functions. The resulting vector has the same dimension as the model input tokens so that it can be summed to the original vector, adding positional information.

After the Transformer, significant advances have been attained in the NLP area. [11] used the Transformer in a language representation model called BERT, designed to pre-train deep bidirectional representations from unlabeled text and then fine-tune the model for a specific task with just one additional output layer.

The Transformer's success inspired many researchers to apply its model components in computer vision tasks. [34] proposed ViBERT (inspired by BERT) connecting NLP and computer vision in task agnostic self-supervised pre-trained models that produce rich representations that can be later fine-tuned. [12] explore the usage of Transformers blocks in computer vision tasks with only minor changes to adapt the input, an image, to a series of tokens. [51] propose The Visual Transformer, which is a model for tokenizing the image feature map into semantic groups through self-attention.

For human action recognition, [36] proposed a simple method entirely based on the Transformer architecture. The pose sequence key points are mapped by a simple

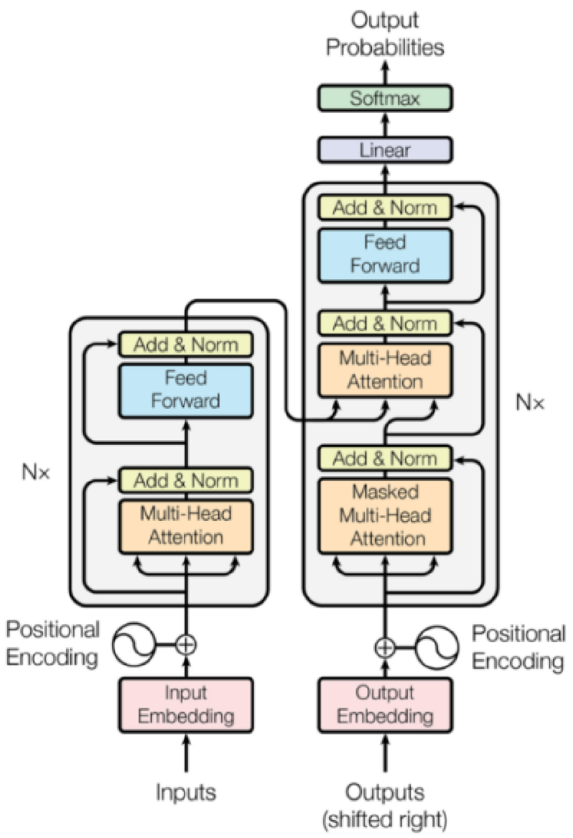


Figure 2.5: The Transformer architecture. Image extracted from [45].

linear transformation to match the Transformer input space dimensions. Additionally, a class token is included at the beginning of the sequence and is responsible for aggregating information through the layers. The class token representation after the last transformer layer is served for the classifier.

[37] also applied the Transformer architecture to the human action recognition problem. Instead of the typical structural adjacency matrix, they use the Transformer self-attention blocks to model dependencies between joints. The ST-GCN's spatial convolution (GCN) is replaced by a Spatial Self-Attention module (SSA) which is used to capture intra-frame interactions between different body parts, and the ST-GCN's temporal convolution (TCN) is replaced by a Temporal Self-Attention module (TSA) in order to model inter-frame correlations.

2.4 Contrastive Learning

Contrastive learning is a technique that can be used to train a model to learn representations of data such that similar samples are closer in the embedding space, while dissimilar ones are far apart. Contrastive learning can be applied to both supervised and unsupervised data and has been shown to achieve good performance on a variety of tasks. [4] propose a simplified contrastive self-supervised learning algorithm for image classification tasks, and through systematic study of the major components of their framework, they identify important factors to improve the results of the application of this technique, *i.e.*, (1) composition of data augmentations are very important in defining predictive tasks, (2) introducing a learnable nonlinear transformation between the image representation and the vector presented to the contrastive loss substantially improves the quality of the learned image representation, and (3) contrastive learning benefits from larger batch sizes and more training steps compared to supervised learning algorithms.

[23] state that besides good results achieved by unsupervised contrastive learning approaches, the cross-entropy loss remained the preferred method to achieve state-of-the-art results in problems like large-scale datasets image classification. This indicates an opportunity to generalize the contrastive loss to a supervised mode, using the labels to leverage the results. They proposed a loss that, instead of using data augmentation to produce *positive* samples and choose randomly from the mini-batch *negative* samples, the *positive* and *negative* samples are selected considering the labels. This makes the algorithm to produce clusters of the *same class* and pull off *different classes* in the embedding space. This approach permitted the authors to achieve state-of-the-art results in ImageNet [10] dataset.

Most of the relevant works that applies contrastive learning in computer vision is focused on providing good image classifiers. Consequently, those methods fail to produce good results for problems that involve dense predictions like image segmentation or object detection. To bridge the gap, [49] propose a dense self-supervised learning method that directly works with local features. This is achieved by segmenting the feature map after passing the image through a backbone network and producing for each segment a set of *negative* samples from segments of other images and, for the *positive* sample, finding the best correspondence of the key segment in a data augmented view of the same image.

Another relevant gap in applying contrastive learning in visual tasks is filled by [1] in a sequence-to-sequence method for text recognition. Their method splits the image into instances and views each feature map as a sequence of individual instances. This allows applying contrastive learning at a sub-word level, such that each image yields several positive pairs and multiple negative examples. The sequence of the instances is considered, especially during the augmentation procedures, to preserve the consistency

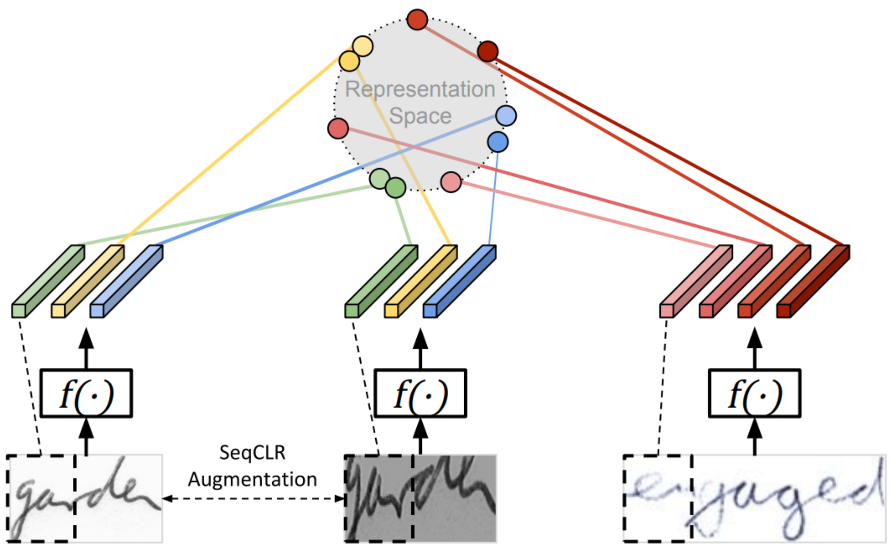


Figure 2.6: Sequence level contrastive learning. The image is split into instances and views each feature map as a sequence of individual instances. The contrastive loss is applied at the sub-word level. Image extracted from [1].

among different views. Failing to consider a sequence would lead to a poor generation of *positive* samples. Their method is illustrated in Figure 2.6.

Contrastive learning has also been applied to tasks related to action recognition. In the domain of RGB image sequences, [43] propose a self-supervised method where the data augmentation is built from the variation of the video speed. They consider that no matter how fast the video is, the action represented in the sequence of images remains the same. For skeletons as input data, [17] also adopt the self-supervised approach. They seek to improve the generation of positive and negative samples through a combination of eight forms of augmentation, followed by filters that seek to optimize a bank of samples with different movement patterns to improve the universality of the learned representations.

2.5 Focal Loss

Classification tasks usually use categorical cross-entropy loss. Since entropy refers to the disorder of a system, it quantifies the degree of uncertainty in the model's predicted value for the variable. The sum of the entropies of all the probability estimates is the cross-entropy. The cross-entropy function, however, does not perform well in all situations. It performs poorly particularly in situations where the classes of the dataset are unbalanced, and it has no mechanism to distinguish between hard and easy examples.

To address the problem of training from unbalanced data, the cross-entropy function may be changed by introducing a weighting factor that is inversely proportional to the effective number of samples [8]. The class-balanced loss term can be applied to a wide range of deep networks and loss functions. For a long-tailed dataset where major classes have significantly more samples than minor classes, setting the weighing factor properly re-balances the relative loss across classes and reduces the drastic unbalance of re-weighing by inverse class frequency.

After investigating the differences in performance of one-stage and two-stage object detectors, [29] concluded that it was a result of extreme class imbalance encountered during training. To control the problem, they proposed reshaping the standard cross-entropy loss such that it down-weights the loss assigned to well-classified examples, focusing the training on a sparse set of hard examples and preventing the vast number of easy negatives from overwhelming the detector during training. Easily classified negatives comprise the majority of the loss and tend to dominate the gradient. They proposed to add a modulating factor to the cross-entropy loss with a tunable focusing parameter. The focal loss value is defined by the equation:

$$FL(p_t) = -(1 - p_t)^{\lambda} log(p_t), \tag{2.1}$$

where $FL(p_t)$ is the calculated loss value and p_t is the probability of sample t be correctly classified. The function is configured by the parameter λ, which defines the focus level. The behavior of the loss function for different focusing parameter values can be seen in Figure 2.7.

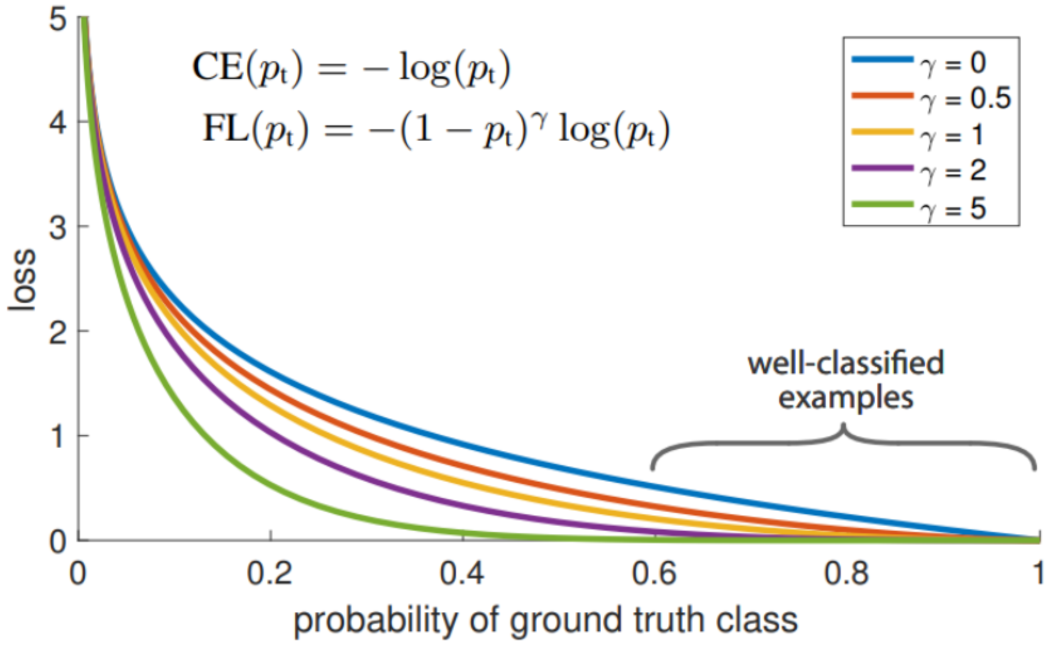


Figure 2.7: Focal loss. The graph shows the behavior of the focal loss for different values of the focusing parameter γ. Well-classified examples receive less attention, reducing their influence in the final loss value. Image extracted from [29].

Chapter 3

Methodology

In this chapter we present the proposed architecture which is based on an encoder that combines two components: multiscale downsampling and the context component based on Transformer. After applying a pose extractor to the image sequence, the skeletons are processed by these two components, the first being responsible for transforming a series of pose graphs into a sequence of pose embeddings. The second is responsible for processing them, along with an additional token – the action embedding – which will be ultimately used to classify the action. The whole process is illustrated in Figure 3.1. The chapter also describes the upsampling layer that can be used to extend the model depending on the applied pre-training mode. Finally, the pre-training modes are described in detail.

3.1 Model Input

The proposed methodology is based on the processing of skeleton graphs. As discussed in Chapter 1, the use of skeletons as input has advantages in terms of separating the activity recognition problem into two steps. First, the pose skeleton is estimated, which can be done in different ways and using various resources, such as MoCaps, which

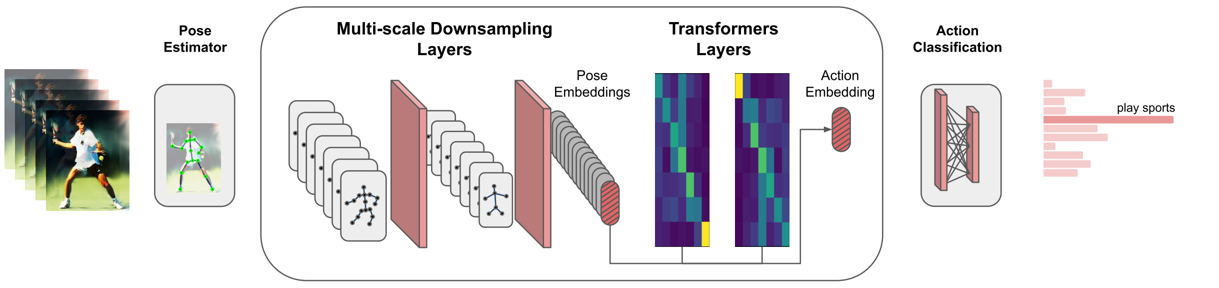


Figure 3.1: After using a pose estimator, a series of downsampling operations are applied to the skeleton graphs until they are reduced to a series of pose embeddings. The sequence of feature vectors, together with a summary token, are presented for the Transformers layers. Finally, the summary token is given as input to a classifier to generate the final classification.

generate the poses directly, or from the processing of RDB-D cameras or even simple cameras. Then the sequence of poses encoded in graphs is analyzed.

Depending on the way of capturing the data, the information can be in 3D or 2D space. Captures performed with MoCaps will naturally generate data directly in three-dimensional space. Captures made with RGB-D cameras will allow estimating the poses on the images and, by adding depth information, coding the pose in a three-dimensional space. A simple camera will generate key points with information only in two-dimensional space. A stereo pair montage can be used to estimate poses in a three-dimensional space, inferring depth by parallax.

The methodology proposed in this dissertation uses skeletons extracted from a scene in three-dimensional space. The way poses are captured may vary depending on the data source. The preferred form of capturing is through MoCaps, which guarantees the skeleton's completeness and produces a less noisy data stream. However, the proposed method can also use extraction from RGB-D and stereo pair cameras without changes.

The collected data are pre-processed for input into the model, translating and rotating the skeleton coordinates so that the base of the pelvis is positioned at the coordinate system's origin and the line connecting the left and right shoulders starts aligned with the X axis. This processing is made to normalize the coordinate system between the different samples and thus facilitate the network learning process concerning the motion analysis.

The use of skeletal data in 2D space is also possible. The model requires no modifications since the number of coordinates for each key point of the skeleton is configurable through parameters. However, skeletal data in 2D space needs to be treated with cautiousness due to the large differences in appearance caused by observations made from different perspectives. As it is not possible to rotate the skeleton on the vertical axis and thus normalize perspective, as can be done in the case of skeletons in 3D space, restrictions on image capturing need to be imposed, such as the definition of maximum angles of inclination and lateral deviation to the observed person.

3.2 Model Blocks

The encoder can be split into two components. First, the multi-scale downsampling is responsible for transforming through a series of downsampling steps a sequence of poses into a sequence of feature vectors, each feature vector representing a pose. The second component is the Transformer encoder which is responsible for enriching pose representations with context information.

The decoder has three components. The first is the same multi-scale downsampling described above. The downsampling weights in both encoder and decoder branches are shared, since they have the same objective. The second component is the Transformer decoder, which receives the action embedding from the encoder branch as memory. The third component is the upsampling, which has the objective of predicting the next pose considering the sequence of poses received as input so far.

The output of the encoder branch can be interpreted as an action embedding that summarizes the whole action. The action embedding can be input into a final task. In this work, we tested the embedding in an action classification task. In the following sections, we describe each component of the architecture in detail.

3.2.1 Multi-scale Downsampling

The multi-scale downsampling is a series of k layers that gradually reduces the number of nodes V of the input tensor $\mathbf{X_k}$, simplifying the skeleton graph, while keeping semantic information necessary to describe the pose. In our case the pose skeletons are reduced from $V_0 = 25$ in the input tensor, $V_1 = 19$ after layer $k = 0$, $V_2 = 13$ after layer $k = 1$, $V_3 = 5$ after $k = 2$ and $V_4 = 1$ after $k = 3$. The whole downsampling scheme is illustrated in Figure 3.2.

The downsampling sequence was defined as seeking to reduce the number of skeleton nodes in a balanced way, avoiding the reduction of the number of nodes being abrupt in a given step and insignificant in another step. Therefore, we sought to eliminate six nodes from the graph at each step, which can be done in steps 0 and 1. Extra care should be taken not to eliminate nodes in an adjacent position on the graph, as this would make the transmission of information between nodes in opposite positions harder.

Another factor that influenced the definition of eliminated nodes was prioritizing the maintenance of arms nodes for longer during the execution of the downsampling steps. This decision was motivated by the understanding that the arms make more complex movements, and its reduction in the first steps would make it difficult to learn through the model of relevant features to understand the action performed.

Each layer k is composed of two modules. First, the data passes through an aggregation step implemented by a graph convolution network responsible for extracting pose and motion features. The aggregation module could follow any spatiotemporal convolution block such as those proposed by [54], [27], [42] and [5]. This work uses the AGCN convolution block as proposed by [42]. The AGCN convolution operation is defined by

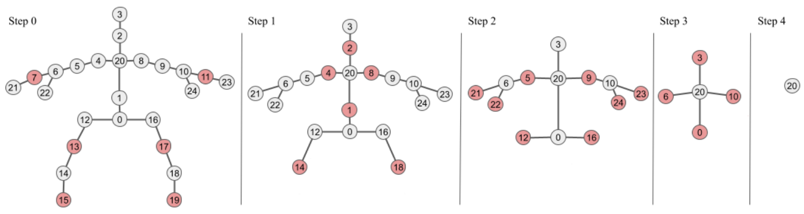


Figure 3.2: Downsampling scheme - The skeleton graph is reduced by each layer k according to the steps above. The eliminated nodes in each step are highlighted in red. In the final stage, features and structural pose information are condensed into a single vector.

$$h_k = W_{hk} X_k (A_k + B_k + C_k), \tag{3.1}$$

where $\mathbf{h_k}$ is the tensor after the aggregation operation, $\mathbf{W_{hk}}$ is the matrix of weights learned by the network, $\mathbf{A_k}$ is the adjacency matrix of the graph that models the skeleton in layer k, $\mathbf{B_k}$ is a matrix of weights learned by the model to allow the network to establish connections between nodes that are not linked by $\mathbf{A_k}$. $\mathbf{C_k}$ is a calculated matrix that measures similarities between two node feature vectors. The matrix $\mathbf{C_k}$ acts like an attention mechanism, considering that nodes with similar features should communicate with each other. The matrix $\mathbf{C_k}$ calculation involves a linear transformation of the input vector by the parameters $\mathbf{W_{\theta k}}$ and $\mathbf{W_{\phi k}}$, which are weights learned during training. This operation is given by equation

$$C_k = X_k^T W_{\theta k}^T W_{\phi k} X_k. \tag{3.2}$$

The second step is the downsampling module which is responsible for simplifying the skeleton, reducing the number of nodes from V_k to V_{k+1}, as defined by

$$Y_k = W_{sk} A_k h_k, \tag{3.3}$$

where $\mathbf{W_{sk}}$ is a weight matrix of dimension $V_{k+1} \times V_k$. $\mathbf{Y_k}$ is the output tensor of layer k.

The multi-scale features are extracted from the output tensor $\mathbf{Y_0}$, and injected into the upstream layers. Let V_a, V_l and V_b be the set of node indexes i in dimension V_0, representing arm nodes, leg nodes and body nodes. The multi-scale features are constructed through the following rule:

$$\begin{cases} x_{arms} = max(y_{0i}) & \forall i \in V_a \\ x_{legs} \;= max(y_{0i}) & \forall i \in V_l \\ x_{body} = max(y_{0i}) & \forall i \in V_b \end{cases} \tag{3.4}$$

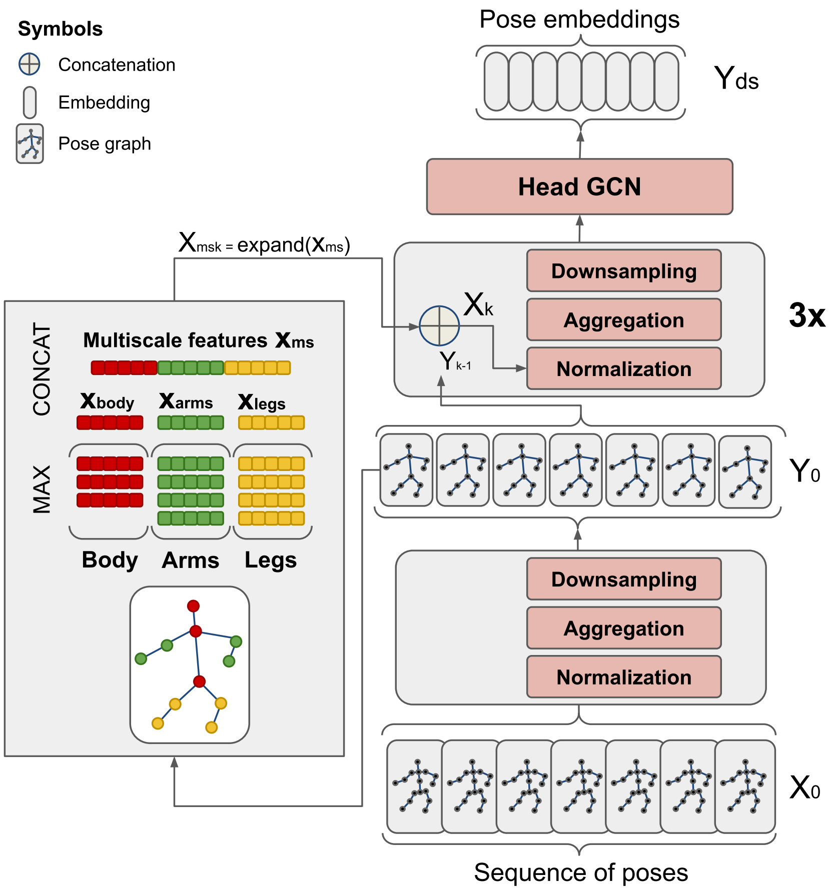


Figure 3.3: **Multiscale Downsampling Component**. The downsampling component receives a sequence of poses. Each pose is downsampled until it becomes a single vector. After the first downsampling step, features from the body, arms, and legs are extracted and max-pooled. Those features are concatenated to form the multi-scale features which are injected in all nodes of all subsequent layers, finally producing the pose embeddings.

Then these components are concatenated to form the final multi-scale vector $\mathbf{x_{ms}}$, as defined by Equation 3.5:

$$\mathbf{x_{ms}} = \mathbf{x_{arms}} \oplus \mathbf{x_{legs}} \oplus \mathbf{x_{body}}. \tag{3.5}$$

The vector $\mathbf{x_{ms}}$ is stored internally and, in each subsequent layer, it is expanded into the matrix $\mathbf{X_{ms_k}}$ to meet the number of nodes in each pose input into that layer. Finally, the expanded vector is concatenated with the previous layer output to form the next layer input, as defined by Equation 3.6:

$$\mathbf{X_k} = \mathbf{X_{ms_k}} \oplus \mathbf{Y_{k-1}} \tag{3.6}$$

The organization of the downsampling component can be better seen in Figure 3.3.

3.2.2 Context Components

The context layers are, in fact, a standard Transformer's encoder-decoder pair as proposed by [45]. In our method, a stack of three identical layers composes the encoder. Moreover, each layer is composed of two sub-layers: a multi-head self-attention mechanism and a fully connected feed-forward network. The decoder is also composed of three identical layers. Besides the sub-layers already described for the encoder layer, it adds a third sub-layer, which performs multi-head attention over the output of the encoder stack.

The layer's input is the sequence of pose embeddings $\mathbf{Y_{ds}}$ produced by the downsampling component. A summary token is concatenated to the sequence of pose embeddings, whose values are parameters learned by the network. The token is responsible for collecting and summarizing context data from Transformer layers in such a way it can describe the entire action. Finally, a positional encoding vector is added to the embeddings, as proposed by [45]. It allows the network to sense the relative position of each token in the sequence.

The encoder is present both in the pre-training and in the fine-tuning phases. The encoder output is a sequence of arrays, where the first vector is the summary token after passing through the Transformer's layers and collecting context data. We call this token Action Embedding Token. The remaining vectors are pose embeddings enriched by context information. The organization of the context component is depicted in Figure 3.4.

The decoder is present only during the pre-training phase when the model is pre-trained using the reconstruction mode. It receives a series of pose embedding produced by the downsampling component as input. In the decoder branch, the poses are shifted back one-time step, and a subsequent mask is applied to the self-attention block to model an auto-regressive task. The encoder information is provided by the action embedding token, which is input to the encoder-decoder attention block of each decoder layer. The output of the decoder is a sequence of arrays, one for each pose, with features capable of describing the pose with sufficient information to reconstruct the pose.

3.2.3 Upsampling

The upsampling block is present only during the reconstruction pre-training phase. It is responsible for translating a pose embedding outputted by the decoder into a pose skeleton in three-dimensional space. The upsampling component is similar to the down-

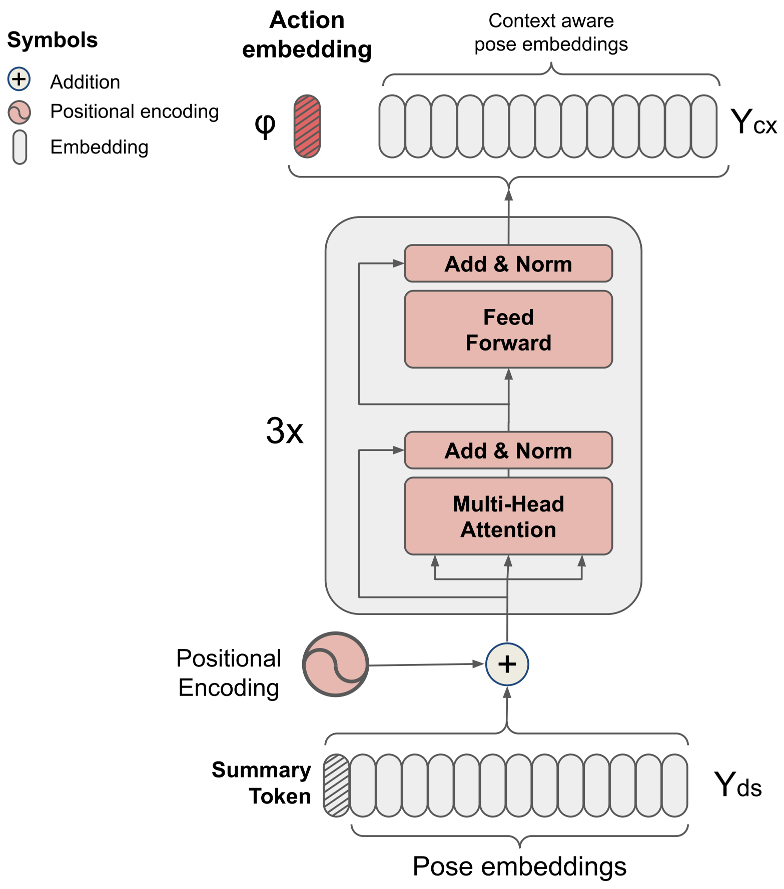


Figure 3.4: **Encoder Context Component**. The pose embeddings feed a stack of transformer layers together with a summary token. The transformers enrich each token with context information. The summary token collects context information in each layer producing the action token

sampling component, simply replacing the downsampling operation with an upsampling operation. The latter is responsible for increasing the number of nodes in the graph following the inverse order of the downsampling. The upsampling operation is defined by

$$Y_{uq} = W_{uq} A_q h_q, \tag{3.7}$$

where $\mathbf{W_{uq}}$ is a weight matrix of dimension $V_{q+1} \times V_q$. $\mathbf{Y_{uq}}$ is the output tensor of upsampling layer q.

3.2.4 Classifier

The classifier component is present only in the fine-tuning phase. It is a composition of two fully connected layers. The first layer receives the action embedding token produced by the encoder. The second layer is responsible for producing logits that classify the action. The complete fine-tuning architecture is shown in Figure 3.7.

3.3 Training and Classification

Pre-training has become a successful paradigm in many computer vision tasks. In a typical setup, models are initially trained with a self-supervision algorithm on a given dataset (usually larger) and then fine-tuned on target tasks, typically using less training data. The self-supervised algorithms can be broadly split into two categories: reconstruction methods and contrastive methods. Both of these methods were experimented during this research. After the pre-training, the encoder branch is extracted, and a final layer responsible for the activity recognition is added.

The action is modeled as a sequence of poses represented by a graph and stored in a tensor $\mathbf{X_0}$. The tensor $\mathbf{X_0}$ has dimensions $T \times V \times C$, where T is the number of poses, V is the number of vertices of the skeleton graph, and C is the number of spatial dimensions. In our case $C = 6$ (a three-dimensional vector defining the position of the node and a vector describing the velocity of the node between time frames).

3.3.1 Reconstruction Pre-training

The reconstruction approach is inspired by [34], but instead of representing visual and linguistic characteristics, the objective is to create a rich representation of the action in the hidden layers of the model, especially in an encoder output vector that we call action embedding. Since it is a self-supervised method, the objective of the pre-training is to reconstruct the pose sequence itself through an encoder-decoder model. The setup is illustrated in Figure 3.5.

For pre-training, an auto-regression pretext task is used, whose objective of the model is, given the input of a partial sequence of poses of a given motion, to predict a new pose that would better allow the reconstruction of the movement. This operation is applied repeatedly, concatenating the new pose obtained from the decoder output to the end of the sequence given as input in the previous step. Ideally, the model should be able to reconstruct the entire motion. Pretext tasks like these are used in other approaches to generate robust representations in the internal layers of the network, capable of correcting

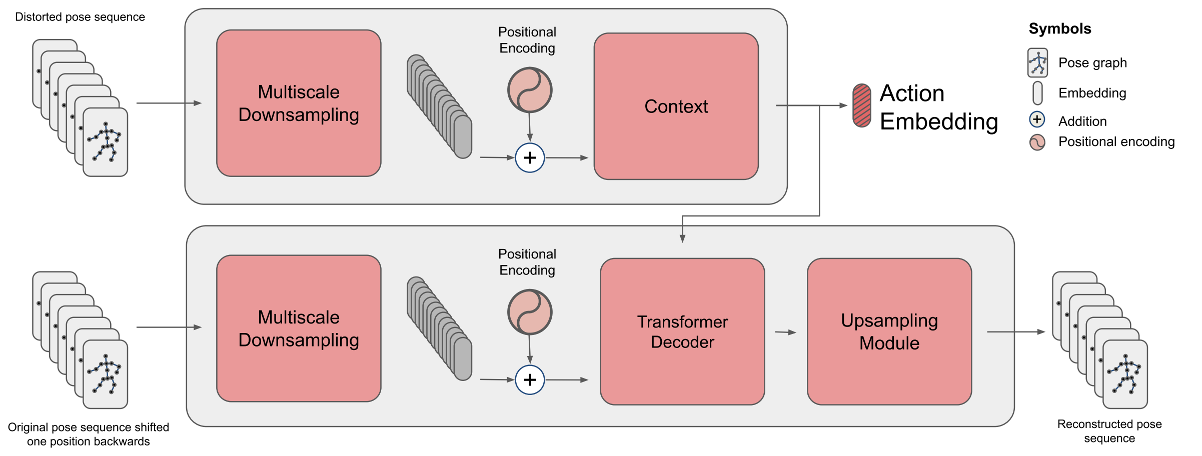


Figure 3.5: **Pre-training using reconstruction** - The pre-training architecture is composed of a auto-regressive encoder-decoder pair whose objective is to reconstruct a sequence of poses. The encoder is connected to the decoder by an action token which is trained to encode the whole action into a single vector.

errors in the input data.

In our setup, this is done by inputting the entire motion sequence into the encoder, which will be responsible for encoding the motion. To render the task more challenging, operations to distort the data are applied to the input sequence. The following operations were performed on input data to cause damage:

- Adding random noise to pose key points, generated from a normal distribution;
- Deleting pose key point information (set position coordinates to zero);
- Deleting all information of a given limb (*e.g.*, erasing the skeleton leg during 5 time frames);
- Deleting all information of a given frame.

All distortion operations were applied during pre-train. They were selected for each batch by a random process during the training process. The distortion operations were modeled as transformations through which input data were submitted before entering the network.

The noise addition is applied to all key points with the generation of samples from a normal distribution, then the noise is added to the original key-point coordinate. The coordinates deletion starts with the selection of one of the key points, and then all its coordinates are set to zero. For limb deletion, key points were grouped into four groups: right arm, left arm, right leg, and right arm. One group is randomly selected, and then the coordinates of all key points belonging to this group are set to zero during five time frames. The time window is also selected at random. Finally, for frame deletion, a frame is selected at random, and then the coordinates of all key points are set to zero.

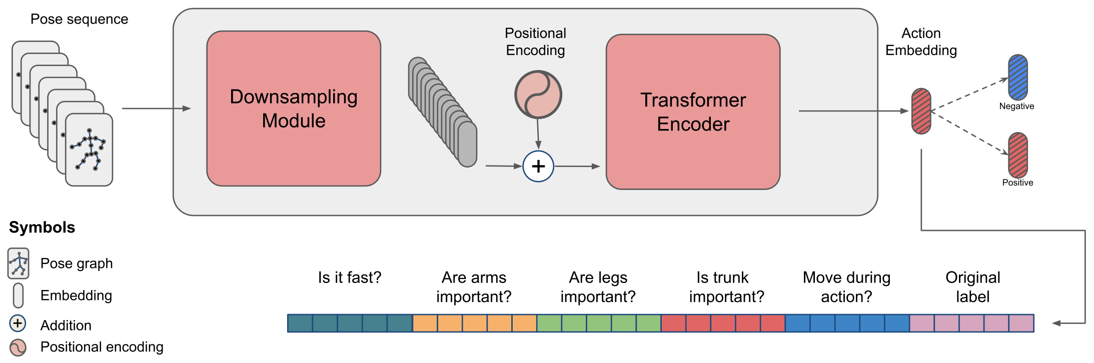


Figure 3.6: **Pre-training using contrastive learning** - Contrastive loss is applied to the action embedding. However, the comparison is not made between the complete vectors. The vector is partitioned into six parts, and the positive and negative samples are selected according to the pseudo-labels.

For the decoder to be able to reconstruct the movement, a partial sequence of data is provided (simulating the result of previous executions of the procedure). For efficiency, this is not done step by step, but by shifting the entire input sequence a time step to the past and adding a mask that hides from a given token position all the information that would be in subsequent positions.

3.3.2 Contrastive Pre-training

The contrastive pre-training adopts a supervised approach. Since the objective of this phase is not to classify the sample but to generate a rich action representation at the end of the encoder, the labels were not directly used to contrast embeddings by class but used as a reference to generate a set of pseudo-labels. The criteria behind each pseudo-label are to semantically be capable of encoding characteristics considered as important to understand the movement, especially for disambiguate classes with similar pose sequences. For instance, walk and run have similar pose sequences, but the speed of the movement is different. To be able to disambiguate those classes, it would be interesting to encode the speed of the movement in the action embedding. To accomplish this objective, a pseudo-label indicating if a movement is fast or not was defined and used to contrast samples.

Five pseudo-labels where created: 1) *Move* indicates if the action typically involves the displacement of the actor; 2) *Hands* indicates if, to perform the action, the actor needs to use hands and arms; 3) *Feet* indicates if, to perform the action, the actor needs to use feet and legs; 4) *Trunk* indicated if to understand the action, attention must be paid to the trunk movement and, finally, 5) *Fast* indicates if the movement described is typically fast. The way the actual classes were mapped to the pseudo-labels can be seen in Table 4.1 and Table 4.2.

To calculate the model loss, the encoder output vector was segmented into six equal parts. The first five parts were assigned for the defined pseudo-label. Each of the five first segments was presented to the contrastive loss function together with positive and negative samples selected accordingly to their respective pseudo-label. The goal of this arrangement is to force each part of the embedding to encode as well as possible the characteristic mapped by the pseudo-label. The 6th segment was presented to the contrastive loss with the original class label to capture special characteristics. The setup is illustrated in Figure 3.6. The partition loss value $L_{constrastive}$ is defined by Equation 3.8

$$L_{constrative} = [m_{pos} - s_p]_+ + [s_n - m_{neg}]_+, \tag{3.8}$$

where m_{pos} and m_{neg} are the positive and negative margins and s_p and s_n are the similarity measurement of the anchor compared to the positive sample and negative sample respectively. The similarity measurement is defined as the dot product between the normalized anchor embedding and the normalized sample embedding. The summation of all partition losses computes the final loss.

3.3.3 Fine-tunning

The fine-tuning process is performed after pre-training. During the previous phase, the model is optimized to generate an internal representation of the action that is generic and robust. In the fine-tuning stage, the network is optimized for a specific objective. The network, however, is not trained from scratch but has as starting point the state of the model after the pre-training.

The fine-tuning can be performed for different tasks. In this work, the task selected was action classification. This forces us to change our model, including a final module responsible for receiving the action embedding as input and generating the probabilities of a given sample belonging to one of the pre-established classes as output.

The classification module was built as a simple MLP (Multi-Layer Perceptron), consisting of 3 layers: two hidden layers and an output layer, all fully connected. Among the hidden layers, the ReLU activation function was used. In the final layer, the activation function used was the softmax since it is necessary to generate the probabilities for

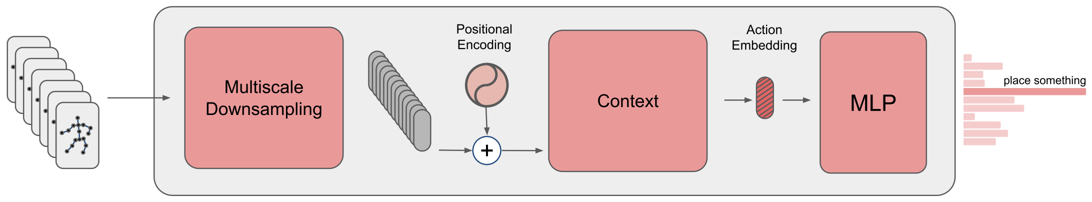


Figure 3.7: **Fine-Tunning Phase** - In fine-tunning architecture the decoder branch is dropped and a classifier is added after the encoder. The classifier uses the action embedding as input and outputs score values for each class.

each class. The architecture after the inclusion of the classification layer can be seen in Figure 3.7.

The dimension of the MLP input vector corresponds to the dimension of the action embedding, and the module output depends on the dataset used for training. In the experiments, described in Chapter 4, the output dimension varied between 60 and 120. The inner layers had dimensions fixed at 128 for the two layers.

Chapter 4

Experiments

In this chapter, we first present the dataset used in the experiments and detail our testing setup and implementation. We then discuss our results, comparing our performance with the current state-of-the-art. Finally, we perform an ablation study to analyze the effect of each major component of our architecture.

4.1 Dataset

To validate our methodology, we used the BABEL [39] action recognition dataset. BABEL is a large dataset with language labels describing the actions being performed tracked by a mocap system and it was conceived to support research aiming to understand the semantics of human movement. It originally consisted of action labels for about 43 hours of mocap sequences from AMASS [35]. There are over 28 thousand sequence labels and 63 thousand frame labels in BABEL, pertaining to 260 unique action categories.

The authors of BABEL, however, proposed a pre-processed version of the dataset as a new benchmark for action recognition since current state-of-the-art models reach above 95% of accuracy in most used benchmark datasets such as NTU RGB+D 60 [41] and NTU RGB+D 120 [30], leaving little room to estimate the progress of new action recognition models. The pre-processed version of the dataset has the characteristics desired for this work, (*i.e.*, a high level of ambiguity, as we can see in Figure 4.1), and an unbalanced distribution of samples, as shown in Figure 4.2.

The BABEL dataset for action recognition is organized in two subsets: BABEL 60, containing 45,473 samples and representing the 60 more common action labels of the original dataset, and BABEL 120, contains the same samples from BABEL 60, plus 3,505 additional samples of the following 60 more common classes of the original dataset, totaling 48,978 samples and 120 classes.

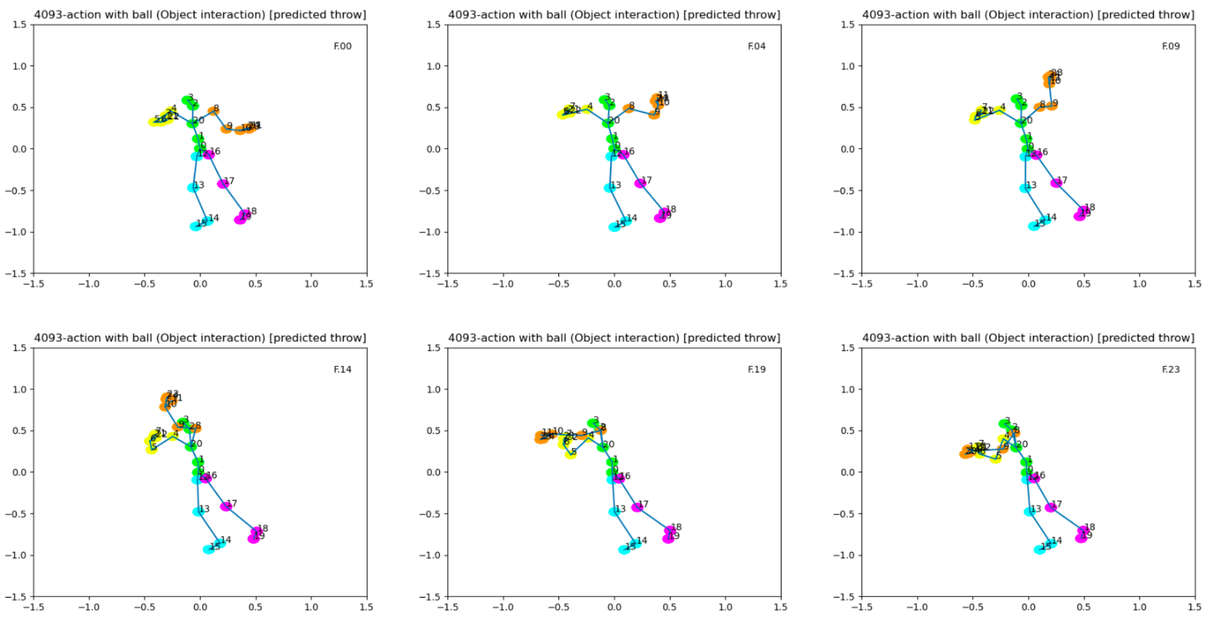


Figure 4.1: This sequence of poses illustrates a sample dataset that has ambiguous classification. The dataset authors defined the ‘action with ball’ label for the action. If we carefully look at the sequence, we see that the person is throwing something (probably a ball); thus, the authors could select the label ’throw’ for the same sample.

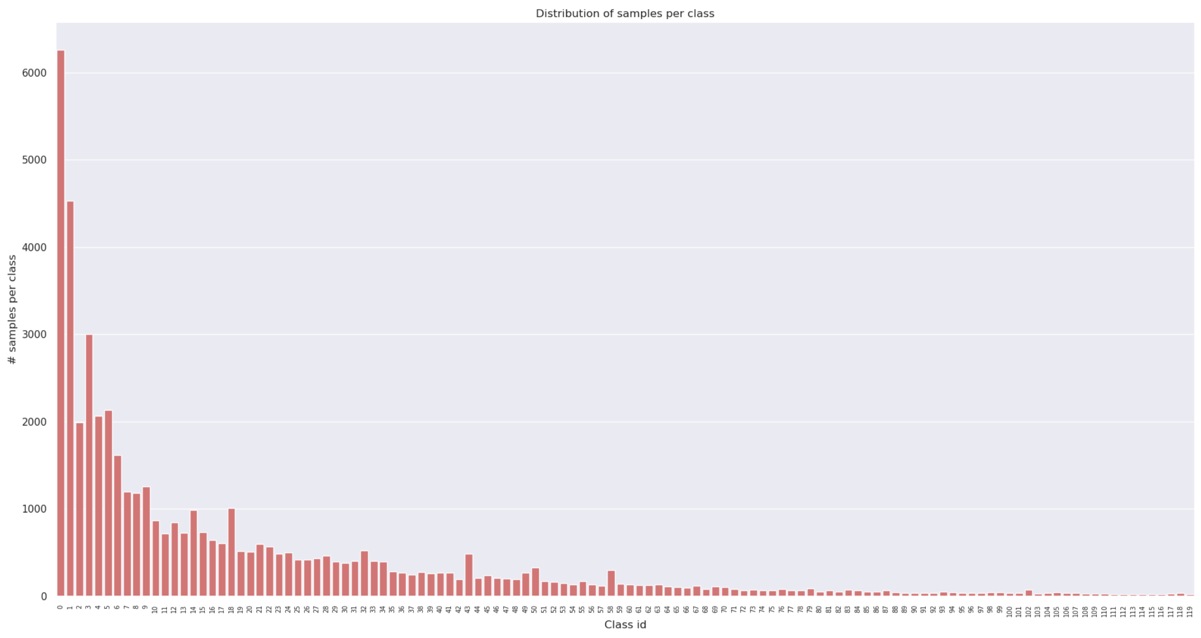


Figure 4.2: Distribution of samples per class, showing that BABEL is a very unbalanced dataset. The first ten classes amount to over 50% of all samples.

4.2 Implementation Details

4.2.1 Data Pre-processing

Each pose skeleton was normalized by transforming the key-points coordinates such that the middle spine is fixed at the origin, shoulder blades are parallel to the X-axis, and the spine to the Y-axis, following the methods proposed by [41] and repeated by [39]. All pose sequences are sampled at 30 fps and limited to five seconds. Poses sequences that span less than five seconds are repeated until it completes five seconds. After normalization, key point velocity features are calculated and concatenated to the input tensor. The velocity is calculated by subtracting node spatial features of two adjacent temporal frames.

4.2.2 Pseudo-label Assignment

As described in the methodology chapter, for the contrastive learning pre-training approach, a variation of supervised learning was adopted. But instead of using class labels directly, a set of pseudo-labels was defined in order to cluster the samples according to a list of characteristics considered important to reduce ambiguity among samples. For this pre-training approach, the BABEL 60 dataset was used since its number of samples covers close to 90% of the entire BABEL dataset for action recognition.

To generate the pseudo-labels each label was evaluated according to its semantics. No pose sequence data were analyzed to help define the pseudo-label, only the label title itself. For instance, the *run* label received the *fast* label since the run activity, by definition, involves a person moving fast. The same is not valid for the class *jog*, since, even though the pose sequence of a person jogging is very similar to a sequence of poses of a person running, the speed is greater when analyzing how fast a person runs compared to how fast a person jogs.

For each sample of the dataset, the class label was mapped to a set of five true-or-false variables, each one containing the value of the pseudo label of one specific characteristic. Those values were saved into a data structure to be used by the pre-training algorithm. The values selected for each pseudo-label are organized in Tables 4.1 and 4.2.

Table 4.1: Pseudo-label assignment - Classes 0 to 29

Label	Class	Move	Hands	Feet	Trunk	Fast
0	Walk	✓		✓		
1	Stand				✓	
2	Hand movements		✓			
3	Turn	✓		✓	✓	
4	Interact with use object		✓			
5	Arm movements		✓			
6	T pose		✓		✓	
7	Step	✓		✓	✓	
8	Backwards movement	✓		✓		
9	Raising body part				✓	
10	Look					
11	Touch object		✓			
12	Leg movements			✓		
13	Forward movement	✓		✓	✓	
14	Circular movement	✓		✓	✓	
15	Stretch				✓	
16	Jump	✓		✓		✓
17	Touching body parts		✓			
18	Sit			✓	✓	
19	Place something		✓			
20	Take/pick something up		✓			
21	Run	✓		✓		✓
22	Bend				✓	
23	Throw		✓			✓
24	Foot movements			✓		
25	A pose		✓		✓	
26	Stand up			✓	✓	
27	Lowering body part				✓	
28	Sideways movement			✓	✓	
29	Move up/down incline				✓	

Table 4.2: Pseudo-label assignment - Classes 30 to 59

Label	Class	Move	Hands	Feet	Trunk	Fast
30	Action with ball	✓		✓		
31	Kick			✓		✓
32	Gesture		✓			
33	Head movements					
34	Jog	✓		✓		
35	Grasp object		✓			
36	Waist movements			✓		
37	Lift something		✓			
38	Knee movements			✓		
39	Wave		✓			
40	Move something		✓			
41	Swing body part				✓	
42	Catch		✓			
43	Dance	✓	✓	✓	✓	
44	Lean				✓	
45	Greet		✓			
46	Poses				✓	
47	Touching face		✓			
48	Sports moves	✓	✓	✓	✓	✓
49	Exercise/training		✓	✓	✓	
50	Clean something		✓			
51	Punch		✓			✓
52	Squat			✓		
53	Scratch		✓	✓	✓	
54	Hop	✓		✓		
55	Play sport	✓	✓	✓	✓	✓
56	Stumble			✓		
57	Crossing limbs					
58	Perform	✓	✓	✓	✓	
59	Martial arts	✓	✓	✓	✓	✓

4.2.3 Training Regime and Inference

For the reconstruction pre-training, we used the L1 metric to measure the distance of the position of each key-point predicted by the decoder in 3D space to the position defined by the sequence of ground-truth poses. The reconstruction pre-training phase has a duration of 300 epochs. For the optimization method, we selected AdamW optimizer [33] with a learning rate of 5e-5. The mini-batch size was 64 samples.

For the contrastive learning pre-training we used the contrastive loss function, as defined by Equation 3.8, fixing the parameters $m_{pos} = 0$ and $m_{neg} = 1$. The number of epochs, the optimizer, learning rate and mini-batch size were kept the same as for the reconstruction pretraining, *i.e.*, 300 epochs, AdamW optimizer, learning rate of 5e-5 and 64 samples/batch.

For the fine-tuning phase, we experiment using focal loss [29]. Focal loss compensates for class unbalance by up-weighting the cross-entropy loss for inaccurate predictions. The models were trained for a maximum of 300 epochs. The experiment is subject to early stop to avoid overfitting. For the fine-tuning phase, we kept the AdamW with a learning rate of 5e-5 and 64 samples/batch. Each fine-tuning training epoch required approximately 5:36 to process 766 batches.

Regarding the computational effort for inference, the network requires approximately 49 seconds to process an epoch consisting of 287 batches, each containing 64 samples, totaling 18,368 samples. This means that the network takes 2.66 milliseconds to process an action sequence. The use of the model in a real-time application needs to take into account other variables, such as data preparation times, which are optimized here because a batch processes 64 samples in parallel, and the computational resources available. Under ideal conditions, it would be possible to use the model in a real-time application.

4.3 Results

To analyze the results attained by our methodology, we used the same metrics of [39]. Top-1 measures the accuracy of the most activated class. Top-5 measures if the correct class ranks among the top 5 highest-scoring predictions. Top-1-norm is the mean Top-1 accuracy across all classes. If the dataset is balanced, both Top-1 and Top-1-norm should have similar values. The Top-1-norm may give an indication of the class-specific

Table 4.3: Results - Top 5 (Focal loss).

	Variation	Accuracy (%)
BABEL 60	Dataset Benchmark	69.0
	ST-GCN	44.2
	2s-AGCN	67.8
	MST-GCN	70.3
	Ours (Reconstruction)	**70.4**
	Ours (Contrastive)	**70.9**
BABEL 120	Dataset Benchmark	59.0
	ST-GCN	28.6
	2s-AGCN	58.0
	MST-GCN	60.1
	Ours (Reconstruction)	**65.4**
	Ours (Contrastive)	**65.6**

bias in the model performance.

Since Top-5 accuracy accounts for labeling noise and ambiguity, it is considered as the best indicator to evaluate the model. We trained our model with focal loss to compare with current state-of-the-art skeleton-based action recognition methods. The results for Top-5 using focal loss can be seen in Table 4.3. Both our methods outperform all competitors by a small margin for the BABEL 60 dataset (0.1 p.p. and 0.6 p.p. for reconstruction and contrastive methods, respectively) and for a bigger margin for the BABEL 120 (5.3 p.p. and 5.5 p.p. for reconstruction and contrastive methods respectively).

Our method also presented significant results for the Top-1 metric when optimizing with focal loss. The reconstruction method could outperform all competitors for both BABEL 60 and BABEL 120 (0.1 p.p. and 1.5 p.p. respectively). The contrastive method, however, loses by 4 p.p to MST-GCN when tested against the BABEL 60 dataset. However, when tested against the BABEL 120 dataset, its performance is good, outperforming the best competitor by a margin of 3.7 p.p. The results for the Top-1 metric with focal loss optimization are summarized in the Table 4.4.

For the Top-1 metric with accuracy normalized by the number of samples, what we see is a good performance, but not enough to outperform all competitors. Our method is able to surpass or equal the ST-GCN and 2s-AGCN methods in both versions of the dataset, but it is below the MST-GCN, with a smaller margin (1.4 p.p) in the case of the BABEL 120 dataset, but higher (2.0 p.p.) in the case of the BABEL 60. The results for this metric are compiled in the Table 4.5.

Table 4.4: Results - Top 1 (Focal loss).

	Variation	Accuracy (%)
BABEL 60	Dataset Benchmark	34.0
	ST-GCN	24.2
	2s-AGCN	33.8
	MST-GCN	36.3
	Ours (Reconstruction)	**36.4**
	Ours (Contrastive)	34.9
BABEL 120	Dataset Benchmark	29.0
	ST-GCN	20.5
	2s-AGCN	27.9
	MST-GCN	29.9
	Ours (Reconstruction)	**31.4**
	Ours (Contrastive)	**33.6**

Table 4.5: Results - Top 1 (Focal loss - Normalized).

	Variation	Accuracy (%)
BABEL 60	Dataset Benchmark	30.0
	ST-GCN	14.4
	2s-AGCN	30.4
	MST-GCN	**35.4**
	Ours (Reconstruction)	30.3
	Ours (Contrastive)	33.4
BABEL 120	Dataset Benchmark	23.0
	ST-GCN	5.5
	2s-AGCN	26.2
	MST-GCN	**29.8**
	Ours (Reconstruction)	28.4
	Ours (Contrastive)	26.4

4.4 Hyperparameter Sensitivity Testing

A systematic search for hyperparameters was not carried out in this work. However, as a way of evaluating opportunities for optimization, some tests were performed in order to evaluate the sensitivity of the model to variations in the hyperparameters. The tests were made to help define only the hyperparameters of the fine-tuning phase. The variations made had no effect on the pre-training phases. This effort was restricted to the fine-tuning phase to avoid an exaggerated expansion of the effort in this search, given that there are many possible variations of hyperparameters. The results can be seen in the Table 4.7.

Table 4.6: Parameters evaluation.

	Variation	Top-1	Top-1-Norm	Top-5
BABEL 60	Dropout = 0.2	32.8	31.9	67.2
	Dropout = 0.5	30.5	30.7	63.6
	Optim = SGD	32.6	**32.0**	68.6
	LR = 1e-5	33.8	31.4	65.9
	LR = 2e-4	30.6	30.8	63.5
	Weight decay = 1e-3	31.9	31.5	67.4
	Selected	**36.6**	30.3	**70.4**
BABEL 120	Dropout = 0.2	28.3	25.2	63.7
	Dropout = 0.5	26.9	26.3	56.8
	Optim = SGD	24.0	27.2	56.7
	LR = 1e-5	28.5	26.4	57.2
	LR = 2e-4	22.8	25.8	49.5
	Weight decay = 1e-3	26.9	27.6	63.9
	Selected	**31.5**	**28.4**	**65.4**

What can be observed is, firstly, that increasing the dropout did not result in performance gains, which justified keeping the dropout fixed at 0.1. In terms of the learning rate, we tested values above and below 5e-5, keeping the AdamW optimizer, in which case we could observe that increasing the learning rate resulted in worse results. The test performed with a lower learning rate also resulted in lower accuracy, but in this case, it is important to note that the training was limited to 300 epochs. Increasing the weight decay regularization term did not deliver good results, and therefore the value of 1-e4 was kept.

Regarding the optimizer, we observed that the use of SGD instead of AdamW showed promising results. In the case of BABEL 60, the use of this optimizer surpassed the best result obtained with AdamW in the Top-1-Norm metric by 1.7 p.p. In BABEL 120 dataset, however, the SGD optimizer was overcome by AdamW by 1.2 p.p.

Unlike the other hyperparameters, in this case, a systematic search was performed for the configuration parameter of the focal loss function. As explained in Section 2.5, the gamma parameter determines the relative weight that will be applied to the error relative to samples of each class, together with the number of samples available for each one of them. The tested values for this hyperparameter and the results can be seen in Table 4.7. Looking at the table, we can see that this parameter has a significant effect on the Top-5 metric, with the accuracy increasing consistently with the increase in the parameter value up to the limit of gamma = 20. Above this value, the model performance degrades rapidly. In the case of the Top-1 parameter, an oscillation around 36.5% is observed, with the best result occurring for gamma = 5. Also, in the case of the Top-1 metric, the performance degrades for values above gamma = 20. Due to the fact that it is the best combination

Table 4.7: Results - Focal loss tunning.

	Gamma	Top5 (%)	Top1 (%)
BABEL 60	1	68.4	36.1
	2	68.5	36.3
	5	69.3	**36.9**
	10	69.8	36.4
	20	**70.4**	36.4
	50	53.9	21.5

of values, the value selected for this parameter was gamma = 20.

4.5 Ablation Study

In the ablation study, we selectively examine parts of our architecture in order to analyze the impact each of them has on the results. First, to evaluate how the pre-training contributes to the results, we execute experiments training directly for action recognition, without a pre-training phase. Then we explore the effect of the Transformer, changing the size of the stack of Transformer blocks. Finally, we remove the multi-scale mechanism to verify its effect. The results are shown in Table 4.8.

It can be verified from the results that, especially in the metrics taken as a reference, *i.e.*, the Top-5 and Top-1 accuracy when trained with focal loss, all components contribute to the result. The component with the greatest impact on key metrics was the use of the multi-scale engine. This mechanism alone contributed to an increase of 4.2 p.p. in Top-5 accuracy, both for the BABEL 60 and BABEL 120 datasets when trained with focal loss. Also, in the case of the Top-1 matrix, the result shows a gain of 3.2 p.p. for BABEL 60 and 2.4 p.p. for BABEL 120.

The use of pre-training also showed a relevant effect on the final result, given that for the main metrics, the result was consistent in both versions of the dataset. In the case of BABEL 60, it is observed is an increase of 0.7 p.p. in the Top-5, and 1.9 p.p. in the case of the Top-1 accuracy. The use of pre-training had no observable effect on the normalized Top-1 metric. In the case of BABEL 120, pre-training was responsible for an increase of 1.9 p.p. in performance measured by the Top-5 metric and 1.6 p.p in the performance measured by the Top-1 metric.

Table 4.8: Ablation study.

	Variation	Top-1	Top-1-Norm	Top-5
BABEL 60	No pretrain	34.5	30.3	69.7
	Transformer (1 layer)	34.2	**31.4**	68.2
	Transformer (2 layer)	33.8	31.2	69.2
	No multi-scale mechanism	33.2	29.4	66.2
	Complete model	**36.4**	30.3	**70.4**
BABEL 120	No pretrain	29.8	25.3	63.5
	Transformer (1 layer)	29.9	26.8	61.9
	Transformer (2 layer)	30.3	**28.6**	64.8
	No multi-scale mechanism	29.0	24.7	61.2
	Complete model	**31.4**	28.4	**65.4**

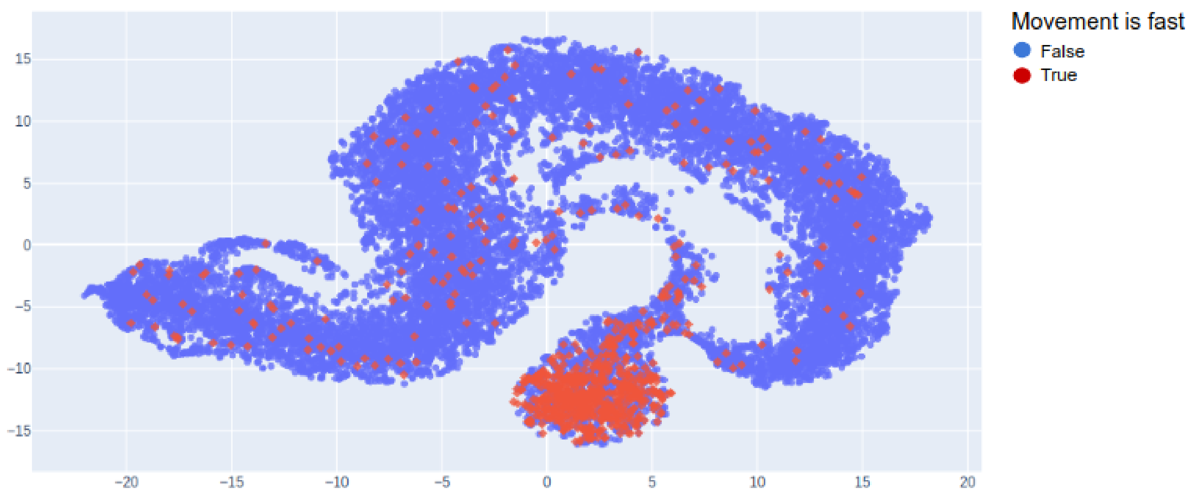
4.6 Discussion

From the results of the ablation study and experiments, some issues deserve a more detailed evaluation. These findings may indicate opportunities for improvement in the methodology or points of attention. A brief discussion will ensue in order to clarify some observations.

As verified in the ablation tests, both forms of pre-training produced positive effects on the final result, contributing to better performance. The pre-training step is responsible for a gain in the Top-5 metric (in this discussion, we will always be referring to the metrics obtained by models trained with focal loss). The gain, however, was greater in the case of pre-training performed using contrastive learning, where we see a 0.5 p.p. performance increase compared to the model generated from pre-training by reconstruction.

This increase can be explained by the fact that, through the definition of pseudo-labels, the objective was to give the network the ability to disambiguate very similar movements but with a striking feature. A clear example of this can be seen in the Figure 4.3. The images, show how the action representation space was organized when considering only the fraction of the embedding associated with a pseudo-label. To facilitate the visualization, the representation space had its dimensionality reduced by the TSNE algorithm.

What was evidenced in Figure 4.3 is that the embeddings associated with classes that have fast movements (indicated in red) were well-differentiated in relation to slow movements (indicated in blue). When we compare two classes that have similar pose sequences, but that can be disambiguated by the speed of movements, such as the walk and run classes, it is evident that the use of the pseudo-label contributed strongly to facilitating the correct classification of these two classes for the network.



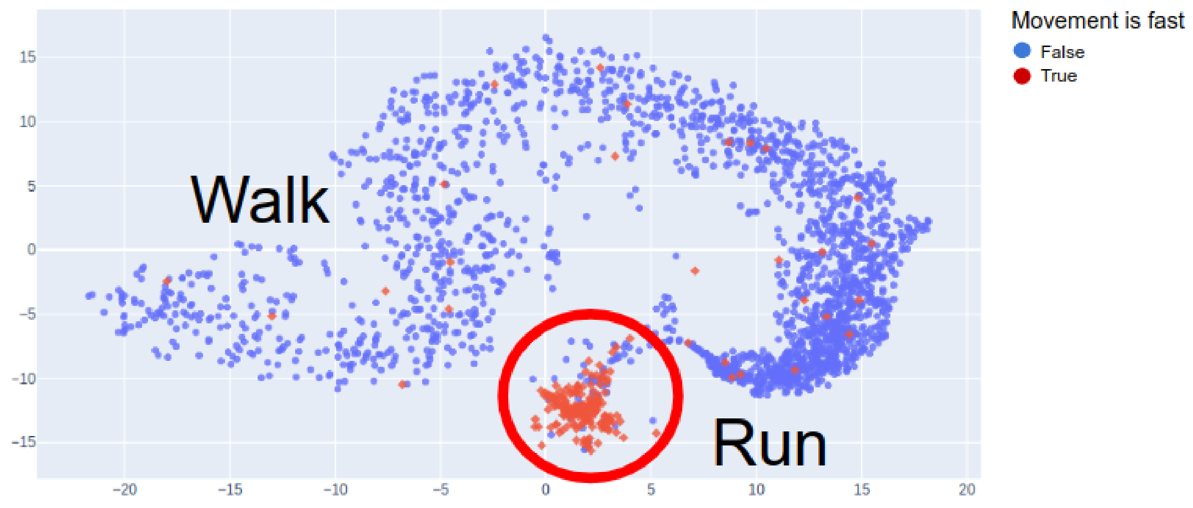
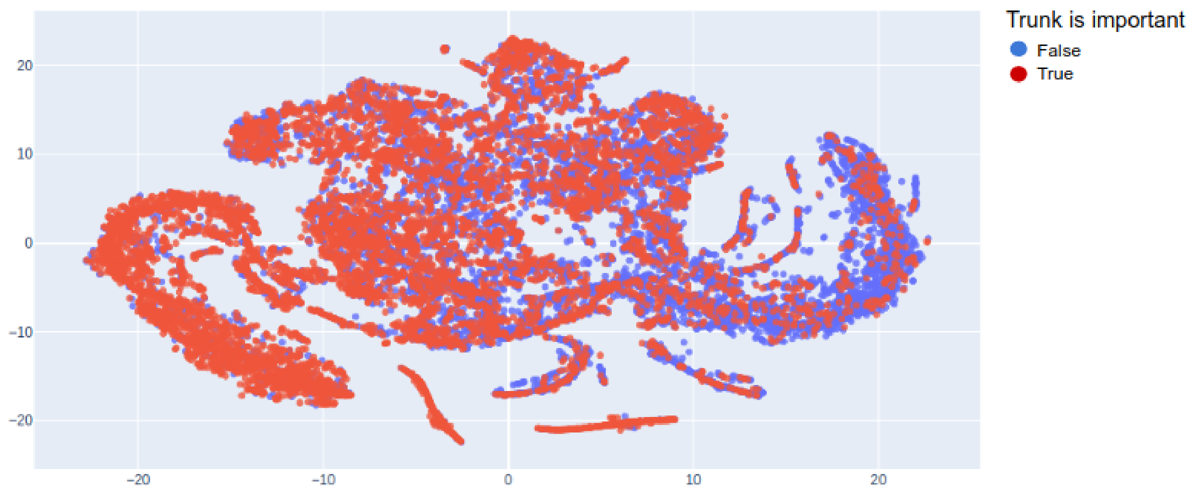


Figure 4.3: Visualization of partition vectors - pseudo-label “Fast movement”. At the top we see all the samples organized by the vector associated with the pseudo-label “Fast movement”. Below we see only the samples associated with the actions “walk” and “run”.

In the case of the pseudo-label which aims to separate actions where trunk movement is relevant, the separation was not so clear. As we can see in Figure 4.4, after the reduction of dimensionality, what is observed is that, although there is a tendency favoring separation (blue points becoming more to the right and blue points becoming more to the left), there is no well-defined border between the two spaces, indicating that this pseudo-label did not make a major contribution to disambiguation. Although some effect is present, as can be seen in the comparison of the embeddings of the incline (exercise) and wave actions. In both cases, the movement with the hands is similar, but the position of the trunk is important to disambiguate.

However, despite the consistent result in the Top-5 metric, we see that in the case of the Top-1 metric the result is inconsistent when comparing the BABEL 60 and BABEL 120 datasets. Despite the strong result for the BABEL 120, outperforming the



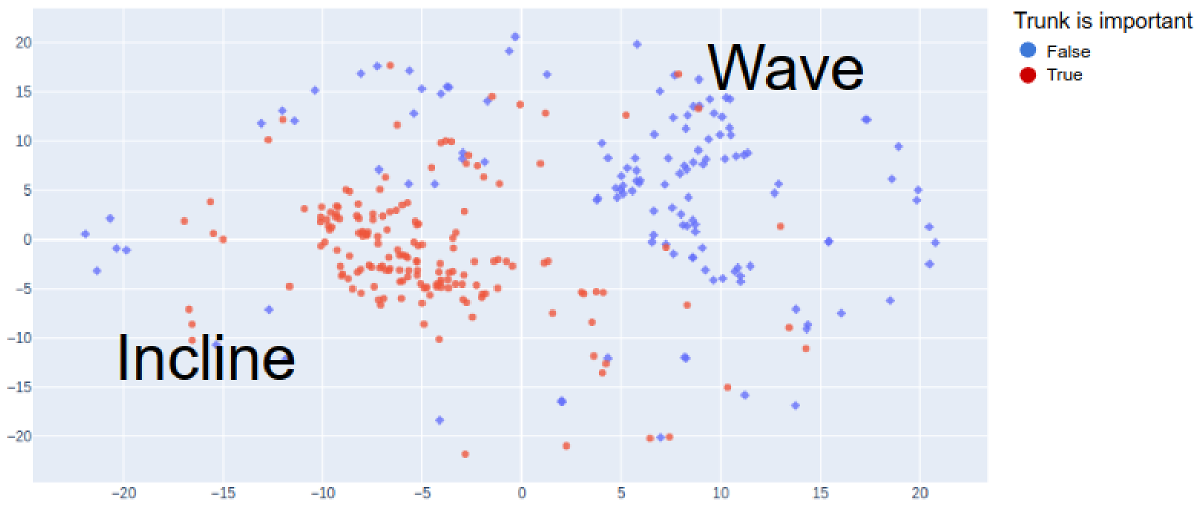


Figure 4.4: Visualization of partition vectors - pseudo-label ”Trunk is important”. At the top we see all the samples organized by the vector associated with the pseudo-label ”Trunk is important”. Below we see the samples associated with the “wave” and “incline” actions.

other competitors by more than 2 p.p., in BABEL 60 it loses to both the version pre-trained by the reconstruction method and the competitor MST-GCN. It is also possible to observe this behavior in the case of the normalized focal loss metric. The selection of pre-training form, therefore, must be made according to the objective to be achieved. For a situation where the objective is to better understand the context, accepting that multiple classes can be accepted as an answer, pre-training using contrasting learning is the best option, as it delivers better results for the Top-5 metric. If the objective of the network is to obtain an accurate answer, (*i.e.*, the most precise among ambiguous options), the pre-training by reconstruction presented the best result in general.

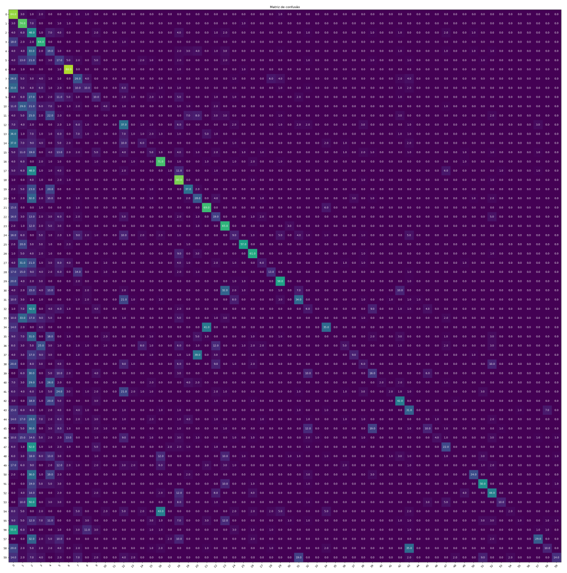
4.6.1 Dealing with Class Imbalance

As defined throughout the work, we have a special concern about the way the model behaves in a long-tailed dataset. The way chosen to tackle this problem was through the use of focal loss. In this section, we seek to analyze in more detail the effect of this choice and highlight the reason why we pay special attention to metrics that use this loss function. To perform this analysis, we can use as a reference the results organized in confusion matrices. The confusion matrix of a model trained using cross-entropy, and the confusion matrix of a model trained using focal loss are illustrated in Figure 4.5.

When training the model with cross-entropy, we observed that there is a significant bias toward classifying in the model output classes that are represented in the first columns of the matrix, that is, classes with a greater number of samples. This ends up favoring this model if the analysis performed was only Top-1 accuracy, directly hitting the label defined for the dataset. It is easy to see however, that this is not a fair approach, given that the model would not be learning to analyze the action itself but would only be learning to exploit the dataset's imbalance in its favor.

A very different situation is what we observe in the case of the model trained with focal loss. The bias towards classes with the highest number of samples is not present. The false positives in the first columns are in balance with what was observed in columns associated with classes with fewer samples. The cases of confusion are the result not of class imbalance but of movement similarity itself. This fact becomes clearer when we observe that in the confusion matrix of the model trained using cross-entropy, there are practically no false positives above the main diagonal of the matrix. In the case of the matrix of the model trained with focal loss, there are several occurrences of false positives above the main diagonal.

What we could observe in the confusion matrices of both models is even more evident when we analyze the difference between the two matrices, illustrated in Figure 4.6. To obtain the difference between the two matrices, we simply perform the element-by-element subtraction of each cell of the matrix. Dark colors mean that the value of that cell is smaller in the matrix of the model trained with focal loss, and light colors mean that the cell value is larger in the matrix trained with focal loss.


(a) Trained using cross-entropy.

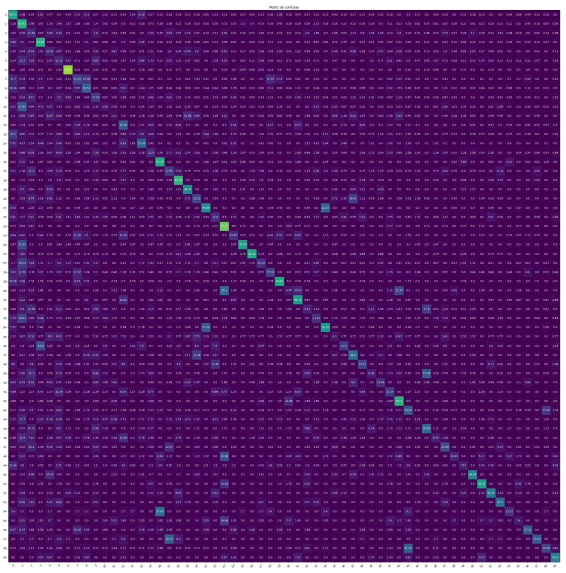

(b) Trained using focal loss.

Figure 4.5: Confusion matrix comparison. The confusion matrix of a model trained with cross-entropy (a) shows that, besides it has higher accuracy, the is biased in favor of more represented classes. This fact is evidenced by the large number of false positives observed in the first columns (which represent the best-represented classes). In the model trained using focal loss (b) this problem is not present.

4.6.2 Dealing with Ambiguity

Another important aspect of this work is its ability to deal with ambiguity. We saw in the quantitative results (organized in Table 4.3) that we performed better on the Top-5 metric when trained with focal loss, which is the one that best captures the model's ability to deal with ambiguities. In this section, we will make a qualitative analysis, seeking a better understanding of the result obtained, why using focal loss turned out to be important when dealing with ambiguity, and what this means in practical terms.

For that, we observed the accumulated activation profile of selected class samples when presented to models trained with cross-entropy compared to models trained with focal loss. The accumulated activation profile is generated from all samples of a selected class inputted in a trained model. The activation levels output by each sample are accumulated, resulting in a histogram. Through the histogram, it is possible to visualize how the model is interpreting a certain class and if the activation level of similar classes is well represented.

The first example is shown in Figure 4.7 where we can see the activation profile for two networks, one trained with cross-entropy and the other trained with focal loss, when all samples of the 'touch object' class are inputted into the model. What we can see in the example is that for the cross-entropy trained network, the 'touch object' class stands

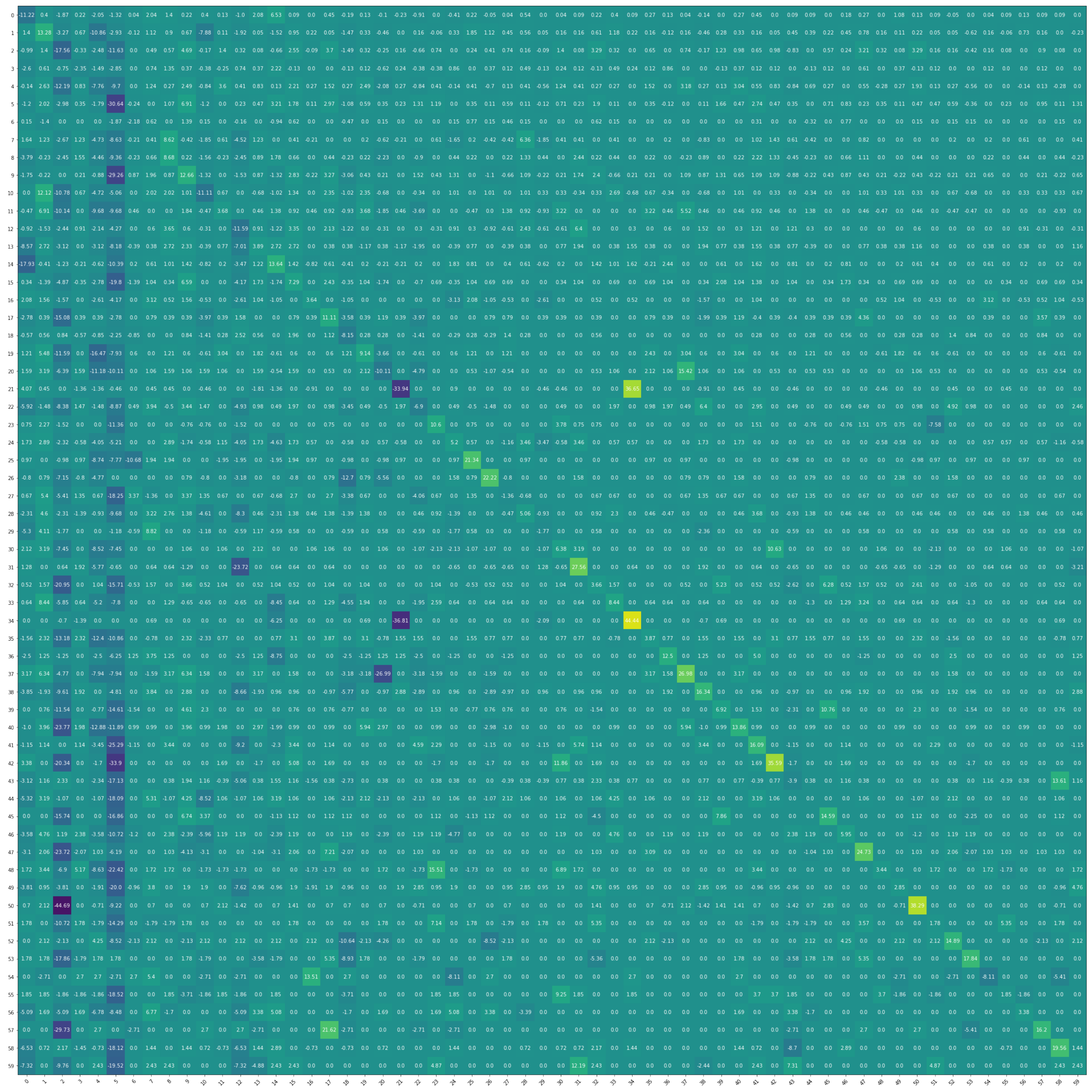
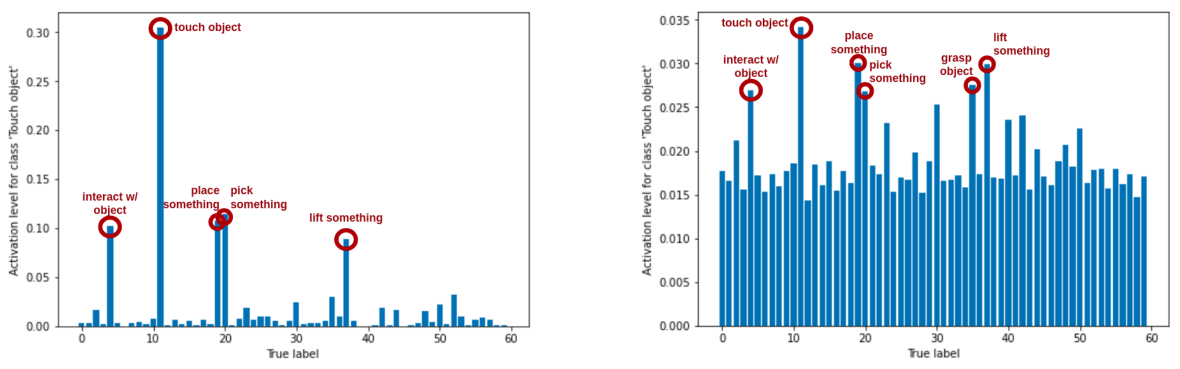


Figure 4.6: Matrix highlighting the differences between the confusion matrix of a model trained using cross-entropy and a model trained using focal loss.

out with a large peak, with similar classes having clear activation levels but significantly less intense.

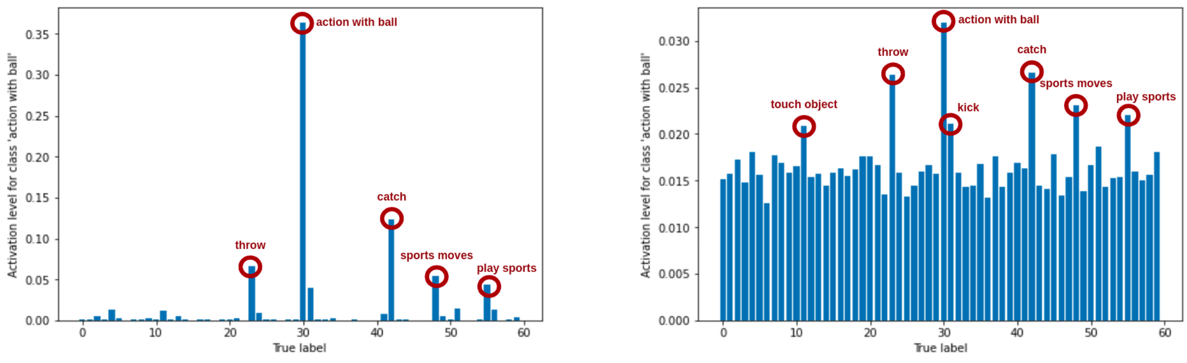
On the one hand, if we consider this difference in semantic terms, we can conclude that it is not justified. The meaning of touching an object (represented in class 11, highlighted in the image) is very similar to the meaning of interacting with an object (represented in class 4, also highlighted in the image). The level of certainty presented by the network does not match the real meaning of the available options. The same can be said for other classes like ‘place something’, ‘pick something’, and ‘lift something’.

On the other hand, when we observe the activation profile for the same set of classes when trained with focal loss, we still observe a peak in the ‘touch object’ class, but the



(a) Cross-entropy (b) Focal

Figure 4.7: Activation profile for class 'touch object'.



(a) Cross-entropy (b) Focal

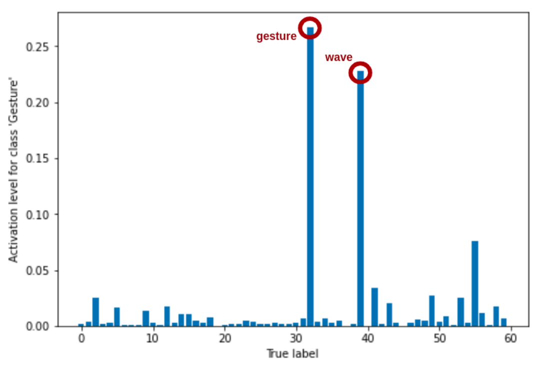
Figure 4.8: Activation profile for class 'action with ball'.

other classes with similar semantics to the studied class also show relevant activation, in balance to the activation of the main peak. We also observed that the set of activated classes is better representing the action itself, with the inclusion in the set of highly activated classes actions like 'grasp object' that presented very weak values in model trained using cross-entropy.

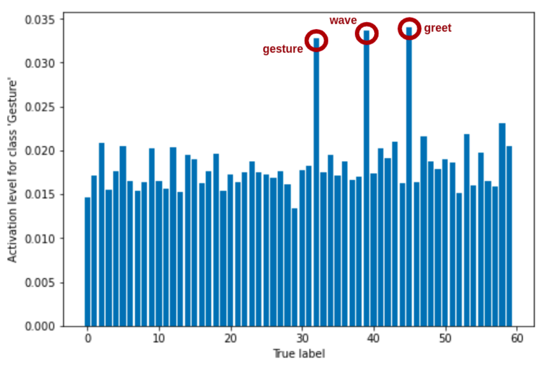
A similar situation can be observed in the activation profile of samples of the class 'action with ball', illustrated in Figure 4.8. Again in the case of models trained using cross-entropy, the network presented a level of certainty in the choice of a class that is not justified when analyzing the semantics of the available options. Classes like 'sport moves' and 'play sports' receive activation but at a level lower than reasonable. Actions such as 'catch' and 'throw' are also present in the activation profile but with much lower values than those obtained for the main peak.

In the activation profile of the model trained using focal loss, the levels are more balanced, better representing the expected equilibrium among classes that are closely related to each other, including classes such as 'kick' and 'touch object' that was not present in the activation profile of the model trained using cross-entropy.

One last example can be seen in the profiles obtained for the samples of the 'gesture'



(a) Cross-entropy.



(b) Focal

Figure 4.9: Activation profile for class 'gesture'.

class, illustrated in Figure 4.9. In the case of the model trained using cross-entropy, we see only two peaks, the first for the 'gesture' class and the second for the 'wave' class. If we analyze only these two classes, we observe an adequate balance between them. However, when we train the network using focal loss, the 'greet' class, which previously received weak activation, starts to stand out, even surpassing the class designated by the dataset. The results obtained for the focal loss better describes the set of samples, as it shows a balance between three classes that have similar meanings.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

The search for robust solutions to treat the problem of action recognition had consistent advances in increasing the accuracy in datasets primarily dedicated to this task. However, when researchers test these on real cases, they still find it challenging to deliver the necessary reliability, indicating that this line of research still has a long way to go to reach maturity.

In this work, we seek to address two critical aspects that are generally not addressed: class unbalance and the ambiguity inherent to datasets built from the observation of everyday life. To achieve this goal, we propose a new learning framework based on multiscale features and the context of motions. Additionally, the learning process was subject to a training regime that involved pre-training, seeking to generate robust representations of the movement in the internal layers of the network to boost classification.

Considering that the selected dataset is highly unbalanced and sufferers from a high degree of ambiguity, the most appropriate indicator is the accuracy among the top 5 predictions. Our approach beat our competitors in both BABEL 60 and BABEL 120 in the Top-5 (focal) accuracy metric, the most relevant from our problem definition. We also achieved a better result in Top-1 (focal) accuracy when trained with focal loss.

Using the focal loss function in the fine-tuning phase proved very important. Its use made it possible to deal with the problem of class imbalance and made an important contribution to the problem of ambiguity. With the help of focal loss, as demonstrated qualitatively through the analysis of the activation profiles of the network, the outputs better reflect ambiguous situations, unlike what we observed in a network trained using cross-entropy. This good effect in the treatment of ambiguity directly impacts our primary metric, which is the Top-5 accuracy.

The field of study of action recognition is very dynamic. Undoubtedly, we will see advances over the next few years, especially considering the momentum of technologies based on deep learning. The development of this area will therefore continue to be heavily dependent on the construction of large datasets. This dependency will be true both in the academic-only domain and in the portability of technology to real world applications. Addressing class imbalance and ambiguity problems helps the community to work with datasets with these characteristics and to approach a technology that evolves rapidly from its application in everyday issues.

5.2 Future Work

Recent years have seen a rapid development of various techniques in tasks related to motion analysis and in machine learning in general. This research is far from having exhausted all opportunities in the field. Some even came to be the object of preliminary tests, but because they initially did not have the desired result, they were passed over for other choices that obtained more effective results. In future work, those options can be revisited and better investigated.

The first point to be explored is feature engineering. Our methodology uses the key-point positions enriched with the velocity. However, human movement is the result of the articulation of limbs, which makes the angles of the joints powerful pose descriptors. We see potential in adding the calculated angles around a joint to input data, especially through angle descriptors that do not have discontinuity problems.

There are opportunities in the two main components of the architecture. For example, systematic research around aggregation functions will likely improve the down-sampling layer. The investigation can be done by taking advantage of convolution strategies in skeletal graphs from more recent works in the action recognition literature and seeking to identify the advantages and disadvantages of each one of them. The way our model's architecture is structured allows for an exchange of these functions with minor adaptations.

We used Transformers in its most straightforward formulation, which suggests that searching for more tuned arrangements for our problem could improve the results obtained. Something interesting to note is that when applied in NLP, Transformers deal with word tokens. If we observe the embedding vectors of a set of words of a sentence, we will see that their position in the embedding space is distant from another. Our case is very different since the sequence of vectors represents a continuous motion. That results in a set of vectors that are positioned next to each other in the embedding space, particularly in adjacent poses. This attribute allows applying a strategy to reduce the number of tokens presented to the Transformer by selecting key poses.

We see the potential for improvement of the training regime in the case of learning by contrast. Despite having considered the strategy of using pseudo-labels successful, other arrangements regarding the use of partitions can be made. The number of partitions could be increased, seeking to directly encode other features such as circular movements, periodic movements, or interactions with body parts (*e.g.*, hand-head interactions). Moreover, because it is a technique used in a lot of research, there is a field of opportunity to refine our approach using techniques that have been successful in other areas.

In addition to the two pre-training forms used in this work, other strategies could be used. For example, the BABEL dataset was originally designed to link text and movement. This correlation between the description of the sequence and the actions that make up the sequence could be used in line with one of the objectives we proposed, *i.e.*, to generate rich and robust representations of the movement in the network's hidden layers.

Bibliography

[1] Aviad Aberdam, Ron Litman, Shahar Tsiper, Oron Anschel, Ron Slossberg, Shai Mazor, R. Manmatha, and Pietro Perona. Sequence-to-sequence contrastive learning for text recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15302–15312, June 2021.

[2] Yuanhao Cai, Zhicheng Wang, Zhengxiong Luo, Binyi Yin, Angang Du, Haoqian Wang, Xiangyu Zhang, Xinyu Zhou, Erjin Zhou, and Jian Sun. Learning delicate local representations for multi-person pose estimation. In *European Conference on Computer Vision*, pages 455–472. Springer, 2020.

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[5] Zhan Chen, Sicheng Li, Bing Yang, Qinghan Li, and Hong Liu. Multi-scale spatial temporal graph convolutional network for skeleton-based action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1113–1122, 2021.

[6] Huaining Cheng and Soon M Chung. Orthogonal moment-based descriptors for pose shape query on 3d point cloud patches. *Pattern Recognition*, 52:397–409, 2016.

[7] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 183–192, 2020.

[8] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.

[9] Claudio Marcio de Souza Vicente, Erickson R Nascimento, Luiz Eduardo C Emery, Cristiano Arruda G Flor, Thales Vieira, and Leonardo B Oliveira. High performance

moves recognition and sequence segmentation based on key poses filtering. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[13] Haodong Duan, Yue Zhao, Kai Chen, Dian Shao, Dahua Lin, and Bo Dai. Revisiting skeleton-based action recognition. *arXiv preprint arXiv:2104.13586*, 2021.

[14] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints, 2015.

[15] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr. Res2net: A new multi-scale backbone architecture. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[16] Rohan Ghosh, Anupam Gupta, Andrei Nakagawa, Alcimar Soares, and Nitish Thakor. Spatiotemporal filtering for event-based action recognition. *arXiv preprint arXiv:1903.07067*, 2019.

[17] Tianyu Guo, Hong Liu, Zhan Chen, Mengyuan Liu, Tao Wang, and Runwei Ding. Contrastive learning from extremely augmented skeleton sequences for self-supervised action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 762–770, 2022.

[18] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2017.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[20] Zhuolin Jiang, Viktor Rozgic, and Sancar Adali. Learning spatiotemporal features for infrared action recognition with 3d convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 115–123, 2017.

[21] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[22] Muhammad Attique Khan, Kashif Javed, Sajid Ali Khan, Tanzila Saba, Usman Habib, Junaid Ali Khan, and Aaqif Afzaal Abbasi. Human action recognition using fusion of multiview and deep features: an application to video surveillance. *Multimedia tools and applications*, pages 1–27, 2020.

[23] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[24] Youngwook Kim and Taesup Moon. Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks. *IEEE geoscience and remote sensing letters*, 13(1):8–12, 2015.

[25] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

[26] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011.

[27] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3595–3603, 2019.

[28] Dawei Liang and Edison Thomaz. Audio-based activities of daily living (ADL) recognition with large-scale acoustic embeddings from online videos. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(1):1–18, mar 2019.

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[30] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C Kot. NTU RGB+D 120: A large-scale benchmark for 3d human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2684–2701, 2020.

[31] Jun Liu, Amir Shahroudy, Dong Xu, Alex C Kot, and Gang Wang. Skeleton-based action recognition using spatio-temporal lstm network with trust gates. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):3007–3021, 2017.

[32] Rex Liu, Albara Ah Ramli, Huanle Zhang, Erik Henricson, and Xin Liu. An overview of human activity recognition using wearable sensors: Healthcare and artificial intelligence. *arXiv preprint arXiv:2103.15990*, 2021.

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[34] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019.

[35] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. Amass: Archive of motion capture as surface shapes. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2019.

[36] Vittorio Mazzia, Simone Angarano, Francesco Salvetti, Federico Angelini, and Marcello Chiaberge. Action transformer: A self-attention model for short-time human action recognition. *arXiv preprint arXiv:2107.00606*, 2021.

[37] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. Spatial temporal transformer network for skeleton-based action recognition. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part III*, pages 694–701. Springer, 2021.

[38] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[39] Abhinanda R. Punnakkal, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra Quiros-Ramirez, and Michael J. Black. BABEL: Bodies, action and behavior with english labels. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 722–731, Jun. 2021.

[40] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[41] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. NTU RGB+D: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016.

[42] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12026–12035, 2019.

[43] Ankit Singh, Omprakash Chakraborty, Ashutosh Varshney, Rameswar Panda, Rogerio Feris, Kate Saenko, and Abir Das. Semi-supervised action recognition with temporal contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10389–10399, 2021.

[44] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017.

[47] Antonio W Vieira, Erickson R Nascimento, Gabriel L Oliveira, Zicheng Liu, and Mario FM Campos. On the improvement of human action recognition from depth map sequences using space–time occupancy patterns. *Pattern Recognition Letters*, 36:221–227, 2014.

[48] Fei Wang, Yunpeng Song, Jimuyang Zhang, Jinsong Han, and Dong Huang. Temporal unet: Sample level human action recognition using wifi, 2019.

[49] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3024–3033, June 2021.

[50] Yancheng Wang, Yang Xiao, Fu Xiong, Wenxiang Jiang, Zhiguo Cao, Joey Tianyi Zhou, and Junsong Yuan. 3dv: 3d dynamic voxel for action recognition in depth video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 511–520, 2020.

[51] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2006.03677*, 2020.

[52] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[53] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *arXiv preprint arXiv:2204.12484*, 2022.

[54] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition, 2018.

[55] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022.

[56] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *6th international conference on mobile computing, applications and services*, pages 197–205. IEEE, 2014.

[57] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.