



Degree Project in Technology

Second cycle, 30 credits

Skeleton-based Football Referee Action Recognition

YIZHOU XU

Skeleton-based Football Referee Action Recognition

YIZHOU XU

Master's Programme, Machine Learning, 120 credits

Date: June 19, 2024

Supervisors: Josephine Sullivan, Tiesheng Wang

Examiner: Atsuto Maki

School of Electrical Engineering and Computer Science

Host company: Tracab

Swedish title: Skelettbaserad igenkänning av fotbollsdomares handlingar

Abstract

Referee signals are crucial for understanding football games. While existing tracking techniques provide accurate skeleton data for players and referees, automatic recognition of referee actions remains largely unexplored. This work aims to develop self-supervised and semi-supervised learning methods for referee action recognition from skeleton data, leveraging abundant unlabeled data, and addressing the challenge of labeled data scarcity.

We propose a pre-training and fine-tuning pipeline based on transformer-based masked autoencoders for this task, with granularly differentiated approaches for main referees and assistant referees. For assistant referees, whose actions are more static, we introduce a frame-level model. The model is initially pre-trained on unlabeled frames and subsequently fine-tuned on task-specific data to perform action recognition. For main referees, we pre-train a sequence-level model to capture more contextual information. A novel multi-task pre-training objective is proposed, combining motion prediction and data2vec, where the prediction target is latent contextualized representations calculated by a teacher model. Furthermore, we fine-tune the model with a sequence labeling task supervised by a binary classification model, eliminating the need for frame-level annotations and utilizing the labels more efficiently. Additionally, we implement strategies to integrate additional inputs during fine-tuning, such as the ball position, to provide necessary details for action recognition.

Experiments show that our proposed methods excel in both the evaluation set and real-game scenarios. Our frame-level model achieves an accuracy of 99.46% on the test set and an F1 score of 0.92 on a real game. For the more challenging task of main referee action recognition, our sequence-level model achieves an accuracy of 91.88% on the test set and an F1 score of 0.79 on a real game.

Keywords

Skeleton-based action recognition, Referee action recognition, Masked autoencoders, Self-supervised learning, Semi-supervised learning

Sammanfattning

Domarsignaler är avgörande för att förstå fotbollsmatcher. Befintliga spårningstekniker ger exakt skelettdata för spelare och domare, men automatisk igenkänning av domarens handlingar fortfarande till stor del utforskad. Syftet med detta arbete är att utveckla självövervakade och semiövervakade inlärningsmetoder för igenkänning av domarens handlingar från skelettdata, utnyttja en stor mängd ej annoterad data och hantera utmaningen som uppstår när annoteringar fattas.

Vi föreslår en förtränings och finjusteringspipeline baserad på transformerbaserade maskerade autoencodernätverk med särskilda tillvägagångssätt för huvuddomare och assisterande domare. För assisterande domare, vars handlingar är mer statiska, introducerar vi en modell på bildnivån. Modellen förtränas initialt på ej annoterade bilder och finjusteras därefter på en uppgiftsspecifik datamängd för att utföra handlingsigenkänning. För huvuddomare förtränar vi en modell på sekvensnivå för att fånga mer kontextuell information. Ett nytt multitask-förträningsmål föreslås, som kombinerar rörelseförutsägelse och data2vec där målen för förutsägelser är latent kontextualiserade representationer beräknade av en läramodell. Utöver detta finjusterar vi modellen med ett sekvensannoteringsproblem som övervakas av en binär klassificeringsmodell, vilket eliminerar behovet av annoteringar av enskilda bilder och använder annoteringar mer effektivt. Dessutom implementerar vi strategier för att integrera ytterligare indata under finjustering, såsom bollpositionen, för att ge nödvändiga detaljer för igenkänning av handlingar.

Experiment visar att våra föreslagna metoder utmärker sig både i utvärderingsuppsättningen och i verkliga matchsituationer. Vår bildnivåmodell presterar utmärkt för assisterande domare, och uppnår en noggrannhet på 99,46% på testuppsättningen och en F1-poäng på 0,92 i en verklig match. För den mer utmanande uppgiften att känna igen huvuddomarens handlingar visar vår sekvensnivåmodell också tillfredsställande prestanda, och uppnår en noggrannhet på 91,88% på testuppsättningen och en F1-poäng på 0,79 i en verklig match.

Nyckelord

Skelettbaserad handlingsigenkänning, Igenkänning av domares handlingar, Maskerad autoencoder, Självövervakad inläring, Semiövervakad inläring

Acknowledgments

I would like to begin by thanking Tracab for providing such an interesting thesis project. As a football fan and a football referee, I thoroughly enjoyed my four-month journey, with each day filled with motivation and enthusiasm.

I would like to thank my supervisors, Tiesheng and Romina from Tracab, for their careful guidance each week and their valuable, insightful suggestions throughout this work.

I would also like to thank everyone on the team at Tracab. I greatly appreciate the relaxed, positive, and energetic atmosphere within the team, which provided me with additional motivation.

I also want to thank my KTH supervisor, Josephine, for her guidance and suggestions during the project.

Finally, I would like to thank my parents for their continued support. May everything go smoothly in the future.

Stockholm, June 2024

Yizhou Xu

Contents

1	Introduction	1
1.1	Problem	2
1.1.1	Problem Definition	2
1.1.2	Scientific and Engineering Challenges	4
1.1.2.1	Scientific Challenges	4
1.1.2.2	Engineering Challenges	4
1.2	Purpose	5
1.3	Research Content	5
1.4	Delimitations	6
1.5	Structure of the Thesis	7
2	Background	9
2.1	Skeleton-based Action Recognition	9
2.1.1	ST-GCN	10
2.1.2	MAMP	13
2.1.3	data2vec	15
2.1.4	Other Methods	16
2.1.4.1	Supervised Learning Methods	16
2.1.4.2	Unsupervised Learning Methods	18
2.1.4.3	Few-shot Learning Methods	20
2.2	Data Available	20
2.3	Football Referee Actions	21
2.3.1	Standardized Referee Actions	22
2.3.2	Non-standardized Referee Actions	23
2.4	Deep Learning for Action Understanding in Football	24
2.5	Summary	24
3	Frame-level Assistant Referee Action Recognition	27
3.1	Overview	27

3.2	Method	29
3.2.1	Pre-training	30
3.2.1.1	Model Architecture	30
3.2.1.2	Joint Embedding	30
3.2.1.3	Mask Strategy	31
3.2.1.4	Pre-training Task	32
3.2.2	Fine-tuning for Two-stage Recognition	32
3.2.2.1	Two-stage Recognition	32
3.2.2.2	Fine-tuning Method	33
3.2.3	Post-processing	34
3.3	Experiments and Results	35
3.3.1	Data Preparation	35
3.3.1.1	Pre-training Data	35
3.3.1.2	Classification Dataset for Stage 1	35
3.3.1.3	Classification Dataset for Stage 2	37
3.3.2	Pre-training	38
3.3.2.1	Network Architecture	38
3.3.2.2	Pre-training Details	38
3.3.3	Fine-tuning	38
3.3.3.1	Fine-tuning Details	38
3.3.3.2	Experiment Results	39
3.3.4	Evaluation on a Real Game	41
3.3.5	Case Study	43
3.3.6	Summary	45
4	Sequence-level Referee Action Recognition	47
4.1	Overview	47
4.2	Method	50
4.2.1	Pre-training	50
4.2.1.1	Model Architecture	50
4.2.1.2	Data Pre-processing	51
4.2.1.3	Data Augmentation	53
4.2.1.4	Joint Embedding	53
4.2.1.5	Motion-aware Tube Masking	54
4.2.1.6	Multi-task Pre-training	55
4.2.2	Linear Evaluation	57
4.2.3	Fine-tuning for Referee Action Recognition	58
4.2.3.1	Targeting Action Types	58
4.2.3.2	Integration of Additional Information	59

4.2.3.3	Fine-tuning with Sequence Classification Task	61
4.2.3.4	Fine-tuning with Sequence Labeling Task	62
4.2.4	Post-processing	64
4.3	Experiments and Results	65
4.3.1	Data Preparation	65
4.3.1.1	Pre-training Data	65
4.3.1.2	Dataset for Linear Evaluation	66
4.3.1.3	Sequence Classification Dataset for Fine-tuning	67
4.3.2	Pre-training	68
4.3.2.1	Network Architecture	68
4.3.2.2	Pre-training Details	69
4.3.2.3	Main Results	69
4.3.2.4	Ablation Study	70
4.3.3	Fine-tuning for Referee Action Recognition	71
4.3.3.1	Fine-tuning Details	71
4.3.3.2	Results with Sequence Classification Task	71
4.3.3.3	Results with Sequence Labeling Task	74
4.3.4	Evaluation on Real Games	75
4.3.4.1	Model Fine-tuned with Sequence Classification Task	76
4.3.4.2	Model Fine-tuned with Sequence Labeling Task	78
4.3.4.3	Inference Time	80
4.3.5	Case Study	81
4.3.6	Summary	85
5	Conclusions and Future work	87
5.1	Conclusions	87
5.2	Limitations	88
5.3	Future work	89
5.4	Reflections	89
	References	91
A	Details of Frame-level Data Preparation	109
A.1	Data Cleaning for Pre-training Data	109
A.2	Classification Dataset for Stage 1	110
A.2.1	Manual Annotation	110

A.2.2	Cosine Similarity Matching	110
A.2.3	Auto-annotation with Binary Classification Models . .	111
A.2.4	Throw-in Direction Division	114
A.2.5	Data Cleaning Conditioned on Arm Angles and Joint Distances	114
A.2.6	Data Supplementation	115
A.3	Classification Dataset for Stage 2	116
A.3.1	Cosine similarity matching	116
A.3.2	Binary classification model	116
A.3.3	By-product of the goal kick binary model	117
A.3.4	Manual annotation from raising flags	117
B	Details of Sequence-level Data Preparation	119
B.1	Data Cleaning for Pre-training Data	119
B.2	Sequence Classification Dataset for Fine-tuning	120
B.2.1	Manual Annotation	120
B.2.2	Converting from Frame-level Data	120
B.2.3	Inlier Detection with Pre-training Loss	124
B.2.4	Binary Whistle Classification Model	124
B.2.5	Combination of Rules	126
B.2.6	Collecting OTHER_DECISION	127
B.2.7	Expansion for the Sequence-labeling Task	128

List of Figures

1.1	Problem definition	3
2.1	Partitioning of ST-GCN. The black cross indicates the center of the skeleton. Green nodes are root nodes; Blue nodes are centripetal nodes; Red nodes are centrifugal nodes.	12
2.2	Referee signals defined by the Laws of the Game 2023/24 [37]	22
2.3	Assistant Referee signals defined by the Laws of the Game 2023/24 [37]	23
3.1	Overview of the frame-level model	29
3.2	Two-stage recognition of assistant referee action frames	33
3.3	Output probabilities for a THROW_IN	43
3.4	Output probabilities for a GOAL_KICK	43
3.5	Output probabilities for a CORNER_KICK	44
3.6	Output probabilities for a SUBSTITUTION	44
3.7	Output probabilities for a RAISE_FLAG_VERTICALLY and the subsequence THROW_IN_LEFT	44
4.1	Overview of the multi-task pre-training pipeline of S-SPT _R . The two different losses are combined by a weighted sum for pre-training.	50
4.2	Overview of the fine-tuning process of S-SPT _R . The student encoder in the pre-training phase is used for fine-tuning. . . .	58
4.3	Fine-tuning the model with the sequence labeling task under supervision from a binary classification model	63
4.4	Output probabilities for a DIRECT_FREE_KICK	82
4.5	Output probabilities for an INDIRECT_FREE_KICK	83
4.6	Output probabilities for a GOAL in the beginning	84
4.7	Output probabilities for the entire process of the GOAL signal .	85

List of Tables

3.1	Statistics of frame-level pre-training data	35
3.2	Final frame-level classification dataset for assistant referee action recognition Stage 1	37
3.3	Final frame-level classification dataset for assistant referee action recognition Stage 2	38
3.4	Performance of fine-tuned model for Stage 1, 7-way classification task.	40
3.5	Performance of fine-tuned model for Stage 2, 9-way classification task.	40
3.6	Performance of the two-stage action recognition model on a real game	41
3.7	Inference time of F-SPT _{AR} on the real game. Evaluations are performed on an Nvidia RTX 3060 GPU.	42
4.1	Referee action types in our classification task	59
4.2	Dataset collected for training binary classification model to provide token-level supervision. Each data point is a one-second-long referee skeleton sequence.	63
4.3	Statistics of sequence-level pre-training data	66
4.4	Sequence classification dataset for linear evaluation. Each sequence is 3 seconds long.	66
4.5	Final sequence-level classification dataset collected for referee action recognition. Each sequence is 4 seconds long.	68
4.6	Performance of the pre-trained model with the linear evaluation protocol.	70
4.7	Ablation study on pre-processing method and pre-training tasks.	70
4.8	Ablation study on the coefficient of multi-task pre-training components. The multi-task loss is defined as $\mathcal{L} = \alpha\mathcal{L}_{\text{motion}} + \beta\mathcal{L}_{\text{data2vec}}$	71

4.9	Performance comparison on referee action recognition on the sequence classification task.	73
4.10	Ablation study on additional inputs and their integration methods for the sequence classification task.	74
4.11	Performance comparison on referee action recognition on the sequence labeling task. Pre-masking means pre-sequence random-length masking as introduced in Section 4.2.3.4. . . .	75
4.12	Performance of the model trained with sequence classification task on real games.	77
4.13	Performance of the model trained with sequence labeling task on real games. 'Hard w/o pre-masking' means the model is fine-tuned with hard labels without pre-sequence random-length masking; 'Hard w pre-masking' means the model is fine-tuned with hard labels and pre-sequence random-length masking; 'Soft w/o pre-masking' means the model is fine-tuned with soft labels without pre-sequence random-length masking; 'Soft w/ pre-masking' means the model is fine-tuned with soft labels and pre-sequence random-length masking; . . .	79
4.14	Inference time of S-SPT _R on real games. Evaluations are performed on an Nvidia RTX 3060 GPU. Context sequences are 6 seconds long.	81
A.1	Empirical optimal values for cosine similarity matching for different action types. The frame rate is 25Hz.	111
A.2	Frames annotated automatically by binary classification models (the last column). The number before the '+' sign represents the count of positive samples, while the number after the '+' sign indicates the count of negative samples. . . .	113
A.3	Frames annotated by the sequence-level binary ST-GCN model for NO_DECISION.	113
A.4	Statistics of final collected data for assistant referee action recognition Stage 1. *NO_DECISION frames includes 93 frames labeled noisy from other classes in data cleaning. . . .	116
A.5	Dataset collected for training binary classification model for NEAR_SIDE_OFFSIDE	117
A.6	Statistics of three extra classes for assistant referee action recognition Stage 2.	118
B.1	Labeled skeleton sequences collected by converting frame data	123

B.2	Dataset collected for training binary classification model for whistle events	125
B.3	Composition of <code>OTHER_DECISION</code> class	128

Chapter 1

Introduction

Human action recognition is one of the key tasks in video understanding. Recent years have witnessed remarkable developments in this area, in both tracks of RGB-based [1, 2] and skeleton-based methods [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. The rapid advancement led to transformative applications in various domains, notably in sports analytics [13, 14, 15, 16, 17]. Especially in the field of football, the adoption of action recognition technology has successfully promoted the recognition of player actions such as shooting, passing, and dribbling [18, 19, 20, 21, 22, 23, 24], thereby providing analysts and professional football clubs with richer and more detailed data to formulate strategies and improve sports performance.

However, the domain of referee action recognition has not kept pace with these technological advances [25]. Despite the critical role referees play on the field, their signals have not been as thoroughly explored or understood by automated systems. Given the significant influence of referees' decisions on the game, their actions constitute a crucial component for computers to fully understand football matches [26, 27]. This work aims to close this gap by extending current advances in human action recognition to the field of referee signals, thereby bridging an important link in the chain of analysis and understanding of football matches.

Skeleton-based action recognition has attracted considerable attention in the field due to its resistance to weather, lighting conditions, and other background disturbances, as well as its lightweight nature. For daily life actions, researchers have explored various methods, including approaches based on Graph Convolutional Networks (GCNs) [3, 4, 28, 29, 9] and transformers [30, 31, 32, 33], utilizing both supervised and unsupervised learning strategies. These techniques have proven effective in interpreting

actions from skeleton inputs, indicating promising prospects for developing systems for referee action recognition. On the other hand, while extensive 3D skeleton data for football matches is available from Tracab *, the analysis of referee data remains relatively unexplored. This study seeks to explore the application of advanced skeleton-based action recognition methods to referee action analysis, aiming to deepen the understanding of referee behaviors with state-of-the-art techniques.

1.1 Problem

The core problem in this work is to identify and classify the specific actions and signals of football referees based on their skeletal inputs. These inputs include the skeleton sequences of three referees: the main referee and two assistant referees. The skeleton of the fourth official is excluded as they typically do not perform critical signaling actions. Given the abundance of unlabeled referee skeleton data and the scarcity of labeled data, a key focus is on how to utilize self-supervised learning methods to learn general patterns of referee actions, thereby enhancing recognition capabilities. Additionally, considering the unique characteristics of referee actions, this work also aims to explore optimal model designs that effectively integrate the knowledge of referee positioning, orientation, and ball location to boost recognition performance.

1.1.1 Problem Definition

Referee skeleton sequences can be represented as $\mathbf{x} \in \mathbb{R}^{l \times V \times d}$, where l denotes the number of frames, V is the number of joints, and d represents the dimensions. Typically, l is calculated by multiplying the time span by the frame rate. For instance, a 10-second sequence with a frame rate of 25 would have $l = 250$ frames. In this study, each skeleton comprises 21 joints ($V = 21$), and the 3D positions of these joints are given ($d = 3$) in a coordinate system centered at the center mark of the field.

*<https://tracab.com/>

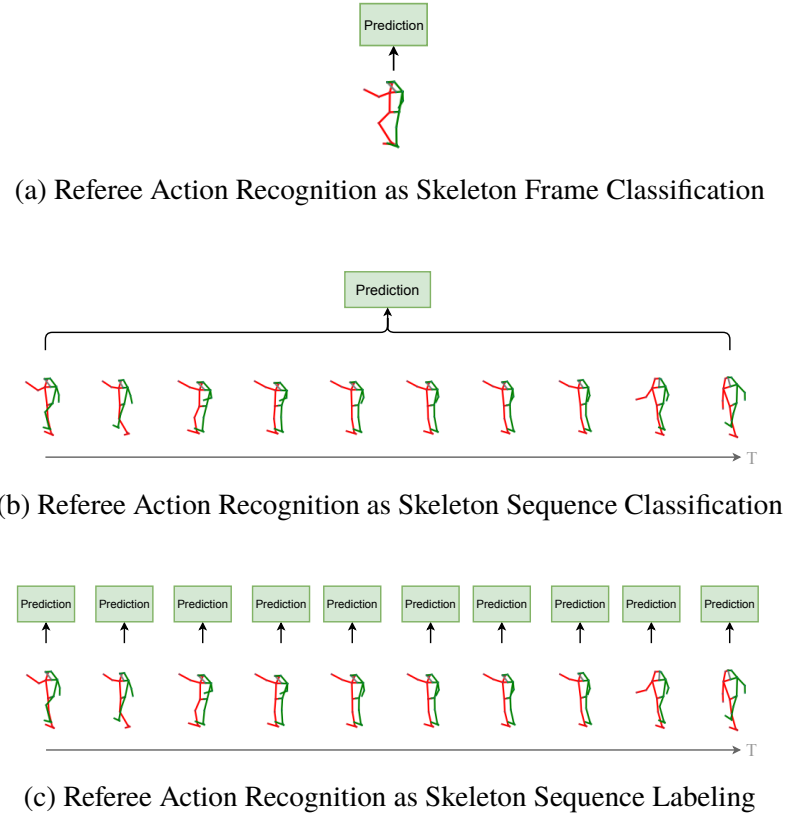


Figure 1.1: Problem definition

The problem of referee action recognition can be defined at either frame level or sequence level:

- **Referee Action Recognition as Skeleton Frame Classification:** The input for recognition is a single frame of skeleton $x_i \in \mathbb{R}^{V \times d}$, as shown in Figure 1.1a, and a label is predicted for each skeleton frame.

This approach is straightforward and lightweight but may lack sufficient contextual information. To address this, additional context can be incorporated at the sequence level. For the sequence level, the problem can be formulated in two ways: as a sequence classification or a sequence labeling task:

- **Referee Action Recognition as Skeleton Sequence Classification:** The input is a fixed-length skeleton sequence $x \in \mathbb{R}^{l \times V \times d}$, and a single label is predicted for the entire sequence to indicate the presence of an action, as shown in Figure 1.1b. Ideally, each skeleton sequence should contain only one action to minimize confusion; thus, the sequence length should not be too long.

- **Referee Action Recognition as Skeleton Sequence Labeling:** A longer sequence $\mathbf{x} \in \mathbb{R}^{l \times V \times d}$ serves as the input to provide more contextual information and a label is predicted for each frame within the sequence, as shown in Figure 1.1c, indicating whether an action is occurring at that particular frame.

1.1.2 Scientific and Engineering Challenges

Though previous studies have studied extensively on skeleton-based action recognition, their focus is mainly on everyday actions [34, 35, 11]. There remain both scientific and engineering challenges to extend existing skeleton-based action recognition methods to referee actions.

1.1.2.1 Scientific Challenges

Complex Motion Patterns for Football Referees Referee actions differ significantly from common daily activities in terms of dynamics and complexity. Compared to common daily actions, the skeleton data for referees often spans a larger area and involves more rapid movements, necessitating specialized pre-processing techniques and model designs that are not typically required for general action recognition.

Need for Granular Predictions Most existing action recognition systems are tailored for sequence classification, which may not adequately meet the needs of referee action recognition. For effective real-time applications, such systems must be capable of frame-by-frame prediction, necessitating the development of sequence labeling techniques. This requirement for detailed, continuous output is less explored in academic research and poses a unique challenge for this work.

Integration of Additional Information Incorporating context information, such as the position of the ball or the gesture of other referees, can provide richer information to enhance the accuracy of action recognition systems. However, this is not typically considered for recognizing everyday actions. Therefore, it poses challenges to develop specific models that can integrate and interpret multiple streams of contextual data effectively.

1.1.2.2 Engineering Challenges

Scarcity of Labeled Data The scarcity of labeled data for specific referee actions is a significant obstacle in this project. While self-supervised learning methods can facilitate the learning of general representations for referee

actions, labeled data remains crucial for training a robust recognition model. Therefore, the efficient collection, annotation, and validation of high-quality datasets becomes a significant challenge and is critical to support successful model training.

Gap between Model Prediction and Real-game Application For referee action recognition systems to be practical, they must operate in real games. However, models are trained with supervision on frames or short sequences, which does not guarantee consistent performance in real-game scenarios. Moreover, models trained in controlled environments may not account for the variability and unpredictability of diverse referee behaviors in real games. This necessitates the development of post-processing techniques to adapt model predictions to real-game scenarios, ensuring reliability and accuracy in real-game applications.

1.2 Purpose

This project aims to develop an effective recognition model for referee actions, holding significant potential benefits across various aspects of football. Referee actions influence the dynamics of the game, and accurately predicting these actions can enhance event detection. For referee committees, accurate recognition of referee signals can enhance the analysis of referees' performances. This could help make more informed decisions regarding referee arrangements and improve officiating. Additionally, a robust referee action recognition system can serve as a valuable tool for audiences and commentators, offering real-time interpretations of the referee's signals during games. Moreover, such a system could facilitate the realization of automatic officiating technology. For example, it could assist in labeling data from historical matches by pairing players' actions with the referees' decisions.

1.3 Research Content

In this work, we propose a pre-train and fine-tune approach for action recognition of referees and assistant referees, employing a transformer-based autoencoder. The model is initially pre-trained on large amounts of unlabeled data to capture general patterns of referee movements, and then fine-tuned on a dataset specifically collected for the recognition task. This pre-trained model also enhances the data collection process by providing a base model for automated annotations.

We utilize models of different granularities for action recognition of assistant referees and main referees:

For assistant referee action recognition, where actions tend to be more static, we employ a frame-level model. This model is pre-trained on individual frames through a joint position prediction task and is subsequently fine-tuned for frame classification specific to assistant referees. Various strategies are implemented for dataset collection, including rule-based methods and model-driven annotations. The frame-level data collected also facilitates the sequence-level data collection for the main referees.

For the main referee, whose actions are more dynamic and complex, we adopt a sequence-level approach to capture broader contextual information. The model is pre-trained on unlabeled referee skeleton sequences with a multi-task objective, including motion prediction objective [33] and data2vec objective [36], which predicts latent representations on a masked view of the input in a self-distillation setup. During fine-tuning, we explore both sequence classification and sequence labeling tasks for referee action recognition. Specific strategies are also deployed for collecting the sequence classification dataset. Additionally, inputs like the ball's position, the referee's absolute location, and other referees' actions are integrated during fine-tuning to boost the model's performance.

To validate the effectiveness of our models, we assess them not only on the validation and test sets derived from the collected datasets but also in real-game scenarios to evaluate their real-world performance, ensuring models perform reliably and accurately in practical applications.

1.4 Delimitations

The scope of this project does not include the development of novel action recognition algorithms that outperform existing methods. Instead, the focus is on applying and possibly adapting existing methods to the context of referee actions. Therefore, the evaluation of these methods will be specific to referee data and is not expected to be evaluated on public benchmarks commonly used in action recognition research. This limit is set to maintain the goals and feasible scope of the project.

1.5 Structure of the Thesis

In Chapter 2, we present the relevant background of skeleton-based action recognition methods and discuss the relevant referee signals as specified by the Laws of the Game [37]. Chapter 3 details the methodology and results for frame-level action recognition for assistant referees, outlining the specific techniques and strategies employed for frame-level pre-training, fine-tuning, and data collecting. Chapter 4 describes sequence-level action recognition for main referees, presenting detailed approaches for sequence-level pre-training and fine-tuning. Chapter 5 concludes the project and outlines potential directions for future improvements in the field of referee action recognition.

Chapter 2

Background

2.1 Skeleton-based Action Recognition

Action Recognition is one of the most important video understanding tasks and has significant applications in the real world. Skeleton-based action recognition has attracted extensive attention in the area of action recognition thanks to its advantage of being insensitive to weather, lighting conditions, and other background interference and being lighter.

Skeleton-based Action Recognition takes a sequence of skeleton data as the input, where the skeleton data consists of the 3D position of each joint of the human body. The task is a classification task, so the model is required to predict the action type based on the input skeleton sequence. For better development in this field, multiple benchmarks have been established and are widely used, such as Northwestern-UCLA [38], NTU-RGB+D [34], NTU-RGB+D 120 [11], Kinetics [35], and PKUMMD [39]. The action types in these benchmarks include both single-person actions and two-person interactive actions. Nevertheless, the action types are mostly actions in everyday life, which are different from the referee actions we want to classify in this work.

Studies for skeleton-based action recognition can be generally categorized into supervised learning methods and unsupervised learning methods. Graph Convolutional Networks (GCNs) [3, 4, 28, 29, 9] are currently the mainstream method for skeletal action recognition in supervised learning. GCNs are proposed to effectively model skeletal data using the natural connection graph of human joints. By defining partitions with an order for each joint's neighbors, GCNs successfully apply the convolution to a graph and improve the modeling efficiency. Other methods in a supervised learning manner

include RNNs [40, 41, 42, 43], CNNs [44, 45, 46], and Transformers [30, 31, 32, 33].

Unsupervised learning also plays a crucial role in skeleton-based action recognition because of its ability to learn complex patterns and features from unlabeled data, which are often abundant and easily available in real-world scenarios. This approach doesn't require extensive manual labeling, significantly reducing the time and resources required for training advanced deep models in real applications. Common unsupervised learning methods for skeleton-based action recognition include contrastive learning-based methods and autoencoder-based methods. Contrastive Learning is a powerful approach in the field of unsupervised learning [47, 48, 49]. The main idea is to pull the representations of positive pairs closer while pushing the representations of negative pairs farther apart. Autoencoder is a classical method that provides another way of utilizing unlabeled data. The core idea of an autoencoder is to first use an encoder to map the input to a latent space, and then ask the decoder to reconstruct the original input. A variant of autoencoder is Masked Autoencoder (MAE), which holds out a part of the input and requires the model to predict the missing part. MAE has been shown to work excellently on NLP tasks [50, 51] and computer vision tasks [52]. Following the remarkable success of Transformer-based MAE [52] in the field of computer vision, numerous studies have started to explore the application of MAEs for skeleton-based action recognition [53, 33].

In this study, we employ the Spatial-Temporal Graph Convolutional Network (ST-GCN) [3] from the GCN family as a baseline for our system. To achieve efficient pre-training, our approach leverages self-supervised learning methodologies, notably the masked autoencoder techniques Masked Motion Predictor (MAMP) [33] and data2vec [36, 54]. We will introduce them in the following sections.

2.1.1 ST-GCN

Spatial-Temporal Graph Convolutional Network (ST-GCN) [3] is the opening work of the GCN family and a widely used baseline. Convolutional neural networks are widely used in image tasks where the neighbors of each pixel and their order are well-defined. One challenge in applying the convolution operation to the human joint graph is that the number of neighboring joints of each joint is not fixed, and the neighboring joints have no clear order, which adds to the difficulty of multiplying weights in a convolutional kernel to a neighborhood of a joint. To address this challenge, ST-GCN proposed to define

an ordered partition for the neighbors of each joint so that a convolution can be applied. ST-GCN further proposed to apply a convolution along the time dimension, enabling the model to learn from spatial and temporal information at the same time.

Formally, We want to construct a representation for a spatial-temporal skeleton sequence graph $G = (V, E)$, where V is the node set and E is the edge set. Assume this sequence has N joints and T frames. We use $v_{ti}(t = 1, \dots, T; i = 1, \dots, N)$ to indicate the i -th joint at the t -th frame.

The convolution on the graph can be extended from a general convolution. For a spatial location \mathbf{x} , the output of a general convolution $f_{\text{out}}(\mathbf{x})$ in one channel can be formulated as

$$f_{\text{out}}(\mathbf{x}) = \sum_n f_{\text{in}}(\mathbf{p}(\mathbf{x}, n)) \cdot \mathbf{w}(n), \quad (2.1)$$

where $\mathbf{p}(\mathbf{x}, n)$ is a sampling function enumerating the neighbor position of the location \mathbf{x} ; $\mathbf{w}(n)$ is the weight function that provides weights to conduct the inner product with different neighbors of \mathbf{x} ; f_{in} is the input feature map of the channel.

For traditional convolution on images, $\mathbf{p}(\mathbf{x}, n)$ enumerates a $K \times K$ area around \mathbf{x} , where K is the kernel size; $\mathbf{w}(n)$ iterates weights stored in the $c \times K \times K$ weight matrix. To extend convolution on the graph, ST-GCN defines the sample function $\mathbf{p}(\mathbf{x}, n)$ on the graph as the neighbor set $B(v_{ti}) = \{v_{tj} \mid d(v_{tj}, v_{ti}) \leq D\}$ of a node v_{ti} , where $d(v_{tj}, v_{ti})$ indicates the minimum length of the path from v_{tj} to v_{ti} . In their work, $D = 1$. So the sampling function for a node v_{ti} is simply iterating all its neighbor v_{tj} with distance 1:

$$\mathbf{p}(v_{ti}, v_{tj}) = v_{tj}$$

They define the weight function by partitioning the neighbor set $B(v_{ti})$ into a fixed number of K subsets. So we have a mapping $l_{ti} : B(v_{ti}) \rightarrow \{0, \dots, K-1\}$, mapping each node in the subset to a subset label. In this way, the weight function can be expressed as:

$$\mathbf{w}(v_{ti}, v_{tj}) = \mathbf{w}'(l_{ti}(v_{tj}))$$

Combining the new definition of sample function and weight function on the graph, ST-GCN is formulated as

$$f_{\text{out}}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{\text{in}}(v_{tj}) \cdot \mathbf{w}'(l_{ti}(v_{tj})), \quad (2.2)$$

where $Z_{ti}(v_{tj}) = \|\{v_{tk} | l_{ti}(v_{tk}) = l_{ti}(v_{tj})\}\|$ the normalization term, equaling to the cardinality of the corresponding subset.

ST-GCN proposed several partitioning strategies, among which spatial configuration partitioning is the most efficient. Spatial configuration partitioning divides the neighboring nodes of each node into three subsets:

- root node itself
- centripetal nodes: the neighboring nodes that are closer to the skeleton center than the root node
- centrifugal nodes: the neighboring nodes that are farther away from the skeleton center than the root node

The skeleton center is defined as the gravity center in ST-GCN. As shown in Figure 2.1, the two green nodes are the selected nodes. For each selected node, the neighboring nodes and the node itself are divided into three groups based on their distance to the skeleton center (black cross), as shown in three different colors. Compared to the root node (green), Centripetal nodes (blue) have shorter distances to the center, while centrifugal nodes (red) have longer distances to the center. Formally, the mapping from neighboring nodes to the subset label is

$$l_{ti}(v_{tj}) = \begin{cases} 0 & \text{if root} \\ 1 & \text{if centripetal} \\ 2 & \text{if centrifugal} \end{cases} \quad (2.3)$$

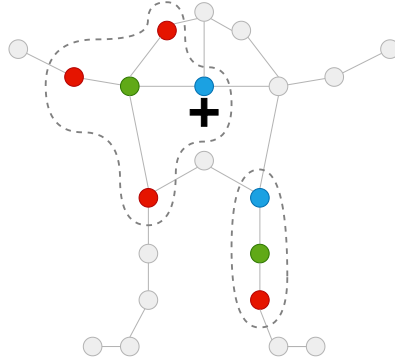


Figure 2.1: Partitioning of ST-GCN. The black cross indicates the center of the skeleton. Green nodes are root nodes; Blue nodes are centripetal nodes; Red nodes are centrifugal nodes.

ST-GCN also proposed applying convolution in the time dimension by connecting the same joint across time to expand the graph. The convolution in the time dimension is built similarly. With well-defined convolutions on human skeleton graphs, ST-GCN successfully leverages the graph structure to better model skeletal sequences, triggering a flourishing exploration of GCNs for skeleton-based action recognition.

2.1.2 MAMP

Masked Motion Predictor (MAMP) [33] is a foundational technique employed in this work to model referee skeletons. The MAMP framework is simple yet effective, outperforming all contrastive learning-based methods without multi-stream input. MAMP proposed a transformer-based MAE trained to predict skeleton motions instead of joint positions, which is suggested by Wu et al. in their SkeletonMAE [53], first introducing Transformer-based MAE into skeleton-based action recognition.

MAMP consists of four parts: A motion-aware masking module, a Joint Embedding module, a Transformer encoder, and a Transformer decoder. For an input skeleton sequence, it is first converted to tokens by the embedding module, then masked by the masking module. After that, the Transformer encoder processes the tokens to generate feature tokens in the latent space, and then the decoder is asked to reconstruct the motion of the masked joints in the original input sequence.

MAMP proposed one token to denote several consecutive frames for one joint since the position typically doesn't change dramatically between adjacent frames. By merging neighboring frames, MAMP reduces the number of total tokens for one skeleton sequence, leading to less memory cost and allowing longer sequences to be modeled. Formally, for the input skeleton sequence $S \in \mathbb{R}^{T_s \times V \times C}$ (where T_s is the sequence length; V is the number of joints; C is the number of channels, typically 3 for 3D position), it is divided temporally into non-overlapping segments $S' \in \mathbb{R}^{T_e \times V \times l \times C}$, where l is the length of each segment, also called time patch, and $T_e = T_s/l$ is the number of time patches. Then for each joint in each time patch, they embed it into one token:

$$E = \text{JointEmbed}(S') \in \mathbb{R}^{T_e \times V \times C_e} \quad (2.4)$$

where C_e is the dimension of embedding features. The `JointEmbed` module is composed of only one linear layer.

The idea of motion-aware masking is to mask the joint which has a larger motion. To compute the intensity of motion, they first extract motion with

stride m : $M_i = S_i - S_{i-m}$. To be consistent with joint embedding, they set $m = l$ and reshape the motion into the same shape as S' : $M' \in \mathbb{R}^{T_e \times V \times l \times C_i}$. Thus, the motion intensity can be expressed as

$$I = \sum_{i=0}^l \sum_{j=0}^{C_i} |M'_{:, :, i, j}| \in \mathbb{R}^{T_e \times V} \quad (2.5)$$

This motion intensity is further converted into probability distribution with a temperature γ :

$$\pi = \text{Softmax}\left(\frac{I}{\gamma}\right) \quad (2.6)$$

Then, positions to mask are sampled by index sampling with Gumbel max:

$$\begin{aligned} g &= -\log(-\log \epsilon), \epsilon \in U[0, 1]^{T_e \times V}, \\ idx^{\text{mask}} &= \text{Index-of-Top-K}(\log \pi + g), \end{aligned} \quad (2.7)$$

where $U[0, 1]$ is a uniform distribution between 0 and 1, and idx^{mask} indicated the indice of joints that are masked.

In the Transformer Encoder, the masked tokens are removed, and the rest of the sequence is served as input. So the input length is $N_u = T_e \times V \times (1 - \text{mask_ratio})$. Assume there are L_e layers in the encoder, then the latent representation is given by

$$\begin{aligned} H_0 &= \text{Flatten}(E), \\ H'_l &= \text{MultiHead-SelfAttention}(\text{LayerNorm}(H_{l-1})) + H_{l-1}, \quad l \in 1, \dots, L_e \\ H_l &= \text{MLP}(\text{LayerNorm}(H'_l)) + H'_l, \quad l \in 1, \dots, L_e \\ H_e^u &= \text{LayerNorm}(H_{L_e}) \end{aligned} \quad (2.8)$$

In the decoder, learnable mask tokens are inserted back into the positions where the joints are masked according to idx^{mask} . The recovered sequence H_e is hence back into shape $H_e \in \mathbb{R}^{T_e \times V \times C_e}$. Then H_e is sent to the decoder, which has the same architecture as the encoder. Therefore, the output feature from the decoder can be denoted as

$$Z_d = \text{Decoder}(H_e) \quad (2.9)$$

For motion prediction, a fully connected layer is added on top of the decoder. For the decoded feature $Z_d \in \mathbb{R}^{T_e \times V \times C_d}$, the predicted motion is

$$M_{pred} = \text{Linear}(Z_d) \quad (2.10)$$

Then an MSE loss is computed between M_{pred} and reconstruction target motion M_{target} to update the whole model:

$$\mathcal{L} = \frac{1}{|idx^{mask}|} \sum_{(i,j) \in idx^{mask}} \left\| \left(M_{i,j,:}^{pred} - M_{i,j,:}^{target} \right) \right\|_2^2 \quad (2.11)$$

2.1.3 data2vec

data2vec [36] is a general self-supervised learning framework proposed for multiple modalities such as language, speech, and vision. This idea is adopted by Skeleton2vec [54] later for skeleton-based action recognition.

The core idea of data2vec is to predict latent representations of the complete input data from a masked view of the input within a self-distillation framework using a standard Transformer architecture [55]. This approach allows the model to predict a contextualized training target computed from the entire sequence rather than from a local token, thereby encouraging the model to learn more meaningful and integrated representations.

Specifically, data2vec only predicts representations for masked tokens. The representations to predict are contextualized representations derived from the encoding of the particular token and other information from the sample using self-attention in the Transformer network. A teacher model is maintained to provide the target representations. The teacher model is parameterized by an exponential moving average (EMA) of the model parameters θ .

$$\theta_{teacher} = \tau \theta_{teacher} + (1 - \tau) \theta \quad (2.12)$$

where τ is the decay factor to control the sensitivity of the EMA to recent changes. Typical τ values are close to 0.999 or 0.9999, sometimes with linearly increasing.

Training targets are constructed based on the output of the top K Transformer blocks of the teacher network. Specifically, the output of the FFN layer before the residual connection is used. The output is normalized before averaging over K blocks to get the training target. Denote the output of the Transformer block l at time step t as a_t^l , the training target is given by

$$y_t = \sum_{l=L-K+1}^L \text{Normalize}(a_t^l) \quad (2.13)$$

where L is the number of total Transformer blocks. A common normalization method is instance normalization.

The training objective used in data2vec is a Smooth L1 loss:

$$\mathcal{L}(y_t, f_t(x)) = \begin{cases} \frac{1}{2}(y_t - f_t(x))^2/\beta & |y_t - f_t(x)| \leq \beta \\ |y_t - f_t(x)| - \frac{1}{2}\beta & \text{otherwise} \end{cases} \quad (2.14)$$

where β controls the transition from a squared loss to an L1 loss, depending on the size of the gap between the target y_t and the model prediction $f_t(x)$ at time-step t . This smoothed loss is less sensitive to outliers. In their work, the β value is set between 1 and 4, depending on the task type.

2.1.4 Other Methods

2.1.4.1 Supervised Learning Methods

GCN-based Methods Following ST-GCN [3], many works are proposed to improve the GCNs in different ways, such as graph connections, modeling efficiency, learning objectives, and long-range dependencies. 2s-AGCN [4] proposed to use two-stream input and allow the model to learn adaptive joint connections from data. DGNN [56] builds GCN on directed acyclic skeleton graphs. AS-GCN [57] extended the skeleton graph by adding extra links inference from body actions and body structures. DynamicGCN [58] introduced a lighter Context-encoding Network to learn the skeleton topology for GCNs, yielding a more lightweight model. MS-G3D [28] unified spatial modeling and temporal modeling, capturing long-range cross-spacetime correlations by introducing new edges across spacetime and disentangling them for smooth convolution. Shift-GCN [59] introduced shift convolution operation on the graph for GCNs and achieved 10x less computational complexity. DC-GCN [60] split channels in GCNs into several decoupling groups with a trainable adjacent matrix each, increasing the model's expressiveness. DDGCN [61] proposed to compute action-dependent dynamic neighboring sets for different nodes and utilized directional graphs in computing. SGN [5] incorporated semantics of joint types and frame indices for GCNs. MST-GCN [62] devised a multi-scale graph convolution module to enlarge the receptive field of each node for better modeling long-range and non-local relations. CTR-GCN [63] proposed to refine skeleton topology in a channel-wise way, leading to flexible and effective correlation modeling. EfficientGCN [64] constructed GCNs with more efficient Convolution blocks and a joint attention mechanism. InfoGCN [65] proposed to train GCNs based on information bottleneck objectives. STF [66] utilized gradient information to let GCNs focus on relevant spatial-temporal features. HD-GCN

[9] proposed a hierarchically decomposed graph combined with an attention-based aggregation module to learn meaningful edges for GCNs. STC-Net [8] introduced Spatial-Temporal Curves and Dilated Kernels for modeling spatiotemporal dependencies. GAP [67] introduced a text encoder powered by GPT-3 to guide the representation learning of GCNs with the encoded natural-language action description from the text encoder.

RNN-based Methods Recurrent Neural Networks (RNNs) are a common method for sequence modeling. Since the recognition is based on skeleton sequences, it is naturally suitable for RNN-based methods. RNN methods usually take the coordinates of a human joint at each frame as the input at each time step. In the pre-GCN era, the RNN-based method is one of the mainstream methods for skeleton-based action recognition. HBRNN [40] divided the human skeleton into five parts and utilized a hierarchical bidirectional RNN to model the skeleton sequence. PA-LSTM [34] was the baseline method provided for the NTU-RGB+D dataset. PA-LSTM incorporated a part-aware module into the LSTM, which enables the model to learn patterns of different human parts flexibly. ST-LSTM [41] proposed to traverse the skeleton data in a tree structure and used a trust gate to better control the information flow in LSTM. VA-LSTM [42] integrated a novel view-adaption scheme to LSTM to automatically determine the most suitable viewpoint for recognition. STA-LSTM [68] incorporated the attention mechanism into LSTM and trained the model in an end-to-end manner. Ind-RNN [69] proposed a new type of RNN where neurons are independent of each other in the same layer but connected across layers, which worked well for skeleton-based action recognition. AGC-LSTM [70] introduced graph convolution to replace the inner product in the LSTM cell and applied attention to enhance feature integration. ARRN-LSTM [43] proposed a Recurrent Relational Network followed by attention to extract features for LSTM. BGC-LSTM [71] leveraged Bayesian inference to a graph convolution and LSTM-based model to better capture the stochasticity and variation in the skeleton data. However, RNN-based methods are no longer the mainstream method since the existence of GCNs and some unsupervised methods later.

CNN-based Methods Convolutional neural networks (CNN) have proven to be highly effective in image processing. However, skeleton data consists of joint 3D coordinates, which are different from image pixels. Only a few earlier works tried to regard skeletons as pseudo-images and applied CNNs. Ke et al. [44] proposed to transform skeleton sequences into three clips with deep models, then used CNNs and Multi-Task Learning Networks to jointly process generated clips for action recognition. HCN [45] employed a CNN

model to hierarchically learn from point-level features to global co-occurrence skeleton features. PoseC3D [46] proposed to use 3D heatmap volume instead of graph sequence as the input and applied a 3D-CNN for action recognition. Though CNN-based methods are not preferred for skeleton-based action recognition, PoseC3D's experiment results outperform many advanced GCN-based methods, such as Shift-GCN and MS-G3D, indicating that 3D-CNN-based methods might have huge potential and deserve more investigation.

Transformer-based Methods Transformer [55] is a pure attention-based network proposed originally for Neural Machine Translation. It was then proven to be highly effective in language modeling [50, 51, 72] for various NLP tasks, and eventually leads to the extraordinary large language models [73] today. Nevertheless, Transformer is also in the process of replacing CNNs in image processing. Recent Transformer-based works [74, 75, 52] have shown its power in vision tasks. Since Transformer networks have generally good learning abilities for all modalities, many works have attempted to apply Transformer networks to skeleton-based action recognition. DSTA-Net [76] was the first attempt to model skeleton sequences solely based on attention modules for skeleton-based action recognition, where it decoupled the sequence into several streams for attention computation. ST-TR [30] devised a Spatial Self-Attention and a Temporal Self-Attention to learn the dynamics of skeleton sequences. STTFormer [31] proposed to capture relations between different joints over frames by using Transformer to explicitly model flattened frames within a tube, and then used a feature aggregation module to integrate tube features. SkeleTR [32] first modeled intra-person skeleton dynamics with GCNs, and then utilized Transformer to capture person interactions for in-the-wild action recognition. While the Transformer network has strong learning ability, it requires a huge amount of data to train in order to learn a capable model, which is not easy in the area of skeleton-based action recognition. However, in the setting of unsupervised learning, more data is available. Hence, more Transformer-based works focus on unsupervised learning for skeleton-based action recognition.

2.1.4.2 Unsupervised Learning Methods

Contrastive Learning-based Methods In the field of skeleton-based action recognition, skeletons can be collected through pose estimation tools from videos, while annotating action labels could be time and resource-consuming. Hence, many works explore the application of contrastive learning frameworks to learn better representations of skeleton sequences for action recognition.

CrosSCLR [77] proposed a cross-view contrastive-learning framework, which attempts to preserve cross-view consistency. MCC [78] built positive pairs by constructing speed-changed and motion-broken clips of skeleton sequences, aiming for maintaining motion consistency and continuity. ISC [79] proposed inter-skeleton contrastive learning, encouraging the representation of different frames in the same sequence to be close together while far away from other sequences. AimCLR [80] applied Abundant Information Mining on extremely augmented samples to improve the effect of contrastive learning. CPM [81] identified non-self-positive samples in a contextual queue to boost the effect of contrastive learning. CMD [82] introduced Cross-modal Mutual Distillation for positive mining. Moliner et al. [83] applied BYOL [49] on skeleton-based action recognition, aligning representation of aggressively augmented samples from a student model to that of conservatively augmented samples from a teacher model, where the parameter of the teacher model is an exponential moving average of the student model. RVTCLR+ [10] proposed to use skeleton sequences with different visual tempos as positive pairs in contrastive learning. ActCLR [12] extracted motion regions in the skeleton as actionlets to better guide contrastive learning. SkeAttnCLR [84] integrated local similarity and global features by attention for better contrastive pair building. PSTL [85] partially masked input streams in both spatial and temporal information to build positive pairs. HaLP [86] proposed to hallucinate new positive samples in the latent space to boost contrastive learning. HiCo [87] represented skeleton sequences into multiple-level features and performed contrastive learning on fusion features. Although contrastive learning methods have shown significant efficacy in skeleton-based action recognition, existing methods mainly rely on limited datasets curated for supervised learning tasks, which leads to questions about the potential for further enhancement by integrating more unlabeled data. Furthermore, the performance of contrastive learning methods strongly depends on the method adopted to construct positive or negative pairs, especially through data augmentation techniques, which is not a straightforward way to utilize unlabeled data.

Autoencoder-based Methods In addition to MAMP [33] and Skeleton2vec [54], many other works investigated autoencoder-based methods for skeleton-based action recognition. SeBiReNet [88] proposed to disentangle pose-dependent features and view-dependent features and train a denoising autoencoder to reconstruct the original input. Colorization [89] designed skeleton cloud colorization tech- unique, in which the skeleton sequences are first represented as 3D skeleton clouds, and each point in the cloud is colored

according to its spatial-temporal order. Then an autoencoder-based framework is applied to reconstruct the cloud. Hi-TRS [90] incorporated hierarchical Transformer-based multi-task feature learner, which includes frame-level, temporal-level and context-level prediction tasks. GL-Transformer [91] devised a global and local attention mechanism and proposed a new pertaining task on predicting multi-interval pose displacement to encourage the model to learn both global and local relations. SkeletonMAE (Yan et al.) [92] used GCNs as backbone models to build an encoder-decoder-based MAE.

2.1.4.3 Few-shot Learning Methods

Given the high cost of labeling data, annotating a small portion of the data in new classes is often more realistic, leading to the problem of obtaining good classification performance for unseen classes based on only a few samples. Existing general few-shot methods such as ProtoNet [93], NGM [94], PairNorm [95], and DropEdge [96] have been established as baselines of the few-shot learning for action recognition, yet their performances are not satisfactory, indicating that task-specific design is needed. DASTM [97] proposed a rank maximization constraint on skeleton representations to achieve spatial disentanglement and then used an attention-based spatial activation module to incorporate disentangled representations for matching. JEANIE [98] performed joint alignment of temporal blocks and selected viewpoint indexes between the support sample and query sample, choosing the smoothest path to avoid sudden jumps in matching temporal positions and view indexes. PAINet [99] introduced a spatial self-attention module and a spatial cross-attention module to mitigate the challenge of similar spatial appearances and inconsistent temporal dependencies during matching. Yang et al. [100] performed multi-scale spatial-temporal feature matching for both skeleton position sequence and motion sequence for one-shot action recognition. ALCA-GCN [101] proposed a new metric that models an action as a matrix of local comparable units on both spatial and temporal dimensions and devises a selective sum to determine the sample similarity.

2.2 Data Available

Tracab has extensive tracking data of referees and assistant referees from numerous games, which forms the foundation of this work. The skeleton data for referees utilized in this work consists of a time series of 3D coordinates of various body joints, with frame rates ranging from 25 Hz to 50 Hz. These

3D coordinates are based on a pitch coordinate system, where the origin is at the center mark, the x-axis runs along the line connecting the two goals, the y-axis runs along the halfway line, and the z-axis extends vertically. The skeleton sequences for referee activities cover the entire duration of each game, providing a substantial amount of available data, which forms a strong foundation for unsupervised representation learning.

However, the referee skeleton sequences lack labels, meaning they do not include information about the specific signals given by referees. Existing labels are annotated for game events, which do not correspond with the timing of the referee's signals. For instance, a label for a goal kick or a free kick is marked at the time the ball is kicked, which typically does not align with the time when the referee shows the signal. Some useful labels like 'whistle' and 'ball dead' could provide some coarse information, but they are insufficient for directly training models for referee action recognition.

Therefore, this work began with almost entirely unlabeled data. Not only is labeled data lacking, but the referee action classes are also undefined. The first challenge is to define the task by identifying the specific referee action classes of interest, ensuring they are both representative for application and practical for data collection. Subsequently, a significant challenge is to collect labeled data for the defined tasks, which is essential for the final model's performance.

2.3 Football Referee Actions

The objective of this project is to recognize referees' actions based on their skeletal data, including the referee and two assistant referees. According to the Laws of the Game [37], some of the referee signs are standardized, while some signs may vary according to different referees' habits.

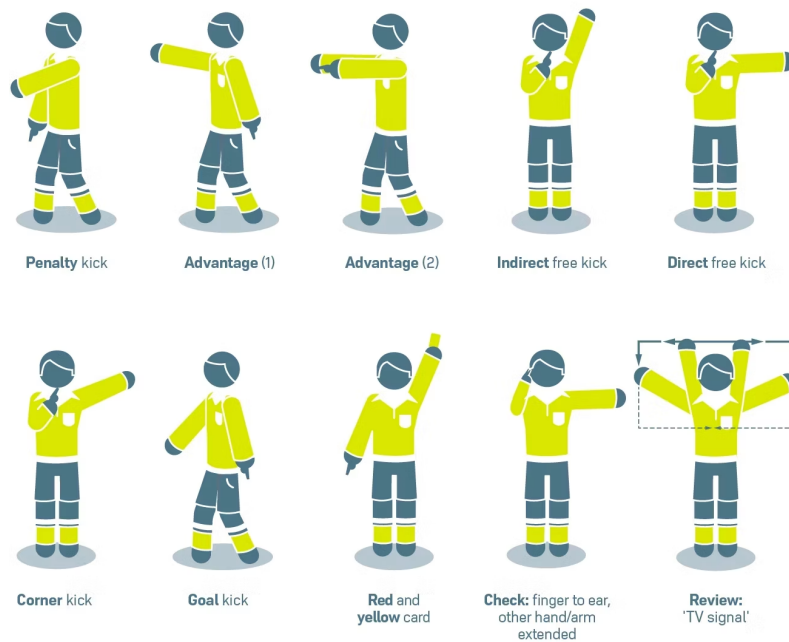


Figure 2.2: Referee signals defined by the Laws of the Game 2023/24 [37]

2.3.1 Standardized Referee Actions

For referees, the Laws of the Game provides standardized signals in Chapter 5 for direct free kicks, indirect free kicks, penalty kicks, advantages, goal kicks, corner kicks, disciplinary sanctions, VAR checks, and VAR reviews, as shown in Figure 2.2.

For assistant referees, Chapter 6 of the Laws of the Game provides their standardized signals, including throw-ins, corner kicks, goal kicks, substitutions, free kicks, and offsides on the near, middle, or far side of the field, as shown in Figure 2.3.

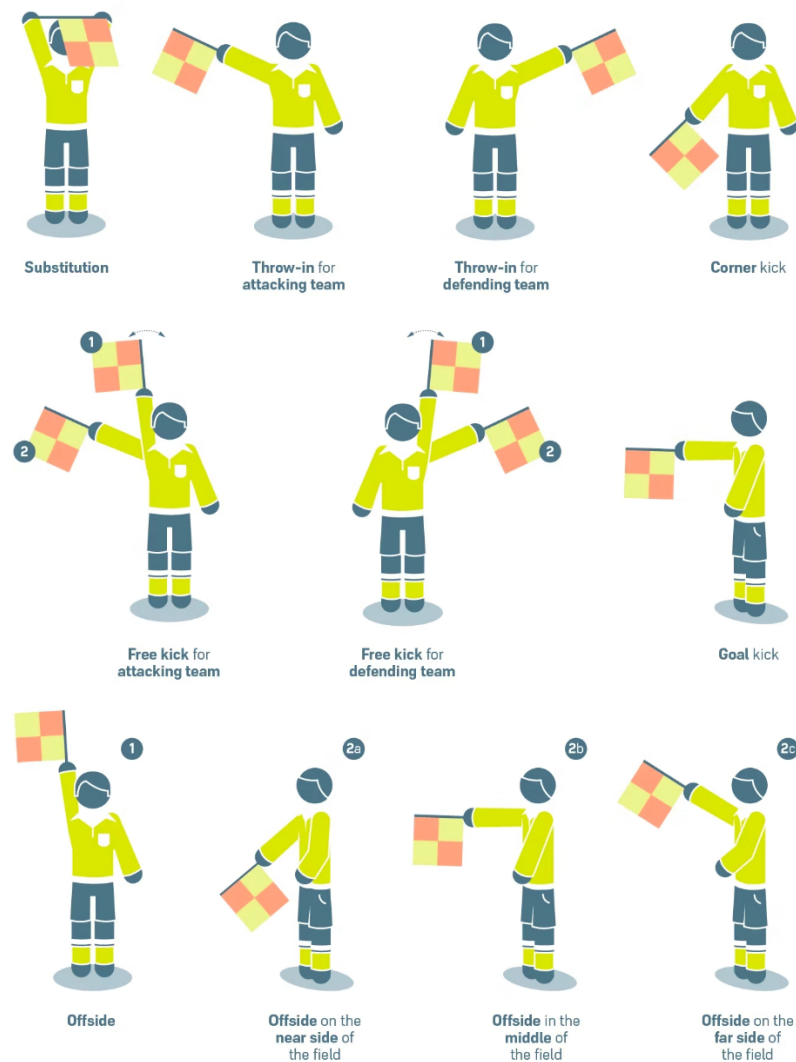


Figure 2.3: Assistant Referee signals defined by the Laws of the Game 2023/24 [37]

2.3.2 Non-standardized Referee Actions

Despite the aforementioned standardized referee actions, different referees typically have various habits of giving signals for some other decisions. For example, when it comes to indicating there is no foul, some referees might swiftly swing their hands back and forth in a 'no' gesture, whereas others may choose to point directly at the ball, indicating the touch is on the ball but not the player. Non-standardized actions are common for the referee, including decisions like no foul, substitution, interruption, calling a doctor, etc. Despite

the variation in gestures for the same decision among different referees, these actions remain clear to both players and audiences. Consequently, deep learning-based methods are still expected to understand these actions with sufficient training data.

2.4 Deep Learning for Action Understanding in Football

To the best of our knowledge, there have been no attempts to classify football referees' actions using either skeleton data or video data. However, attempts are made to understand football through video data [21, 102, 103]. To facilitate football video understanding, Giancola et al. proposed SoccerNet [26], a scalable dataset for football action spotting. Jiang et al. proposed SoccerDB [104], introducing more tasks including object detection, action recognition, temporal action localization and replay segmentation. SoccerNet-v2 [27] further expanded the task to also include camera shot segmentation and replay grounding. Following these prior works, numerous studies continued building on the advancements in football video understanding and analysis [105, 106, 16, 107, 108, 109], addressing various challenges and introducing innovative techniques. Nevertheless, existing works focus on RGB-based video understanding; recognizing football-related actions using skeleton data remains a challenging task.

2.5 Summary

Skeleton-based action recognition has been a popular research topic, including supervised and unsupervised methods. Graph Convolutional Networks (GCNs) are the predominant approach for supervised learning, with Spatial-Temporal Graph Convolutional Networks (ST-GCN) serving as an important baseline for our methods. Pre-training techniques based on masked autoencoders, such as Masked Motion Predictor (MAMP) and data2vec, are the basis of this work. These techniques allow the model to learn from large amounts of unlabeled data we have, enhancing its ability to understand complex patterns. However, since these methods primarily focus on everyday actions, adapting them to recognize the specific actions of football referees is still a challenging task.

To the best of our knowledge, although numerous studies have utilized deep learning methods to understand football, they primarily rely on video data. No

prior research has focused on classifying referee actions based on skeletal data. This study aims to bridge this gap by leveraging the advanced methodologies mentioned before. Therefore, this endeavor requires building all necessary components from scratch, including the collection of training datasets, the establishment of benchmarks, and the implementation and adaption of models specifically for recognizing football referee actions.

Chapter 3

Frame-level Assistant Referee Action Recognition

3.1 Overview

Assistant referees play a crucial role in supporting the main referee by monitoring the sidelines, primarily focusing on offsides and other boundary-related decisions like throw-ins, goal kicks, and corner kicks during a football match. They also assist in showing substitutions, helping to enforce throw-in positions, and signaling fouls and other infringements that occur out of the main referee's line of sight [37]. The typical actions of assistant referees are standardized, as shown in Figure 2.3. Except for the signal of free kicks, the other actions of assistant referees are static, often maintained for longer than one second. Consequently, there is minimal need to model these actions dynamically, as data from single frames is likely sufficient for interpreting their actions. Therefore, we propose a pipeline utilizing a frame-level model specifically designed for recognizing the actions of assistant referees. This pipeline is structured into two steps:

- **Pre-training:** Learn representations from unlabelled skeleton frames with transformer-based masked autoencoder in a self-supervised approach.
- **Task-specific Fine-tuning:** Fine-tune the pre-trained model on a labeled dataset specific to referee action recognition to adapt it for the task of classification.

However, a significant challenge in this pipeline is the scarcity of data. While there is plenty of unlabelled data available for pre-training, specifically

labeled data is limited, which could prevent the fine-tuning stage from being successful and result in an ineffective model. To address this issue, we have implemented several data collection strategies, notably utilizing the pre-trained model. These methods include:

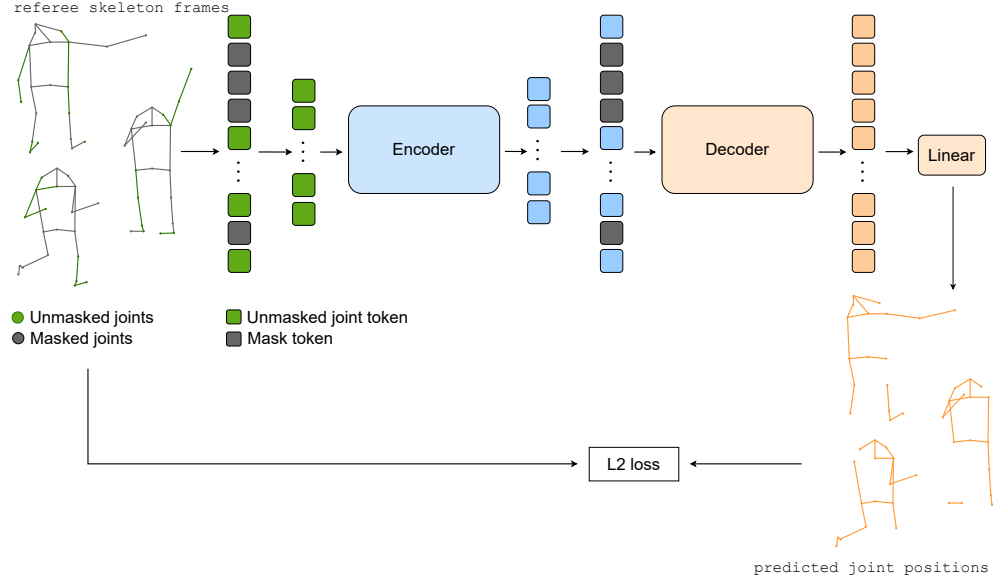
- Manually annotating data
- Finding data based on specific arm angles
- Using cosine similarity matching within the feature space of the pre-trained model
- Fine-tuning a binary classification model with the pre-trained model and using it to help with annotation

Another challenge in the fine-tuning stage is the presence of minor classes that are easily confused with more common ones, potentially leading to ineffective learning. For instance, specific actions like far side offside, middle side offside, and near side offside occur only when an offside is called, which is infrequent, and these actions are easily mistaken for goal kicks or raising flag actions. To mitigate such confusion, we propose a two-stage recognition scheme for assistant referees. In the first stage, we classify seven common actions, including goal kicks, corner kicks, throw-ins, and a general raising flag action. If a raising flag action is identified, we then proceed to a more detailed nine-way classification to accurately recognize the specific action following the raised flag.

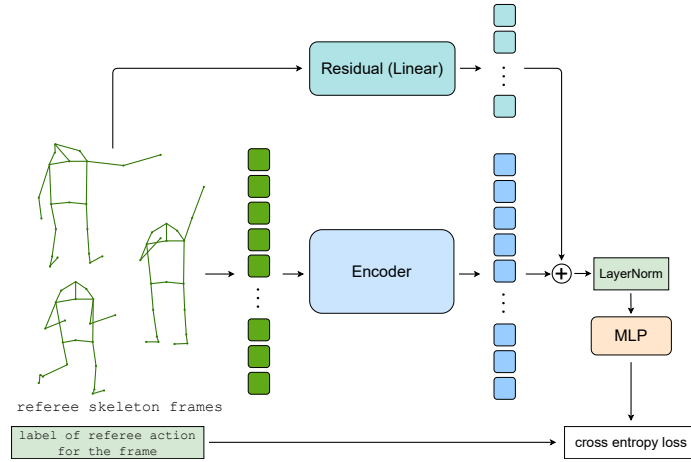
In the following sections of this chapter, we will provide detailed descriptions of the methodologies and strategies implemented to address the challenges outlined. These include thorough explanations of the pre-training techniques in Section 3.2.1, specifics of the two-stage classification system in Section 3.2.2, techniques for post-processing in Section 3.2.3, and additional details on the technical aspects of the proposed pipeline. We will discuss the data collection methods in Section 3.3.1 and the results of the proposed methods in the rest of Section 3.3.

3.2 Method

We denote our model as F-SPT_{AR} (**F**rame-level **S**keleton **P**re-trained **T**ransformer for **A**ssistant **R**eferees) in the following part in this chapter.



(a) Overview of the pre-training process of F-SPT_{AR}



(b) Overview of the fine-tuning process of F-SPT_{AR}

Figure 3.1: Overview of the frame-level model

3.2.1 Pre-training

3.2.1.1 Model Architecture

The transformer model has demonstrated its effectiveness in learning representations across various data types, including text [50, 51, 72], images [74, 75, 52], and skeleton data [31, 32, 33]. Therefore, we propose utilizing a transformer-based masked autoencoder to pre-train our model on the skeleton frame data. We follow the setting in MAMP [33], as described in Section 2.1.4.2; both the encoder and decoder incorporate a bidirectional attention mechanism, enabling a thorough integration of contextual information. As shown in Figure 3.1a, several joints are randomly chosen to be masked from the referee skeleton frames (grey ones). Then the encoder processes the masked frame with only unmasked tokens to produce a representation, which the decoder then uses to reconstruct the masked portions of the input frames. This methodology is designed to enable the model to acquire a general understanding of the relationships between joint positions and the overall skeleton structure. We anticipate that this model will not only enhance the effectiveness of subsequent fine-tuning stages but also facilitate data collection.

3.2.1.2 Joint Embedding

As each frame consists of 3D coordinates of 21 joints, we transform these into 21 dense tokens to serve as inputs for our model. This is achieved by applying a linear layer to map the 3D positions into a dense space, along with a joint-specific embedding for each joint. The final input representation for each joint is the sum of the linear layer output and its corresponding joint embedding. This combined representation is then normalized using a layer normalization layer to ensure training stability.

Formally, the input frame $\mathbf{x} \in \mathbb{R}^{m \times d}$, where $m = 21$ is the number of joints and $d = 3$ indicates the 3D coordinates. The input frame is first centralized before the conversion since the absolute position of the referee is not so important for learning general knowledge of referee gestures, and the normalization could help reduce potential numerical issues.

$$\begin{aligned}\mu^c &= \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i \\ \mathbf{x}^c &= \mathbf{x} - \mu^c\end{aligned}\tag{3.1}$$

To ensure that the tokens contain both spatial and joint-type information, we employ a space encoder to transform 3D coordinates into dense vectors, complemented by learnable joint-specific embeddings that represent different joint types. We anticipate that incorporating joint type awareness into the model could enhance the effectiveness of attention modules in modeling dependencies between joints. The space encoder is a linear layer projecting vectors from \mathbb{R}^3 to \mathbb{R}^h , and the joint embedding $J \in \mathbb{R}^{m \times h}$ is a mapping from joint types to learnable dense vectors, where h is the dimension of hidden space. In this work, h is set to 256. The eventual embedding for each joint can be represented by

$$E_i = \text{LayerNorm}(\text{Linear}(\mathbf{x}_i^c) + J_i). \quad (3.2)$$

Eventually, each frame is converted into a sequence of 21 tokens representing 21 different joints and their positions.

3.2.1.3 Mask Strategy

We randomly mask 50% of the joints from the input frames. In MAMP [33], the mask ratio is set to 90%. However, MAMP is designed for frame sequences with significant temporal redundancy. In the case of a single frame, masking a joint results in insufficient information to infer its position. Therefore, we reduce the mask ratio to 50% to ensure the pre-training task remains manageable. The tokens corresponding to the selected masked joints are removed from the input frames. To reconstruct from modeled representations, learnable mask tokens are inserted back into the output of the encoder, as shown in Figure 3.1. The decoder is then asked to reconstruct the masked joints. It is crucial to note that since only the encoder is utilized during the fine-tuning stage, this approach ensures that the encoder does not encounter any mask tokens. Consequently, this eliminates the discrepancies between the training and testing data distributions.

Formally, for the embeddings of an input frame $E \in \mathbb{R}^{m \times h}$, the indices of the joints to be masked idx^{mask} are given by

$$\begin{aligned} p &= \eta \sim U[0, 1]^m \\ idx^{\text{mask}} &= \text{Index-of-Top-K}(p) \end{aligned} \quad (3.3)$$

where η is random noise drawn from a uniform distribution from 0 to 1; K is the rounded number corresponding to 50% of all joints, which is 10. The input to the encoder is then the extracted unmasked part $E_u \in \mathbb{R}^{K \times h}$ from the

embeddings E according to idx^{mask} .

3.2.1.4 Pre-training Task

We adopt joint position prediction as the pre-training task. Specifically, a linear layer is added on top of the decoder to predict the position of the masked joints. The model is then trained using an L2 loss, which compares the predicted positions to the actual positions of the masked joints.

Formally, the unmasked embeddings E_u are fed into the encoder to get the representation in the hidden space.

$$H_u = \text{Encoder}(E_u) \quad (3.4)$$

The learnable mask tokens are inserted back into the positions according to idx^{mask} . The representation with mask tokens $H_e \in \mathbb{R}^{m \times h}$ is then sent to the decoder.

$$Z_d = \text{Decoder}(H_e) \quad (3.5)$$

A linear layer is then applied to predict the positions of masked joints.

$$P^{\text{pred}} = \text{Linear}(Z_d) \quad (3.6)$$

An L2 loss is computed between predicted positions P^{pred} and the ground truth positions of masked joints P^{target} :

$$\mathcal{L} = \frac{1}{|idx^{\text{mask}}|} \sum_{i \in idx^{\text{mask}}} \left\| \left(P_i^{\text{pred}} - P_i^{\text{target}} \right) \right\|_2^2 \quad (3.7)$$

3.2.2 Fine-tuning for Two-stage Recognition

3.2.2.1 Two-stage Recognition

We propose a two-stage recognition process for assistant referee actions. Stage 1 is a 7-way classification for common actions, as listed in Figure 3.2. Once a `RAISE_FLAG_VERTICALLY` action is identified in Stage 1, Stage 2 is activated to determine the specific action that follows the flag raise, with nine potential actions also listed in Figure 3.2. Different models are trained for different stages. We refer to the model for Stage 1 as F-SPT_{AR}-S1, and the model for Stage 2 as F-SPT_{AR}-S2. Note that we do not include the action of showing a goal scored in our recognition model, as the signal of an assistant referee to show a goal scored is simply running

towards the halfway line, which is difficult to distinguish from normal running actions. This two-stage approach helps to minimize confusion between some rare and common classes. For example, a `GOAL_KICK` might be mistaken for a `MIDDLE_SIDE_OFFSIDE`, and a `RAISE_FLAG_VERTICALLY` could be confused with a `FAR_SIDE_OFFSIDE`. With the two-stage design, these confusions are largely eliminated. In addition, labeled data for rare classes like offside is usually limited, as offsides are infrequent in football matches. The two-stage design restricts the label imbalance problem to Stage 2, lessening its impact on the classification of common actions.

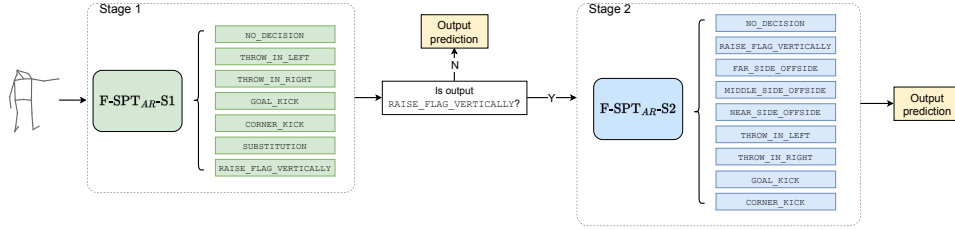


Figure 3.2: Two-stage recognition of assistant referee action frames

3.2.2.2 Fine-tuning Method

In the fine-tuning phase, we adapt our pre-trained model for the classification task by adding a linear layer on top of the pre-trained encoder while the decoder is not used, as shown in Figure 3.1b. We train two different models for the two stages using the same approach with collected data. All parameters of the encoder and the newly added linear layer are learnable, and we use cross-entropy loss to update these parameters. To address the issue of label imbalance in our dataset, we utilize weighted resampling to ensure that all classes have an equal probability of occurrence during the fine-tuning process. Furthermore, to make the model aware of the absolute position of the assistant referee, which is critical for distinguishing between similar actions, we incorporate a residual module. For instance, a `GOAL_KICK` and a `MIDDLE_SIDE_OFFSIDE` may appear similar in gestures, but `GOAL_KICK` occurs only near the goal area, whereas `MIDDLE_SIDE_OFFSIDE` can occur along any part of the touchline.

Residual Module The residual module consists of a straightforward linear layer that processes the original input frames before they are converted into embeddings. The residual layer maps the input frame $x \in \mathbb{R}^{m \times d}$ into a high-dimensional space, matching the dimension of the encoder output. The encoder’s output is then updated by adding the residual output to it, and this

combined output is normalized using a layer normalization step to ensure training stability. Formally, the representation H before the classification layer with a residual module is given by

$$\begin{aligned} \mathbf{o}_{\text{residual}} &= \text{Linear}(\mathbf{x}) \\ H &= \text{LayerNorm}(\text{Encoder}(E_u) + \mathbf{o}_{\text{residual}}) \end{aligned} \quad (3.8)$$

3.2.3 Post-processing

With two fine-tuned models for Stage 1 and Stage 2, we could obtain frame-level predictions for assistant referee actions in real-game scenarios. However, for actual games, our focus shifts from frame-level to action-level predictions. Therefore, we need to implement post-processing techniques to identify actions based on frame predictions. Since referee actions are continuous, an action typically involves several consecutive frames indicating the same action, whereas isolated frame predictions are more likely to represent noise than true actions. Leveraging this, we propose implementing three different thresholding techniques to accurately identify assistant referee actions, effectively reducing false positives.

Duration Threshold We confirm an action prediction only if all frames within the duration consistently indicate the same action. Otherwise, `NO_DECISION` is the prediction by default. We set the duration threshold as 0.5s in all our experiments.

Confidence Threshold We established a threshold for the softmax probability. Normally, using an `argmax` function is equivalent to selecting the class with the highest predicted probability, regardless of the actual probability values. However, when the classification probability is not dominant, it indicates a lack of confidence in the model's prediction. Our observations indicate that predictions made with low confidence are more likely to be incorrect. Leveraging this, we set a confidence threshold of 0.9 for all our experiments. Any predictions with a probability below this threshold are categorized as `NO_DECISION`.

Model-switching Threshold Once a `RAISE_FLAG_VERTICALLY` action is confirmed in Stage 1, we activate the Stage 2 model to identify the subsequent action. However, there are scenarios where the Stage 2 model might only detect `NO_DECISION` or the initial `RAISE_FLAG_VERTICALLY` action may have been incorrectly identified, resulting in no subsequent action. In such cases, returning to the Stage 1 model is necessary. To manage this transition, we introduce a model-switching threshold of 10 seconds.

Specifically, if an action is recognized in Stage 2, we switch back to Stage 1 immediately after the action ends. If no action is detected within 10 seconds, we automatically go back to Stage 1.

3.3 Experiments and Results

3.3.1 Data Preparation

3.3.1.1 Pre-training Data

Substantial data are available at Tracab for self-supervised representation learning of skeleton frames. As we focus specifically on learning patterns of referee gestures, we only use skeleton frames from referees for our pre-training data, including the main referee and two assistant referees. We load skeleton frames from 223 games. To avoid pre-training on data used for future testing, we designated 28 games specifically for validation and testing. Consequently, all validation and test datasets are derived from these 28 games, and we ensure that the pre-training data exclude any frames from them. This separation guarantees that the pre-training phase remains uninformed about the evaluation sets so that the evaluation sets are fair measures of the model’s generalization capabilities. To enhance the quality of the pre-training data, we propose several data cleaning strategies, including removing corrupted frames, as well as noisy frames in both spread and motion. Refer to Appendix A.1 for details of the pre-training data cleaning.

Eventually, after filtering out corrupted and noisy frames, we collected a large-scale dataset for frame-level pre-training, as shown in Table 3.1. From the 28 games designated for evaluation, we selected three games for the validation set.

	# Frames
Training set	78,559,500
Validation set	1,606,500

Table 3.1: Statistics of frame-level pre-training data

3.3.1.2 Classification Dataset for Stage 1

We propose several strategies for collecting datasets for Stage 1. These strategies can be applied in any sequence and may be repeated as necessary

since the primary objective is to create a high-quality frame-level assistant referee classification dataset for Stage 1. The strategies are as follows:

Manual Annotation Manual annotation is straightforward and reliable but inefficient for data collecting. Nevertheless, it can serve as a valuable starting point for implementing the automatic and semi-automatic strategies outlined below.

Cosine Similarity Matching We use cosine similarity within the latent space of the pre-trained model to automatically annotate new data. By applying two specific thresholds, frames with a low cosine distance to known class data are annotated as belonging to the same class.

Auto-annotation with Binary Classification Models For each class of interest, we fine-tune the pre-trained model on a specifically designed binary classification task to determine whether a frame belongs to the action class or not. The fine-tuned model is then used to automatically annotate unlabeled data. This strategy enables more complex decision boundaries, which are more expressive than the simple spherical boundaries used in cosine similarity matching.

Throw-in Direction Division To achieve a more fine-grained classification, we further split the class `THROW_IN` into `THROW_IN_LEFT` and `THROW_IN_RIGHT`, as this provides richer information for future applications, such as event detection.

Data Cleaning Conditioned on Arm Angles and Joint Distances Some errors are observed in the auto-annotated data using the aforementioned strategies. To enhance the dataset quality, action-specific denoising strategies based on arm angle and joint distance conditions are implemented.

Data Supplementation We enhance model robustness by supplementing the `NO_DECISION` data with frames of the assistant referee scratching or wiping sweat. Additionally, we enrich the `RAISE_FLAG_VERTICALLY` dataset according to specific rules to make it more expressive.

Please refer to Appendix [A.2](#) for details of these data-collecting strategies. With all the above strategies, we collected a frame-level classification dataset

for Stage 1, ensuring that both the validation set and the test set are label-balanced to provide a fair assessment of model performance, as shown in Table 3.2.

	Train	Valid	Test
NO_DECISION	5897	80	80
THROW_IN_LEFT	2690	80	80
THROW_IN_RIGHT	1707	80	80
GOAL_KICK	2745	80	80
CORNER_KICK	1326	80	80
SUBSTITUTION	838	80	80
RAISE_FLAG_VERTICALLY	937	80	80

Table 3.2: Final frame-level classification dataset for assistant referee action recognition Stage 1

3.3.1.3 Classification Dataset for Stage 2

A large portion of data collected for Stage 1 can be reused in Stage 2. However, Stage 2 introduces several new classes regarding offsides, requiring tailored data collection strategies. The new offsides classes are `FAR_SIDE_OFFSIDE`, `MIDDLE_SIDE_OFFSIDE`, and `NEAR_SIDE_OFFSIDE`, according to the Laws of the Game 2023/24 [37]. We employ the following strategies to collect data for offside frames:

Cosine similarity matching We apply the same cosine similarity matching strategy for the offside actions.

Binary classification model Similar to data collection in Stage 1, binary classification models for each offside class are trained for data annotation. However, the quality of auto-annotated data is not satisfactory. The strategy of training a binary classification model is less effective with insufficient data.

By-product of the goal kick binary model `MIDDLE_SIDE_OFFSIDE` can be annotated as a by-product of the binary classification model for `GOAL_KICK` in Stage 1 when a `GOAL_KICK` is predicted far away from the goal area.

Manual annotation from raising flags Offside frames often follow `RAISE_FLAG_VERTICALLY` actions. We manually annotate these subsequent frames to obtain high-quality labeled data.

Please refer to Appendix A.3 for more details. The final classification dataset collected for Stage 2 is shown in Table 3.3, ensuring label balance in the validation and test sets.

	Train	Valid	Test
NO_DECISION	5897	10	10
THROW_IN_LEFT	2690	10	10
THROW_IN_RIGHT	1707	10	10
GOAL_KICK	2745	10	10
CORNER_KICK	1326	10	10
RAISE_FLAG_VERTICALLY	937	10	10
FAR_SIDE_OFFSIDE	292	10	10
MIDDLE_SIDE_OFFSIDE	340	10	10
NEAR_SIDE_OFFSIDE	144	10	10

Table 3.3: Final frame-level classification dataset for assistant referee action recognition Stage 2

3.3.2 Pre-training

3.3.2.1 Network Architecture

We employ a Transformer-based masked autoencoder for pre-training, as outlined in Section 3.2.1. We set the number of encoder layers to 6 and the number of the decoder layers to 2 since only the encoder will be used in the fine-tuning stage. We set the hidden size to 256 and the dimension of the feed-forward network to 1024. The head number of the multi-head self-attention module is 8. The mask ratio of joints is set to 0.5.

3.3.2.2 Pre-training Details

During the pre-training phase, we utilize the AdamW Optimizer with a weight decay of $5e-4$ and betas set at (0.9, 0.999). The model is trained for 700k steps. The learning rate starts at 0 and increases linearly to $3e-4$ over the first 10,000 steps, followed by an inverse square root decay. We train the model on a single Nvidia RTX 3060, with the batch size set to 1024. The dropout rates applied to the attention matrix and hidden layers are both set at 0.1.

3.3.3 Fine-tuning

3.3.3.1 Fine-tuning Details

During the fine-tuning phase, we attach an MLP head to the encoder of the pre-trained model and discard the decoder. This MLP head is composed of

two layers, with a hidden layer dimension of 256 and a ReLU function for non-linear activation. Both Stage 1 and Stage 2 utilize the same fine-tuning configuration. The entire model is fine-tuned for 5 epochs. We use an AdamW optimizer, with betas set at (0.9, 0.999) and a weight decay of $5e-4$. The learning rate schedule follows that of the pre-training phase, starting with a linear warmup for the first 100 steps to $5e-5$ and then transitioning to inverse square root decay. The batch size is set to 32. A dropout rate of 0.5 is applied to the encoder output to prevent overfitting. The entire fine-tuning process is performed on a single Nvidia RTX 3060.

3.3.3.2 Experiment Results

Baseline We adopt a simple Multi-Layer Perceptron (MLP) as the baseline for both 7-way and 9-way tasks. The MLP consists of three layers with a hidden size of 256. We use the same optimization hyperparameters as in the fine-tuning, except that we train the MLP for 10 epochs.

The Stage 1 model, devised for a 7-way general action classification, is fine-tuned on the dataset shown in Table 3.2. The performance of the fine-tuned model is listed in Table 3.4. As shown in the table, the simple MLP achieves high accuracy on this task, suggesting that the 7-way classification is relatively easy. In addition, we compare the performance of our fine-tuned model to a vanilla transformer encoder without pre-training, as shown in the second line of the table. When trained from scratch, the vanilla transformer also achieves satisfactory performance comparable to that of the MLP, again suggesting that the classification task itself is not difficult. When fine-tuned from the pre-trained model, as shown in the last line of the table, the model demonstrates further improvement to an accuracy of 99.46% on both validation and test sets. It is important to note that during the data collection phase, we utilized the same pre-trained model for data annotation. As a result, the collected data are naturally distinguishable within the pre-trained model’s latent space. This eases the difficulty of fine-tuning the model to develop a classification model. To conclude, although the task itself may not be challenging, the data collection process contributes a lot to the excellent classification performance.

	#Params	Stage 1, 7-way Classification	
		Valid. Acc.	Test Acc.
MLP	0.08M	97.32	99.11
F-SPT _{AR} w/o pre-training	6.1M	98.75	98.93
F-SPT _{AR} -S1	6.1M	99.46	99.46

Table 3.4: Performance of fine-tuned model for Stage 1, 7-way classification task.

For the 9-way classification of raising flag events in Stage 2, we fine-tuned the pre-trained model with the same configuration on the dataset shown in Table 3.3. The performance of the fine-tuned model for Stage 2 is listed in Table 3.5. As shown in the first line, the MLP model behaves less competitively and only achieves an accuracy of 67.78% on the validation set and 66.67% on the test set. Similar to the model for Stage 1, we also compare the fine-tuned model to the model trained from scratch without the pre-training, as listed in the second line in the Table. For the Stage 2 task, a vanilla Transformer doesn’t perform satisfactorily as well, though it is slightly better than the MLP, the fine-tuned model demonstrates significantly better performance (line 3), achieving an accuracy of 96.67% on both the validation and test sets, thanks to the general knowledge learned in the pre-trained phase. The classification task in Stage 2 is notably more challenging due to sparse data availability, particularly for offside actions, highlighting the critical role of the pre-trained phase in achieving satisfactory performances.

	#Params	Stage 2, 9-way Classification	
		Valid. Acc.	Test Acc.
MLP	0.08M	67.78	66.67
F-SPT _{AR} w/o pre-training	6.1M	75.56	75.56
F-SPT _{AR} -S2	6.1M	96.67	96.67

Table 3.5: Performance of fine-tuned model for Stage 2, 9-way classification task.

3.3.4 Evaluation on a Real Game

Although the fine-tuned models achieve excellent accuracy on both the validation and test sets for both stages, the data collection process heavily relies on model auto-annotation, which does not ensure that the dataset’s distribution accurately reflects real-game scenarios. The most robust evaluation criterion is to assess the models’ predictions directly in a real-game environment. To this end, we have developed the post-processing technique outlined in Section 3.2.3, which allows for integrated predictions from the two-stage models. The evaluation was conducted on an MLS (Major League Soccer, men’s professional soccer league sanctioned by the United States Soccer Federation) game between Inter Miami CF and CF Montreal, played on February 26, 2023. We assessed the integrated prediction of the two models throughout the entire game. The ground truth for action types in the game was annotated entirely by hand. As shown in Table 3.6, the integrated prediction of the two-stage models achieves 0.9242 in both precision and recall on overall evaluation, resulting in an F1 score of 0.9242. Considering that the proportion of NO_DECISION frames in actual games is significantly higher than in the fine-tuning phase, we set tight thresholds as described in Section 3.2.3, effectively reducing false positives. False positives tend to occur on single frames but are generally unstable; therefore, the duration and confidence thresholds are effective in eliminating them. For specific action types such as CORNER_KICK and SUBSTITUTION, our model achieves 100% precision and recall, indicating flawless performance.

	# In Game	Precision	Recall	F1-Score
THROW_IN_LEFT	15	0.9333	0.9333	0.9333
THROW_IN_RIGHT	20	1.0000	0.9000	0.9474
GOAL_KICK	9	0.7500	0.8889	0.8136
CORNER_KICK	10	1.0000	1.0000	1.0000
SUBSTITUTION	5	1.0000	1.0000	1.0000
RAISE_FLAG_VERTICALLY	4	1.0000	1.0000	1.0000
FAR_SIDE_OFFSIDE	2	1.0000	1.0000	1.0000
MIDDLE_SIDE_OFFSIDE	1	-	0.0000	-
NEAR_SIDE_OFFSIDE	0	-	-	-
Overall	66	0.9242	0.9242	0.9242

Table 3.6: Performance of the two-stage action recognition model on a real game

The model missed five actions: two `THROW_IN_RIGHT`, one `THROW_IN_LEFT`, one `GOAL_KICK`, and one `MIDDLE_SIDE_OFFSIDE`. These misses were generally due to either insufficiently high confidence or the actions not maintaining a long enough duration for confirmation, illustrating the tradeoff in setting tight thresholds. Loosening these thresholds could potentially enhance the recognition of these actions but would likely increase the number of false positives. Specifically, the miss of `MIDDLE_SIDE_OFFSIDE` was influenced by the visual similarity between the `MIDDLE_SIDE_OFFSIDE` and `FAR_SIDE_OFFSIDE`, leading to insufficient confidence for a definitive classification in either category.

Additionally, the model generated 5 false positives, with four misclassified as `GOAL_KICK` and one as `THROW_IN_LEFT`. These false positives typically occur when the assistant referee waves their arm without a flag, looking like an authentic action since we cannot see the flag through skeletons. Filtering these out based solely on skeletal data can be challenging. These false positives should be, to a greater extent, considered as a limitation of relying only on skeleton input, and should be regarded as an acceptable shortcoming within the context of the current technological constraints.

Inference Time We evaluated the inference time of our model, as shown in Table 3.7. The inference time is measured for the entire prediction process per frame, including two models for the two stages, sequentially processing the skeletons of two assistants and outputting the prediction. We run the inference for 10,000 frames and report the average inference time per frame. As shown in the table, with a batch size of 1, the inference time per frame is approximately 9 ms, allowing the system to operate smoothly at up to 100 Hz in real-time applications. Increasing the batch size further reduces the inference time per frame. In conclusion, our recognition model for assistant referee frames is efficient enough to support real-time recognition tasks.

	Batch Size	GPU Memory Usage	Inference Time (ms/frame)
F-SPT _{AR}	1	322M	8.56
	4	322M	2.52
	16	322M	0.54
	64	322M	0.21
	256	408M	0.07
	1024	728M	0.04

Table 3.7: Inference time of F-SPT_{AR} on the real game. Evaluations are performed on an Nvidia RTX 3060 GPU.

3.3.5 Case Study

We conduct case studies to observe the variation in probabilities for predicted actions, in order to assess the stability of the model's predictions. We observe five cases for `THROW_IN`, `GOAL_KICK`, `CORNER_KICK`, `SUBSTITUTION`, and `RAISE_FLAG_VERTICALLY`, respectively. Note that we omit classes with probabilities consistently below 0.05 from the plot during our observations. In the skeletons depicted in the following figures, the left side of the body is shown in red, while the right side is shown in green.

THROW_IN As shown in Figure 3.3, the orange line represents the probabilities over frames during a throw-in signal. The probability is almost one when the assistant referee gives the signal and remains very stable. There are predictions of `CORNER_KICK` during the transition from `NO_DECISION` to `THROW_IN`; however, these predictions are short in duration and are easily filtered out by our duration threshold.

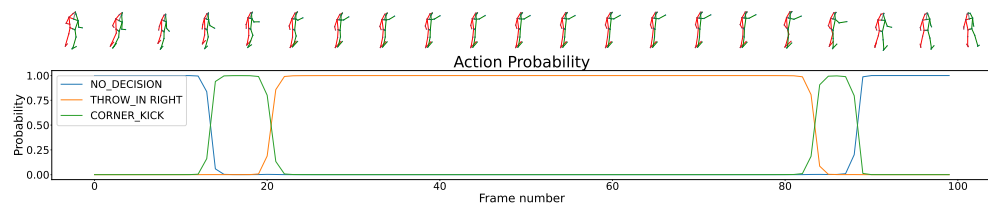


Figure 3.3: Output probabilities for a `THROW_IN`

GOAL_KICK As shown in Figure 3.4, the probability of the action being a `GOAL_KICK` rises to one as soon as the assistant referee raises the flag and remain extremely stable until the signal ends.

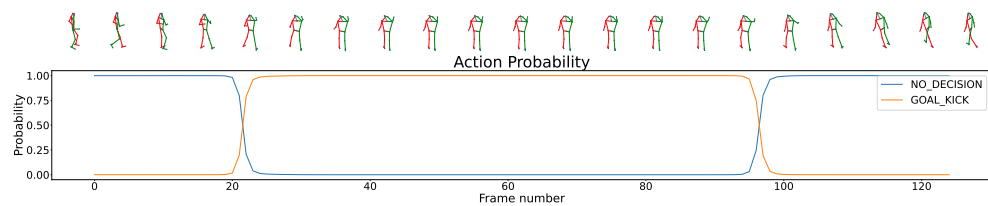


Figure 3.4: Output probabilities for a `GOAL_KICK`

CORNER_KICK Figure 3.5 shows the predicted probability over frames for a `CORNER_KICK`. Similar to the `GOAL_KICK`, the prediction response is quick, accurate, and stable.

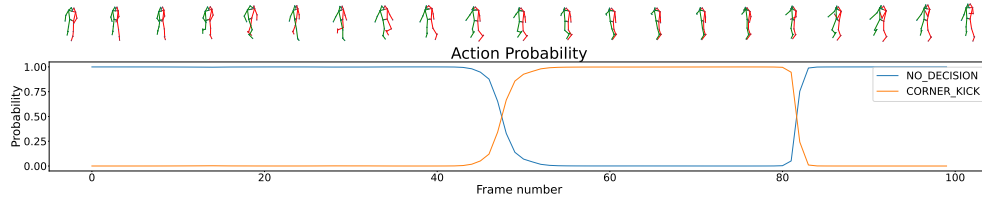


Figure 3.5: Output probabilities for a CORNER_KICK

SUBSTITUTION Figure 3.6 shows the predicted probabilities over frames for a SUBSTITUTION. The probability remains stable at one when the referee is showing the signal. During the transition, the model predicted a small probability for RAISE_FLAG_VERTICALLY, but this is insufficient to affect the overall prediction.

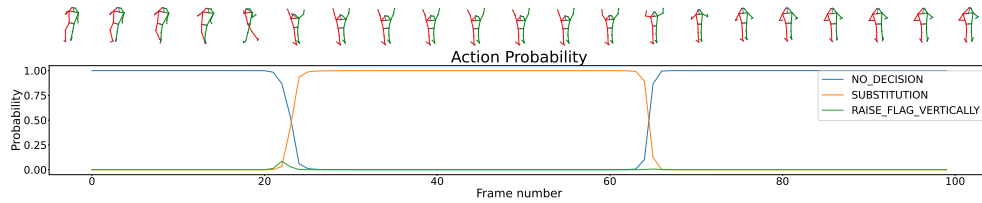


Figure 3.6: Output probabilities for a SUBSTITUTION

RAISE_FLAG_VERTICALLY In Figure 3.7, we show the probability of a RAISE_FLAG_VERTICALLY and the subsequent action. The figure illustrates that the duration is divided into five periods, each with a dominant prediction. The model provides stable predictions, transitioning from NO_DECISION to RAISE_FLAG_VERTICALLY, then back to NO_DECISION, followed by THROW_IN_LEFT, and finally returning to NO_DECISION.

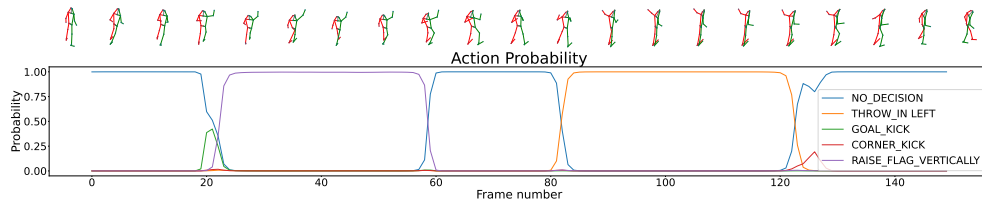


Figure 3.7: Output probabilities for a RAISE_FLAG_VERTICALLY and the subsequence THROW_IN_LEFT

Through case studies, we demonstrate that our model achieves not only high accuracy but also provides consistent and stable predictions, which are crucial for reliable action recognition in real-game scenarios.

3.3.6 Summary

Through experiments, we verify that single-frame predictions are sufficient for achieving good performance. Our proposed model performs well on the collected dataset and excels in real-game scenarios, highlighting its great value for practical applications. Our findings indicate that the pre-trained model contributes to final performance more from data collection than from providing general knowledge for fine-tuning. The carefully curated high-quality dataset is crucial for good model performance and is also beneficial for data collection of main referee action recognition, as will be discussed in the next chapter. Additionally, case studies further demonstrate that our model provides stable predictions, which is essential for real-time applications.

Chapter 4

Sequence-level Referee Action Recognition

4.1 Overview

Each football match is controlled by a referee with full authority to enforce the Laws of the Game in connection with the match [37]. The referee makes critical decisions on the field, such as awarding free kicks and penalties, determining fouls, as well as showing throw-ins, goal kicks, and corner kicks. They also manage the overall conduct of the game, including starting and stopping play, keeping track of the match time, and issuing yellow and red cards for disciplinary infractions. The decisions made by the referee are final and significantly influence the flow of the match. Therefore, accurately understanding the referee’s signals through skeleton data is crucial for understanding the dynamics of the game.

Compared to assistant referees, referees’ actions are dynamic due to their need to make quick, decisive moves across the entire pitch. The referee’s actions are usually in constant motion, and their signals often involve a combination of different gestures. For example, signaling a free kick involves blowing the whistle followed by an arm pointing in the attacking direction. To model this complex nature, sequence-level modeling of skeleton data is essential to provide the model with a broader context. Therefore, we propose a pipeline that utilizes a sequence-level model for action recognition of the main referee. Similar to our approach for assistant referees, this pipeline is designed in two stages but adapted to the more complex and specific nature of referee actions:

- **Pre-training** We propose pre-training a transformer-based masked

autoencoder that employs a multi-task pre-training objective, which combines motion prediction [33] and data2vec [36] strategies to learn representations from unlabeled skeleton sequences.

- **Task-specific Fine-tuning** The pre-trained model is fine-tuned for action recognition, incorporating additional information such as the ball position and the actions of assistant referees. We developed specialized schemes to integrate this extra information more effectively with the pre-trained model. In addition, we introduce a binary action-non-action classification model to provide token-level supervision, enhancing the effectiveness of the fine-tuning process.

Similar to the challenges faced with assistant referees, the scarcity of labeled data presents a significant challenge in developing effective classification models for main referees. To collect sufficient data that accurately represents the features of each action class, we have employed several strategies, including:

- Manual annotation
- Converting from frame-level data
- Identifying no-action sequences with the pre-training loss of the pre-trained model
- Fine-tuning a whistle-detecting model for data finding
- Using a set of rules conditioned on ball positions and arm angles

Evaluating the quality of the pre-trained model presents another significant challenge. While the model can be pre-trained on a large amount of unlabeled data, using pre-training loss on the validation set as a direct quality measure is problematic, as this metric can have different values with changes in preprocessing methods or pre-training tasks. Nevertheless, a reliable assessment of the pre-trained model is essential for comparing different pre-training techniques and ensuring continuous improvement. To address this, we propose employing linear evaluation on a sequence-level classification dataset as a robust metric to assess the effectiveness of the pre-training model.

Sequence-level skeleton-based action recognition is a topic extensively explored in academic research. Studies typically focus on recognizing actions in daily life, with widely used benchmarks such as NTU-RGB+D [34], NTU-RGB+D 120 [11], Kinetics [35], and PKUMMD [39]. In academic research,

a single prediction per skeleton sequence suffices. However, in the dynamic environment of football games, applying a sequence-level classification model to a live stream requires the use of a sliding window to provide contextual information, which can introduce significant delays. Ideally, to reduce this delay, the model should predict a label for each frame in the sequence, not just at the sequence’s end. For instance, the model should generate a prediction for an action on the frame where the action is actively occurring, and it should predict `NO_DECISION` after the action concludes, even if the action is still present within the context window. However, collecting a sequence-level dataset where each frame is individually labeled is nearly impractical. To address this challenge, we propose using a binary action-non-action classification model to supervise the fine-tuning process, automatically generating labels for each token and training with a token-level classification objective.

In the subsequent sections of this chapter, we will delve into the methodologies, implementation, and experiments for referee action recognition. We will provide a detailed introduction to our multi-task pre-training technique in Section 4.2.1, discuss the linear evaluation of pre-trained models in Section 4.2.2, and illustrate the specific designs for incorporating additional information and supervision during fine-tuning in Section 4.2.3. We will describe the post-processing techniques for real-world scenarios in Section 4.2.4. The strategies for data collection will be outlined in Section 4.3.1. Finally, we will present and analyze the results of the models trained using our proposed methods in the rest of Section 4.3.

4.2 Method

For simplicity, we denote our model as S-SPT_R (Sequence-level **S**keleton **P**re-trained **T**ransformer for **R**eferees) in the following sections.

4.2.1 Pre-training

4.2.1.1 Model Architecture

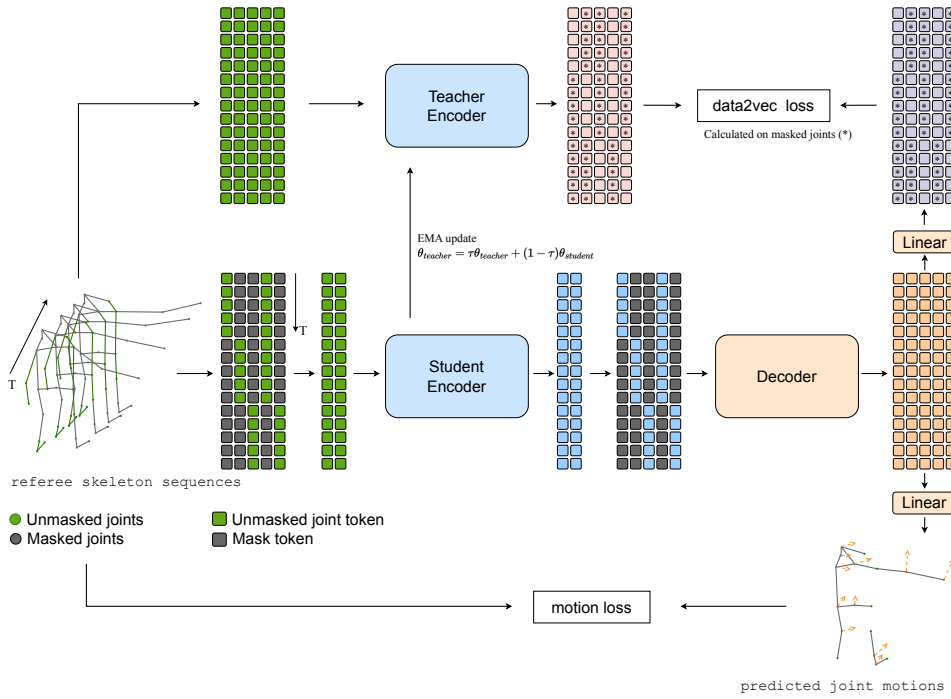


Figure 4.1: Overview of the multi-task pre-training pipeline of S-SPT_R. The two different losses are combined by a weighted sum for pre-training.

Similar to our strategy for assistant referees, we propose pre-training a transformer-based masked autoencoder based on the MAMP architecture [33], as outlined in Section 2.1.4.2. An overview of the pre-training process is shown in Figure 4.1. Initially, the sequence of skeleton frames is pre-processed and converted into a sequence of joint embeddings. The sequence is then partially masked according to specific strategies before being input into the encoder (Student Encoder in the figure), which generates a representation of the skeleton sequence based on solely unmasked tokens. Subsequently,

the decoder receives the representation with the mask token restored and reconstructs the original skeleton sequence with a multi-task pre-training objective. This objective includes two reconstructing targets: one is the motion of masked joints, and the other is a dense contextualized target calculated by a teacher model.

Unlike frame-level pre-training in the last chapter, which uses bidirectional attention, we propose using unidirectional attention to enhance frame-level prediction accuracy. To this end, we employ a causal mask on the attention mechanism, similar to those used in language models [51]. This mask prevents the token of the current frame from accessing future frames in the sequence during attention calculation, ensuring that each token is a representation solely based on preceding information. Without this causal mask, each token on the top layer would represent a composite of all input frames, complicating the distinction between them. Additionally, the causal mask ensures that the attention matrix is a triangular, full-rank matrix, thus fully utilizing the representation space.

By pre-training with sufficient unlabeled skeleton sequences, we aim for the model to capture the nature of human movement, particularly that of football referees, providing a robust foundation for subsequent fine-tuning.

4.2.1.2 Data Pre-processing

The original joint coordinates in skeleton sequences are measured in meters and centered around the pitch's midpoint, typically ranging from -50 to 50, depending on the pitch's size. To facilitate more stable and efficient neural network training, we preprocess these input values into a more restricted range. With specific normalization strategies, we scale most values to fall between -1 and 1. This preprocessing module is integrated directly into the model architecture, enabling the model to handle any input seamlessly in real-world scenarios. To prevent an excessive number of tokens within a single sequence, we segment the skeleton data into 10-second sequences for pre-training. The preprocessing strategies we employ for these sequences are outlined as follows:

Scaled When the referee is running fast, a 10-second skeleton sequence often spans tens of meters, with larger values observed near the pitch boundaries. To normalize these values into the range of -1 to 1 while preserving the relative ratio across different axes, we initially translate the entire skeleton sequence so that the first frame is centered at the origin. Subsequently, we scale the entire sequence by a factor that is determined by the axis with the largest variance. Formally, assume the original input is

$\mathbf{x} \in \mathbb{R}^{l \times V \times d}$, where l is the number of frames in a sequence; V is the number of joints in a frame; d is the dimension of the input which is 3. The sequence is first centered by translating the center point of the first frame to origin:

$$\begin{aligned} \overline{\mathbf{x}}_0 &= (\text{mean}(\mathbf{x}_{0,:,0}), \text{mean}(\mathbf{x}_{0,:,1}), \text{mean}(\mathbf{x}_{0,:,2})) \in \mathbb{R}^d \\ \mathbf{x}^{centered} &= \mathbf{x} - \overline{\mathbf{x}}_0 \end{aligned} \quad (4.1)$$

Then the variance along each axis is computed and the centered input is scaled by the largest variance:

$$\begin{aligned} \sigma_x^2 &= \text{variance}(\mathbf{x}_{::,0}^{centered}) \\ \sigma_y^2 &= \text{variance}(\mathbf{x}_{::,1}^{centered}) \\ \sigma_z^2 &= \text{variance}(\mathbf{x}_{::,2}^{centered}) \\ \sigma_{max}^2 &= \max(\sigma_x^2, \sigma_y^2, \sigma_z^2) \\ \mathbf{x}^{scaled} &= \mathbf{x}^{centered} / \sigma_{max} \end{aligned} \quad (4.2)$$

With scaled pre-processing, each sequence is guaranteed to be transformed to the range -1 to 1 and preserve the information of motion. However, this approach can lead to inconsistencies in the size of the skeleton within individual frames. For instance, the skeleton's size in a sequence where the referee covers a long distance will appear much smaller compared to a sequence where the referee is nearly stationary. This inconsistency may pose challenges to the representation learning process.

Pelvis-centered To focus on relative motions and preserve the consistent size of skeletons, we propose preprocessing each frame in the sequence to be pelvis-centered. The pelvis is one of the most stable parts of the body during movement and is located centrally within the human body, making it an ideal anchor point for representing relative motions. Formally, with the original input $\mathbf{x} \in \mathbb{R}^{l \times V \times d}$, assuming the index of pelvis joint is idx_{pelvis} , the pelvis-centered input is given by

$$\begin{aligned} \mathbf{x}^{pelvis} &= \mathbf{x}_{:,idx_{pelvis},:} \in \mathbb{R}^{l \times d} \\ \mathbf{x}^{pelvis_centered} &= \mathbf{x} - \mathbf{x}^{pelvis} \end{aligned} \quad (4.3)$$

In pelvis-centered preprocessing, we do not employ scaling operations, thereby avoiding confusion about skeleton sizes for the pre-training. However, this approach results in the loss of global movement information, which could pose a challenge for the pre-trained model in comprehensively understanding the movement dynamics.

4.2.1.3 Data Augmentation

We implement two data augmentation techniques to enhance the robustness of our pre-trained model. To maintain the quality of the pre-training data, we select augmentation methods that don't distort the skeletons. These methods include:

Random Length: For each input skeleton sequence, we randomly crop it to a shorter sequence for pre-training. The length of the cropped sequence is determined by sampling from a uniform distribution $U[l_{min}, l_{seq}]$, where l_{seq} is the original sequence length and l_{min} is the minimum allowable length. l_{min} is set to 25 in this work.

Random Rotation: The input sequence is rotated by a random angle around the z-axis. The rotation angle is chosen by sampling from a uniform distribution $U[-180^\circ, 180^\circ]$. We apply rotation exclusively along the z-axis, as this orientation is semantically reasonable while rotating along the x or y-axis could result in an unrealistic skeletal motion direction.

4.2.1.4 Joint Embedding

In most prior studies utilizing transformers for action recognition [110, 31, 76], each joint in each frame is converted into a single token, leading to a substantial total number of input tokens, which limits the ability to model long contexts. However, these tokens often exhibit significant temporal redundancy, with tokens that are temporally close together displaying high similarity. To reduce the number of tokens within a sequence for more efficient training, we followed the approach used in MAMP [33], where one token represents several consecutive frames for a single joint.

For the pre-processed input $\mathbf{x}_p \in \mathbb{R}^{l \times V \times d}$, we define a time patch size l_t and segment the input into $l_e = l/l_t$ non-overlapping segments $\mathbf{x}' \in \mathbb{R}^{l_e \times V \times l_t \times d}$. In each segment, each joint is converted into a token. A linear layer is then applied to increase the dimensionality of the input features. Additionally, a trainable joint embedding $J_e \in \mathbb{R}^{1 \times V \times h}$ is incorporated to represent information specific to each joint type, and a learnable positional encoding $P_e \in \mathbb{R}^{l_e \times 1 \times h}$ is added to provide temporal context. Finally, a layer normalization is applied to ensure stable training. The Embedding $E \in \mathbb{R}^{l_e \times V \times h}$ (h is the dimension of embedding space) is given by

$$E = \text{LayerNorm}(\text{Linear}(\mathbf{x}') + J_e + P_e) \quad (4.4)$$

4.2.1.5 Motion-aware Tube Masking

Skeleton sequences feature high spatiotemporal correlation. Following Skeleton2vec [54], we utilize a motion-aware tube masking strategy to address this issue.

Tube Masking VideoMAE [111] initially proposed tube masking to address spatiotemporal issues in videos by treating the entire video as a single tube and applying a uniform masking map across all frames. This approach effectively minimizes information leakage between adjacent frames. In video contexts, this fixed masking is not problematic as objects typically move across pixels, aiding in the modeling of long-term dependencies. However, for skeleton sequences, where the coordinates of body parts remain in a fixed position within the representation, applying a fixed mask map can result in a body part being masked throughout the entire sequence. This makes it extremely challenging for the model to infer the masked joint from no available data. To address this, Xu et al. [54] propose using multiple tubes for tube masking. The mask map remains consistent within each tube but varies across different tubes, ensuring that no joint is completely masked out. With the number of tubes denoted as n_{tube} , the tube division can be described as $E' \in \mathbb{R}^{n_{tube} \times l_{tube} \times V \times h}$, where $l_{tube} = l_e / n_{tube}$ is the tube length.

Motion-aware Masking Empirically, joints with larger motion contain richer semantic information. To leverage this, we adopt the concept of motion-aware masking as detailed in Section 2.1.4.2. This approach is further refined by integrating tube masking, where we calculate the average motion within each tube and subsequently perform sampling based on the motion. Formally, for each tube, we first calculate the motion $M^{tube} \in \mathbb{R}^{(l_{tube}-1) \cdot l_t \times V \times d}$ with its corresponding original input $x^{tube} \in \mathbb{R}^{(l_{tube} \cdot l_t) \times V \times d}$:

$$M^{tube} = x_i^{tube} - x_{i-l_t}^{tube} \quad (4.5)$$

where l_t is the time patch size. Then the joint-wise motion intensity within the tube $I^{tube} \in \mathbb{R}^V$ is calculated:

$$I_i^{tube} = \text{mean}(M_{:,i,:}^{tube}) \quad (4.6)$$

Similar to the approach in MAMP [33], we sample from the intensity values,

applying a temperature parameter τ to regulate the sampling rate.

$$\begin{aligned}\pi &= \text{softmax}(I^{tube}/\tau) \\ \eta &\sim U[0, 1]^V \\ idx^{mask} &= \text{Index-of-Top-K}(\log \pi - \log(\log \eta))\end{aligned}\tag{4.7}$$

where η is random noise drawn from a uniform distribution from 0 to 1; K represents the rounded number corresponding to the mask ratio α_{mask} applied across all joints: $K = \text{round}(\alpha_{mask} \cdot V)$. We set α_{mask} as 0.9, following the best practice from prior works [33, 54]. The unmasked part $E_u^{tube} \in \mathbb{R}^{l_{tube} \times K \times h}$ from the embeddings E^{tube} is extracted as the input to the encoder according to idx^{mask} . This process is repeated for each tube. Within the same tube, we apply the same sampled mask map.

4.2.1.6 Multi-task Pre-training

We propose pre-training the masked autoencoder with a multi-task pre-training objective comprising a motion prediction task [33] and a data2vec task [36]. In this setup, the motion prediction task aims to reconstruct the original motion of the masked joints, while the data2vec task utilizes a teacher model, which is an exponentially moving averaged version of the pre-trained model, to provide the prediction target for the masked joints in the latent space. data2vec is a general pre-training strategy, not dependent on specifically designed prediction targets or data-augmentation methods, and has demonstrated effectiveness in both vision and language modeling [36]. Xu et al. [54] have further demonstrated its applicability and effectiveness in skeleton-based action recognition.

However, the data2vec objective solely involves self-learning, which poses challenges in the early stages of training. Since the prediction targets are generated by the teacher model, they may not contain meaningful information when the model is not yet well-trained. Training with the data2vec loss typically progresses more slowly and requires more training epochs. Additionally, the self-learning teacher-student design carries the risk of converging to a collapsed representation. To address these challenges, we propose integrating these two pre-training objectives. The motion prediction task enables the pre-trained model to quickly learn meaningful representations with minimal risk of collapse. Simultaneously, the data2vec objective enhances these representations by providing a context-based prediction target in the latent space. The integration of these objectives and their implementation details will be further discussed in the paragraphs that follow.

Motion Prediction An additional linear layer is added to the decoder to predict the original motion of the masked joints, as shown in Figure ???. For the output of the decoder $Z_d \in \mathbb{R}^{l_e \times V \times h}$, the predicted motion is given by

$$M^{pred} = \text{Linear}(Z_d) \in \mathbb{R}^{l \times V \times d} \quad (4.8)$$

The prediction target motion $M^{target} \in \mathbb{R}^{l \times V \times d}$ is calculated from the original input \mathbf{x} using a stride l_t , the same value as the time patch size used in the pre-trained model. The first l_t steps of the motion target are padded with zeros.

$$M_i^{target} = \begin{cases} 0 & \text{if } i < l_t \\ \mathbf{x}_i - \mathbf{x}_{i-l_t} & \text{otherwise} \end{cases} \quad (4.9)$$

Following MAMP [33], the motion prediction loss for pre-training is defined as an L2 loss between M^{pred} and M^{target} on masked joints:

$$\mathcal{L}_{motion} = \frac{1}{|idx^{mask}|} \sum_{(i,j) \in idx^{mask}} \left\| \left(M_{i,j,:}^{pred} - M_{i,j,:}^{target} \right) \right\|_2^2 \quad (4.10)$$

data2vec The data2vec pre-training objective involves using a teacher model to calculate a contextualized representation as the prediction target. This teacher model is updated through an exponential moving average of the pre-trained model (student model). Specifically, the teacher shares the same architecture as the student model and is initialized with the same parameters as the student model. At each updating step, the parameters of the student model are updated by gradient descent, while the parameters of the teacher model are updated through an exponential moving average controlled by a factor τ .

$$\theta_{teacher} = \tau \theta_{teacher} + (1 - \tau) \theta_{student} \quad (4.11)$$

We set the factor τ as 0.9999 in our experiments. As only the encoder part is used for fine-tuning in action recognition, we exclusively utilize a teacher encoder model working with a student encoder model, as shown in Figure ??.

Following data2vec [36] and Skeleton2Vec [54], we extract the output of the FFN block before residual in each transformer layer from the teacher encoder and compute an average over them to form our training target. The features extracted from each layer are processed by instance normalization before averaging. A layer normalization is applied after averaging. Normalization helps prevent the model from learning trivial

representations.

$$\begin{aligned} R' &= \frac{1}{n_e} \sum_{l=1}^{n_e} \text{InstanceNorm}(Z_l) \\ R^{target} &= \text{LayerNorm}(R') \end{aligned} \quad (4.12)$$

where n_e is the number of encoder layers, Z_l is the output of the FFN block before residual from the l -th transformer layer.

Similar to motion prediction, we add an additional linear layer for data2vec target prediction. For the decoder output Z_d ,

$$R^{pred} = \text{Linear}(Z_d) \quad (4.13)$$

The pre-training objective for the data2vec target is an L2 loss, calculated between R^{pred} and R^{target} , also applied solely to the masked joints.

$$\mathcal{L}_{data2vec} = \frac{1}{|idx^{mask}|} \sum_{(i,j) \in idx^{mask}} \| (R^{pred} - R^{target}) \|_2^2 \quad (4.14)$$

Combination of two objectives The final learning objective for the multi-task pre-training is formulated as a weighted sum of the two pre-training losses:

$$\mathcal{L} = \alpha \mathcal{L}_{motion} + \beta \mathcal{L}_{data2vec} \quad (4.15)$$

where α and β are hyperparameters that control the contribution of each loss component. In this work, the weights are set as $\alpha = 0.95, \beta = 0.05$.

4.2.2 Linear Evaluation

To evaluate the quality of the pre-trained model and provide a metric for continuously refining pre-training methods, we employ linear evaluation as a measure of pre-trained model performance. In this method, a single linear layer is added on top of the pre-trained encoder for a sequence classification task, while the decoder is not utilized. Specifically, we apply the linear layer to conduct an 8-way sequence classification task. The task data is converted from frame-level assistant referee data, as detailed in Section 4.3.1.2. A batch normalization layer is added between the encoder output and the linear layer. The prediction probability is then computed as follows:

$$p = \text{softmax}(\text{Linear}(\text{BatchNorm}(Z_e^{(-1)}))) \quad (4.16)$$

where $Z_e \in \mathbb{R}^{l_e \times V \times h}$ represents the output of the pre-trained encoder, and $Z_e^{(-1)} \in \mathbb{R}^{V \times h}$ represents the output corresponding to the last time step. The model is trained with a cross-entropy loss. In linear evaluation, the pre-trained model remains fixed, with only the linear layer being learnable. This restriction limits the model's learning capabilities, making its performance heavily dependent on the quality of the sequence representation produced by the fixed pre-trained encoder. We use the SGD optimizer for linear evaluation. The learning rate follows a cosine decay schedule, decreasing from 0.1 to 0 over 10 training epochs. The batch size is set to 16.

4.2.3 Fine-tuning for Referee Action Recognition

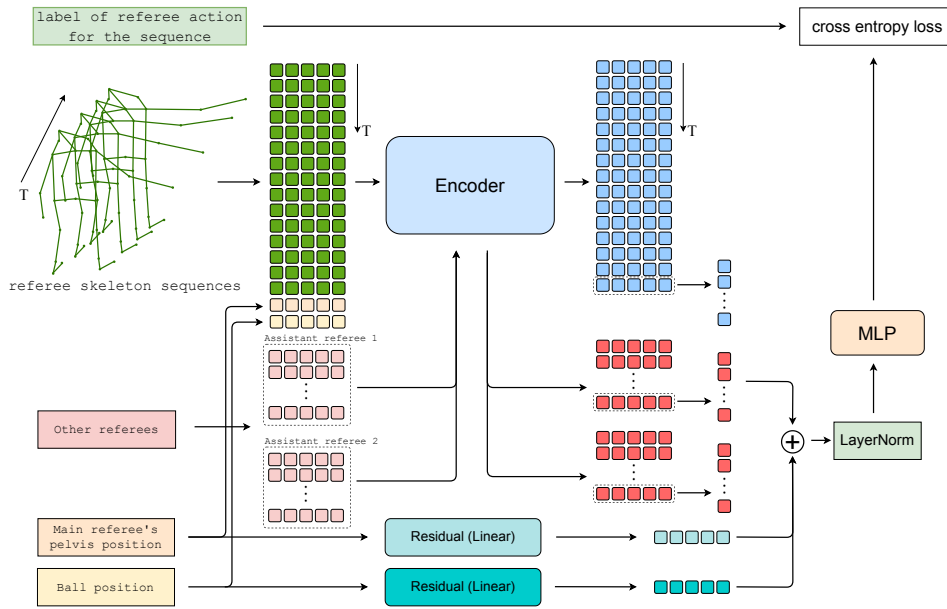


Figure 4.2: Overview of the fine-tuning process of S-SPT_R. The student encoder in the pre-training phase is used for fine-tuning.

4.2.3.1 Targeting Action Types

As discussed in Section 2.3, referee actions include both standardized signals, such as goal kicks and corner kicks, and non-standardized signals, such as no foul and substitution. The latter is less likely to disrupt the flow of the game and is more difficult to collect data in practice. Given the scale of our eventually collected dataset, as detailed in Table 4.5, we propose to merge these less frequent categories into a single class, `OTHER_DECISION`. This

approach helps to differentiate these actions from the `NO_DECISION` action and mitigates the issue of insufficient labeled data. Consequently, we employ a 9-way classification system in our referee action recognition. The specific action types targeted are listed in Table 4.1.

Targeting Referee Actions

`NO_DECISION`
`DIRECT_FREE_KICK`
`INDIRECT_FREE_KICK`
`THROW_IN_HOME`
`THROW_IN_AWAY`
`GOAL_KICK`
`CORNER_KICK`
`GOAL`
`OTHER_DECISION`

Table 4.1: Referee action types in our classification task

4.2.3.2 Integration of Additional Information

Referee actions are more diverse and complex compared to assistant referees. Referee signals conveying different meanings could look similar in terms of skeleton sequences, with only minor differences depending on the context. For example, a referee’s arm pointing in a particular direction could indicate a throw-in, a direct free kick, or something entirely different. To enhance the model’s ability to accurately differentiate between different actions, additional information is crucial. For instance, if the ball is out of play and the assistant referee signals for a throw-in, it is more likely that the main referee’s gesture is also a throw-in. Conversely, in the absence of such contextual clues, the referee’s arm pointing to a direction might more likely indicate a direct free kick. To improve referee action recognition, we propose integrating the following additional information:

- **Absolute Position of the Referee** The input for the pre-trained model is preprocessed to be either first-frame-centered or pelvis-centered, making the model unaware of the referee’s absolute position on the pitch. However, the referee’s position is critical for accurate action recognition. For instance, the likelihood of a referee signaling a goal kick or corner kick while positioned in the center area of the pitch is extremely low. To address this, we incorporate the sequence of absolute pelvis positions,

represented as $\mathbf{x}_{pelvis} \in \mathbb{R}^{l \times d}$, as an additional input to represent the referee's absolute position.

- **Ball Position** The position of the ball is crucial for helping identify different referee signals. For instance, when the ball goes out of the pitch, it is more likely to be a throw-in, goal kick, or corner kick. To integrate ball position information for better decision, we represent the ball's position as its 3D coordinates in each frame, denoted by $\mathbf{x}_{ball} \in \mathbb{R}^{l \times d}$.
- **Other Referees** The actions of the two assistant referees could also enhance the action recognition of the main referee. By including the skeleton sequences of all three referees as inputs, our model may better recognize actions through the interactions observed among the referees. The actions of the assistant referees could provide valuable context for interpreting the main referee's actions. We represent the additional input for the two assistant referees as their skeleton sequences, denoted by $\mathbf{x}_{ar} \in \mathbb{R}^{2 \times l \times V \times d}$.

The method of integrating information from the assistant referees is straightforward. Given that the skeleton data for each referee is structured the same, we employ the same pre-trained encoder module to process the input skeletons. The final representation is achieved by concatenating the representations of all three referees.

$$\begin{aligned}
 Z_e^r &= \text{Encoder}(\mathbf{x}_r) \\
 Z_e^{ar1} &= \text{Encoder}(\mathbf{x}_{ar1}) \\
 Z_e^{ar2} &= \text{Encoder}(\mathbf{x}_{ar2}) \\
 Z_e &= \text{concat}(Z_e^r, Z_e^{ar1}, Z_e^{ar2})
 \end{aligned} \tag{4.17}$$

where \mathbf{x}_r , \mathbf{x}_{ar1} , \mathbf{x}_{ar2} represent the input skeleton for the referee, the first assistant referee, and the second assistant referee, respectively. The weights of the encoder are shared for all three referees.

The additional inputs of the ball positions and pelvis positions, due to their lower dimensionality compared to the encoder's output representation space, cannot be directly processed by the encoder or straightforwardly added to the encoded representation. To integrate this information for action recognition, we propose two strategies:

- **Residual Module** A straightforward method for integrating the additional input into the encoded representation is to map it to the same

dimension as the encoder output with a linear layer. After summing up the representation, a layer normalization is applied to ensure stable training.

$$Z_e^{res} = \text{LayerNorm}(\text{Linear}(x_{pelvis}) + \text{Linear}(x_{ball}) + Z_e) \quad (4.18)$$

- **Extra token in attention** A residual module works well for building representations with fused information. However, it falls short in interactively modeling the relationship between the ball, the referee's absolute position, and their actions. To better capture the interactions between referee gestures and the additional information, we propose creating specific tokens for these additional inputs. These tokens are then fed into the pre-trained encoder along with the skeleton sequence, allowing the attention mechanism within the transformer layers to model the interactions. To create tokens for the additional inputs, we follow the same strategy as that in joint embedding, using a linear layer followed by a layer normalization. The expanded input embedding is given by

$$\begin{aligned} E &= \text{JointEmbedding}(x) \\ E_{pelvis} &= \text{LayerNorm}(\text{Linear}(x_{pelvis})) \\ E_{ball} &= \text{LayerNorm}(\text{Linear}(x_{ball})) \\ E_{extra} &= \text{concat}(E, E_{pelvis}, E_{ball}) \end{aligned} \quad (4.19)$$

Note that these two strategies are not mutually exclusive. They can be applied simultaneously or individually. We show an overview of the fine-tuning process leveraging both strategies in Figure 4.2. We conducted extensive experiments to evaluate these approaches, as detailed in Section 4.3.3.2. The results indicate that using only the residual module for integrating additional information yields optimal performance.

4.2.3.3 Fine-tuning with Sequence Classification Task

A common strategy to adapt a pre-trained model for a specific recognition task is by fine-tuning it through a sequence classification task. In this setup, the model is required to predict a single label for each input sequence. To achieve this, we add a Multi-Layer Perceptron (MLP) on top of the student encoder output to compute the classification probabilities. The model is fine-tuned on the sequence classification dataset collected for referees, as will be described in Section 4.3.1.3. Due to the use of the causal mask in the attention mechanism,

only the representation from the last time step, denoted as $Z_e^{(-1)} \in \mathbb{R}^{V \times h}$, represents the entire sequence. The classification probability is calculated as follows:

$$\begin{aligned} H_{MLP} &= \text{ReLU}(\text{Linear}(Z_e^{(-1)})) \\ p &= \text{softmax}(\text{Linear}(H_{MLP})) \end{aligned} \quad (4.20)$$

The hidden layer's dimension matches the hidden size h of the pre-trained model. When inputs from all three referees are processed, the input dimension for the MLP, $Z_e^{(-1)}$, expands to $V \times 3h$, and the MLP's hidden size is adjusted to $3h$ accordingly. A cross-entropy loss is used for fine-tuning.

During fine-tuning, the entire model remains trainable, providing maximum flexibility to adapt the pre-trained model to the recognition task. Unless otherwise specified, we fine-tune the model using the AdamW optimizer for 10 epochs. The learning rate increases linearly from 0 to $5e-5$ over the first 1000 steps and then follows a cosine schedule to decay to zero. The batch size is set at 16.

4.2.3.4 Fine-tuning with Sequence Labeling Task

A significant limitation of the model fine-tuned with the sequence classification task is its delayed response in real-time scenarios. This delay comes from the requirement of a sliding context window to generate predictions. As long as an action remains within this window, predictions continue, often even after the action has ended. This delay affects the model's real-time performance and does not align with human intuition.

To address this challenge, we propose fine-tuning the model with the sequence labeling task, where each token in the time dimension gives its own prediction. This approach ensures that predictions correspond to the action type of each specific time step rather than the entire context sequence, effectively addressing the issue of delays. Additionally, fine-tuning with a sequence labeling task allows for more effective use of training data and the incorporation of longer contexts during training, enhancing overall model performance.

However, collecting a labeled dataset for the sequence-labeling task is extremely challenging because it requires multiple labels for each sequence. Instead of creating a new dataset for sequence labeling, we propose using the existing dataset designed for sequence classification, employing a binary classification model to offer frame-level supervision. Specifically, the binary classification model provides a prediction on each frame whether there is an action ongoing or not. This prediction is then combined with the original label

of the sequence to provide supervision for each frame, as shown in Figure 4.3.

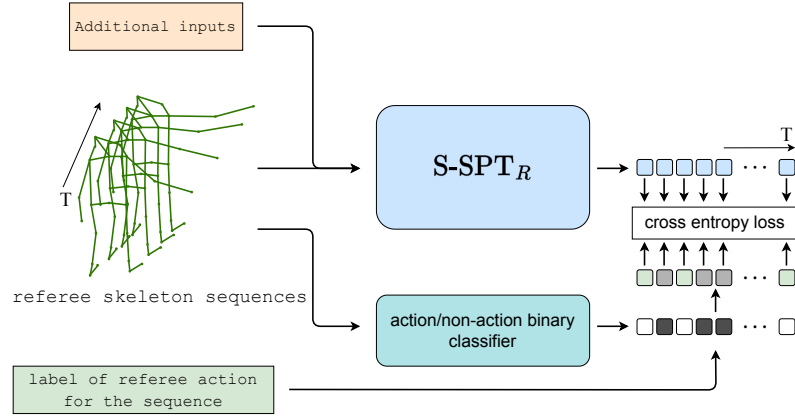


Figure 4.3: Fine-tuning the model with the sequence labeling task under supervision from a binary classification model

Training binary classification model An accurate binary classification model is essential for providing high-quality supervision. To train this model, we collected a sequence classification dataset where each sequence is one second long. The dataset collected is shown in Table 4.2. We employ the fine-tuning strategy outlined in Section 4.2.3.3 to fine-tune the binary classification model. For additional information, only absolute pelvis data is utilized, and integration of this data is achieved exclusively through the use of a residual module. The batch size for training is set at 64. The trained binary classification model achieves an accuracy of 98.66% on the validation set and 98.54% on the test set. Based on these results, we assert that this model is sufficiently accurate to provide frame-level supervision for training in sequence labeling.

	Train	Valid	Test
NO_DECISION	33,770	3,077	3,086
HAS_DECISION	28,222	2,385	2,376
Total	61,992	5,462	5,462

Table 4.2: Dataset collected for training binary classification model to provide token-level supervision. Each data point is a one-second-long referee skeleton sequence.

Supervision from binary classification model For each frame, we use

the binary classification model to predict p_{binary} based on the preceding one-second sequence. This p_{binary} represents the likelihood that the frame is part of an ongoing action. To generate a label for each frame, there are two methods for combining the p_{binary} predictions with the original sequence labels:

- **Hard labels** Discrete labels are generated for each frame based on the predictions. Specifically, if p_{binary} exceeds 0.5, it indicates that an action is occurring, and the original sequence label is assigned to that frame. Otherwise, the frame is considered to have no action, and the label NO_DECISION is assigned.
- **Soft labels** Instead of a discrete label, the label is represented as a probability distribution. Specifically, the probability that a frame is labeled as NO_DECISION is calculated as $(1 - p_{binary})$, and the probability for the original sequence label is given by p_{binary} . The probabilities of all other classes are set to zero.

Pre-sequence random-length masking Since the sequence labeling task involves predicting a label at each timestep and is less sensitive to context length, we propose implementing data augmentation through random-length masking at the beginning of the sequence. This strategy can introduce more varied contexts during fine-tuning, contributing to a more robust model. Specifically, for each skeleton sequence, we randomly determine a masking length, $l_{premask}$, by sampling from a uniform distribution: $l_{premask} \sim U[0, l_{premask}^{max}]$, where $l_{premask}^{max}$ represents the maximum feasible length to mask. In this work, $l_{premask}^{max}$ is set to 50, which corresponds to a 2-second span of the sequence, with the frame rate set at 25 Hz. The initial $l_{premask}$ frames are then masked from the inputs for the subsequent computation.

A Cross-entropy loss is used for fine-tuning with the sequence labeling task. At the beginning of each sequence, the context may not be sufficient for the binary classification model or the target model to make accurate classifications. As a result, we do not calculate cross-entropy loss for all tokens. Instead, we define a starting point within the sequence and only compute the loss for tokens that occur after this point. In our experiments, this starting point is consistently set at the 2-second mark of each sequence.

4.2.4 Post-processing

We employ similar post-processing techniques as described in Section 3.2.3 for frame-level action recognition to improve accuracy and minimize false positives in real-time scenarios. Unlike the method used for assistant referees,

which involves a two-stage prediction model, our approach for referee action recognition utilizes just one model. Consequently, we apply only the duration threshold and the confidence threshold. To predict a frame, a fixed-length sliding window is used to provide context for action recognition. The duration threshold is set at 0.5 seconds, and the confidence threshold is established at 0.8. These thresholds are chosen for a balance between responsiveness and accuracy, ensuring that the model reliably identifies significant actions without being overly sensitive to minor movements.

4.3 Experiments and Results

4.3.1 Data Preparation

4.3.1.1 Pre-training Data

We collected a large-scale dataset for sequence-level pre-training, using the same data source as the frame-level pre-training data. The pre-training skeletons are from 223 games, with 28 of these games designated solely for validation and testing. The division remains the same as that in the frame-level pre-training. We only load skeletons for referees and assistant referees, omitting player skeletons, to focus on learning the movement patterns of referee actions. The skeleton data is segmented into sequences, each 10 seconds long. Since skeleton sequences from different games may vary in frame rate, leading to different input shapes, we standardize the frame rate to 25Hz. This is achieved through linear interpolation, which not only normalizes the frame rate but also helps reduce the occasional occurrence of NaN values in the data. The choice of the sequence length is a balance between providing rich context and maintaining practicality within computational resource constraints. Since our frame rate is 25 Hz and the pre-trained model is trained with a time patch size of 5, every five consecutive frames are converted into a single token for each joint. With the number of joints being 21, this process results in a total of 1,050 tokens per sequence.

To enhance the quality of the pre-training data, we propose implementing denoising techniques to address three specific types of noise, namely spread-noisy sequences, motion-noisy sequences, and pelvis-centered motion-noisy sequences. Refer to Appendix B.1 for details of the pre-training data cleaning.

After filtering out noisy skeleton sequences using the aforementioned criteria, we have collected a high-quality, large-scale dataset for sequence-level pre-training. The statistics of this dataset are shown in Table 4.3. The

training set includes approximately 1,000 hours of referee movements. Of the 28 evaluation games, three have been selected for the validation set, consistent with the selections made for frame-level pre-training.

	# 10s Sequences
Training set	350,289
Validation set	6,489

Table 4.3: Statistics of sequence-level pre-training data

4.3.1.2 Dataset for Linear Evaluation

To facilitate the development of pre-training techniques, we established an evaluation criterion based on linear evaluation using a sequence classification dataset. For rapid implementation, we converted the frame-level classification dataset to sequence-level, maintaining both data quality and scale. For each labeled frame, we extract 3-second skeleton sequences for all three referees, centering the sequence on this frame. The frame’s label is then applied to the sequence. Additionally, we have refined the labels to provide more detailed classifications based on the directions of subsequent attacks, thereby increasing the task’s complexity. These labels are determined according to rules based on the positions of the referees. The frame rate is set to 25 Hz, resulting in a sequence of 75 frames per input, capturing the movements of all three referees. To ensure a fair evaluation, we have balanced the labels in both the validation set and the test set. Detailed information about the dataset is provided in Table 4.4.

	Train	Valid	Test
THROW_IN_HOME	2220	43	43
THROW_IN_AWAY	2178	43	43
GOAL_KICK_HOME	1083	43	43
GOAL_KICK_AWAY	1665	43	43
CORNER_KICK_HOME	529	43	43
CORNER_KICK_AWAY	871	43	43
SUBSTITUTION	833	43	43
NO_DECISION	6183	43	43

Table 4.4: Sequence classification dataset for linear evaluation. Each sequence is 3 seconds long.

4.3.1.3 Sequence Classification Dataset for Fine-tuning

The quality of the dataset used for fine-tuning is crucial for the optimal performance of the final recognition model. To achieve the best results, various strategies are employed during the data collection process. These strategies aim to enhance the richness and diversity of the dataset, ensuring that the samples in each class are adequately representative. By implementing specific methodologies for gathering data, we can significantly enhance the model's capabilities. The strategies employed include:

Manual Annotation Manual annotation is more essential for sequence classification tasks, especially for classes like `DIRECT_FREE_KICK` and `OTHER_DECISION`, due to the challenge of designing reliable automatic rules. Therefore, we use a semi-automated approach, combining automated detection with manual verification to balance efficiency and ensure data quality.

Converting from Frame-level Data Frame-level data for assistant referees can be reused and converted to sequence-level data for referees, as they often react simultaneously to events. This provides a large amount of labeled data for `THROW_IN`, `GOAL_KICK`, and `CORNER_KICK`. Additionally, for collecting `INDIRECT_FREE_KICK` sequences, we first collect frames for `RAISE_HAND_VERTICALLY` and then convert them into sequences.

Inlier Detection with Pre-training Loss To annotate `NO_DECISION` sequences, we detect inliers using the pre-training loss with the pre-trained model. The idea is to mask all the arms throughout the sequence and calculate the pre-training loss of masked arm joints with the pre-trained model. Since most data during pre-training is without referee signals, sequences that don't contain referee signals are likely to have a lower pre-training loss. Thresholds on this loss are set to annotate *`NO_DECISION` sequences*.

Binary Whistle Classification Model We fine-tune a binary classification model based on the pre-trained model to detect whether the referee blows the whistle. This model is used to help annotate `DIRECT_FREE_KICK` sequences.

Combination of Rules We leverage football-specific knowledge and establish a set of rules to identify `GOAL` sequences. The rules involve first detecting a kickoff, then tracing back to the nearest instance where

the referee raises their arm at a certain angle and the ball is near the goal area.

Collecting OTHER_DECISION In addition to the collected action types, various referee signals exist. Due to sparse data, creating separate classes is impractical. We propose merging these signals into a single class, OTHER_DECISION, to distinguish them from main actions and NO_DECISION.

Expansion for the Sequence-labeling Task Context can be longer when the model is trained with the sequence-labeling task, as discussed in Section 4.2.3.4. We expand the existing 4-second sequences to 8 seconds to provide richer contextual information.

Please Refer to Appendix B.2 for the details. With all the aforementioned strategies, we have collected a sequence classification dataset for 9-way referee action recognition, as shown in Table 4.5. Each sequence in the dataset is 4 seconds long. We split the dataset into a training set, validation set, and test set, with the latter two derived from games specifically designated for evaluation.

	Train	Valid	Test
NO_DECISION	1815	200	195
DIRECT_FREE_KICK	109	57	64
INDIRECT_FREE_KICK	1865	117	128
THROW_IN_HOME	916	72	93
THROW_IN_AWAY	1005	78	86
GOAL_KICK	1435	118	129
CORNER_KICK	844	65	36
GOAL	355	55	54
OTHER_DECISION	180	36	39

Table 4.5: Final sequence-level classification dataset collected for referee action recognition. Each sequence is 4 seconds long.

4.3.2 Pre-training

4.3.2.1 Network Architecture

The network to pre-train on is a transformer-based autoencoder, as introduced in Section 4.2.1. It includes 8 encoder layers and 1 decoder layer. The model

features a hidden size of 512 with 8 attention heads. The dimension of the feed-forward layer is set to 2048. The time patch size is set at 5. We use a mask ratio of 0.9, with the number of masking tubes set to 10 and the temperature τ set to 0.75.

4.3.2.2 Pre-training Details

During pre-training, we use the AdamW optimizer with betas set at (0.9, 0.95) and a weight decay of 0.01. The model is pre-trained for 700,000 steps. We utilize a cosine annealing learning rate schedule with a linear warmup, where the peak learning rate is set at $5e-5$, and the number of warmup steps is set at 10,000. The batch size is 32. We train the model on a single Nvidia A4000. Dropout rates of 0.1 are applied to both the attention matrix and the hidden layers.

4.3.2.3 Main Results

We evaluate the pre-trained model with the linear evaluation protocol, as described in Section 4.2.2. In this setup, the backbone of the pre-trained encoder remains fixed, and only the newly added linear layer is trained on the dataset gathered for linear evaluation, as detailed in Section 4.3.1.2. For the pre-trained model, we employ the pelvis-centered pre-processing method using the multi-task pre-training objective with coefficient $\alpha = 0.95, \beta = 0.05$. The linear layer is trained for only 10 epochs. We use a learning rate of 0.1 and a batch size of 16. The learning rate follows a cosine annealing schedule to 0 without warmup stages. An SGD optimizer with momentum is employed. No dropout or weight decay is implemented during this phase. We evaluate the model on the validation set every 100 steps, and the performance on the test set is based on the best-performing model on the validation set.

As shown in Table 4.6, with appropriate preprocessing methods and pre-training tasks, the pre-trained model achieves an accuracy of 90.38% on the validation set and 90.12% on the test set. For comparison, we also conducted a linear evaluation on a model with the same network architecture but without pre-training, as shown in the first line of the table. The results demonstrate that the pre-trained model significantly outperforms the non-pre-trained model. This suggests that the pre-trained model has learned a more effective representation of referee skeleton sequences, reducing the difficulty of the classification task with only one linear layer.

	#Params	Valid Accuracy	Test Accuracy
S-SPT _R w/o pre-training	28.9M	73.18	70.93
S-SPT _R	28.9M	90.38	90.12

Table 4.6: Performance of the pre-trained model with the linear evaluation protocol.

4.3.2.4 Ablation Study

We conduct ablation studies to evaluate the impact of various components and configurations on the performance of pre-training. All the ablation studies utilize the linear evaluation protocol, maintaining the same setup as previously employed.

Impact of Pre-processing Methods and Pre-training Tasks We evaluate the techniques we propose in Section 4.2.1 by trying different combinations of these pre-processing methods and pre-training tasks with the same linear evaluation settings. The results are presented in Table 4.7. As we can see from the table, pre-training with only motion prediction loss (line 3) or only data2vec loss (line 2) yields suboptimal performance compared to a multi-task pre-training approach. Regarding pre-processing methods, when all other settings are consistent, scaled pre-processing (line 4) does not perform as well as pelvis-centered pre-processing (line 3). This suggests that pelvis-centered pre-processing offers a more effective input for pre-training.

Pre-processing Method		Pre-training Task		Valid Acc.	Test Acc.
Scaled	Pelvis-centered	Motion	data2vec		
	✓	✓	✓	90.38	90.12
	✓		✓	86.01	84.59
	✓	✓		79.88	77.62
✓		✓		74.64	75.87

Table 4.7: Ablation study on pre-processing method and pre-training tasks.

Impact of Loss Coefficient We also assess the impact of the coefficients used in the multi-task pre-training loss. As introduced in Section 4.2.1.6, the multi-task pre-training loss is defined as $\mathcal{L} = \alpha\mathcal{L}_{\text{motion}} + \beta\mathcal{L}_{\text{data2vec}}$. We adjust the values of the coefficients α and β and use the linear evaluation to determine the most effective combination of these two pre-training losses. The findings are shown in Table 4.8. The results demonstrate that the balance

between pre-training loss components significantly influences the model’s learning effectiveness. When the coefficient for motion prediction is too low (line 1), the data2vec task becomes dominant, leading to suboptimal results. Conversely, when the motion prediction loss dominates with $\alpha = 0.99$ (line 4), performance also declines. An optimal combination of the two pre-training losses with $\alpha = 0.95, \beta = 0.05$ yields the best pre-training performance.

Coeff. α for \mathcal{L}_{motion}	Coeff. β for $\mathcal{L}_{data2vec}$	Valid Acc.	Test Acc.
0.5	0.5	82.22	80.23
0.9	0.1	89.80	89.24
0.95	0.05	90.38	90.12
0.99	0.01	88.34	88.08

Table 4.8: Ablation study on the coefficient of multi-task pre-training components. The multi-task loss is defined as $\mathcal{L} = \alpha\mathcal{L}_{motion} + \beta\mathcal{L}_{data2vec}$.

4.3.3 Fine-tuning for Referee Action Recognition

4.3.3.1 Fine-tuning Details

We adapt our pre-trained model for referee action recognition through fine-tuning, as detailed in Section 4.2.3. During this process, a two-layer MLP with an intermediate dimension of 512 is added on top of the pre-trained encoder. For fine-tuning, we utilize an AdamW optimizer with a weight decay of 5e-4. The betas for the AdamW optimizer are set at (0.9, 0.999). The model is fine-tuned over 20 epochs, starting with a 1000-step linear warmup to a learning rate of 5e-5, followed by a cosine annealing decay to zero. The batch size is set at 16. A dropout layer with a probability of 0.5 is applied to the encoder output to prevent overfitting.

4.3.3.2 Results with Sequence Classification Task

In line with standard practices in skeleton-based action recognition, we fine-tune the pre-trained model with the sequence classification task, as described in Section 4.2.3.3. In this task, each skeleton sequence is assigned a label.

Baselines In addition to comparing the performance of the fine-tuned model to a model trained from scratch, we also adopt the following baselines:

- **MLP** We train a 3-layer MLP as a simple baseline. The intermediate layer dimension is set to 512. The first layer of the MLP maps the

flattened input $\mathbf{x} \in \mathbb{R}^{l \times V \times d}$ into a hidden space with a dimension of 512. Each referee in the input shares the same linear encoder. The final layer takes the flattened output from the previous layers, which includes dimensions corresponding to the number of referees, and maps it into a 9-way classification space. The MLP undergoes training for 20 epochs using an AdamW optimizer with a weight decay of $5e-4$. The learning rate is set at $3e-4$, and the batch size is set at 16. The learning rate follows an inverse square root decay after a 2,000-step linear warmup.

- **ST-GCN** [3] ST-GCN utilizes graph convolutional networks to leverage the connectivity information of human joints through graphs. It is a widely used and strong baseline for skeleton-based action recognition. We train a 9-layer ST-GCN with hidden sizes (128, 128, 128, 256, 256, 256, 512, 512, 512). The model is trained for 20 epochs with a learning rate of $3e-4$. An AdamW optimizer is used with weight decay $5e-4$. The batch size is set to 16. The learning rate schedule follows an inverse square root decay with a 2,000-step linear warmup.

To better adapt the pre-trained model for the referee action recognition task through fine-tuning, we incorporate additional inputs alongside the skeleton sequences. As detailed in Section 4.2.3, this includes the absolute position of the referee, the position of the ball, and the skeletons of the two assistant referees. To integrate this additional information, we employ only the residual module. The performance of the extra token design is discussed in the ablation study. The model is fine-tuned on a dataset specifically collected for the referee action recognition task, as described in Table 4.5.

The results are presented in Table 4.9. As shown in the table, for the 9-way referee skeleton sequence classification task, a simple MLP achieves an accuracy of 85.85% on the validation set and 84.15% on the test set, which gives an assessment of the task’s complexity. The ST-GCN model, which provides a more appropriate framework for modeling spatiotemporal information, outperforms the MLP by a considerable margin. Our fine-tuned model (line 4) achieves an accuracy of 93.90% on the validation set and 93.59% on the test set, significantly outperforming the model trained from scratch with the same architecture (line 3) and the baseline methods. This underscores the benefits of the pre-training phase in transferring general knowledge and improving the final model performance.

Method	#Params	Valid Acc.	Test Acc.
MLP	3.5M	85.85	84.15
ST-GCN [3]	12.4M	88.05	89.51
S-SPT _R w/o pre-training	30.8M	90.12	91.26
S-SPT _R	30.8M	93.90	93.59

Table 4.9: Performance comparison on referee action recognition on the sequence classification task.

Ablation Study We conducted an ablation study to evaluate the effectiveness of incorporating additional inputs and their integration methods into our model. The results, presented in Table 4.10, show that utilizing all additional inputs with only the residual module as the integration method yields the best results. Conversely, applying both integration methods simultaneously performs slightly worse, and using only the extra token method is the least effective. This suggests that incorporating additional information directly into the attention mechanism does not necessarily lead to a better understanding of these inputs. A potential explanation is that the new token for additional inputs, not seen during the pre-training phase, poses challenges for training new weights in the transformer layer to interpret them.

In the second set of experiments (lines 4 to 7), we systematically removed each additional input—pelvis position, ball position, and actions of other referees—while maintaining the residual module as the only integration method. The findings indicate that each input contributes to the recognition performance. Notably, removing the actions of other referees results in the most significant performance drop, underscoring the critical role of other referees’ actions in understanding the main referee’s actions due to frequent interactions between them.

Furthermore, we experimented with removing both pelvis and ball positions from the additional inputs (the last two lines), eliminating the need for an integration method, to specifically assess the impact of other referees’ actions. The results show only a minimal drop in performance when the actions of other referees are included but a significant decline when they are excluded. This again highlights the importance of including other referees’ actions in the input to maintain robust model performance.

Additional Input			Integration Method		Valid Acc.	Test Acc.
Pelvis pos.	Ball pos.	Other referees	Residual	Extra token		
✓	✓	✓	✓		93.90	93.59
✓	✓	✓	✓	✓	93.41	92.89
✓	✓	✓		✓	91.83	92.68
✓	✓		✓		90.93	89.56
✓		✓	✓		93.66	93.01
	✓	✓	✓		92.56	91.26
		✓			92.56	92.77
					83.09	84.93

Table 4.10: Ablation study on additional inputs and their integration methods for the sequence classification task.

4.3.3.3 Results with Sequence Labeling Task

As discussed in Section 4.2.3.4, one limitation of the model trained with the sequence classification task is that it causes significant delays when implemented in real-game scenarios. To mitigate this issue, we explore fine-tuning the model with a sequence-labeling task. Under this setting, the model is trained under the supervision of a binary classification model that provides action vs. non-action predictions. We use an expanded version of the sequence classification dataset (Table 4.5), detailed in Section B.2.7, where each sequence is extended to 8 seconds to offer additional context. Given that the MLP and ST-GCN models are not designed to generate frame-level predictions, we only compare our fine-tuned model to the model trained from scratch. Further comparisons of model performance will be conducted during the evaluation of the model in real-game scenarios.

The fine-tuning details follow that used in the sequence labeling task. We predict one label for each token within the sequence. With the time patch size set to 5 and a frame rate of 25 Hz, each token corresponds to an action spanning 0.2 seconds in the original sequence. The fine-tuning loss is computed on the tokens from the 2-second mark to the end of each sequence.

We adopt best practices from the sequence classification task concerning the choice and integration of additional inputs. This includes incorporating all available inputs, i.e., ball position, absolute referee location, and actions of other referees, and using a residual module to integrate these into the latent space.

Method	#Params	Pre-masking	Label		Valid Acc.	Test Acc.
			Hard	Soft		
S-SPT _R w/o pre-training	30.8M				75.71	76.33
S-SPT _R	30.8M	✓		✓	92.17	91.01
		✓	✓		92.29	91.51
				✓	92.31	91.13
			✓		92.75	91.88

Table 4.11: Performance comparison on referee action recognition on the sequence labeling task. Pre-masking means pre-sequence random-length masking as introduced in Section 4.2.3.4.

For settings specific to sequence labeling, we experiment with both hard and soft labels, and with and without pre-sequence random length masking. The results are shown in Table 4.11. A model with the same architecture trained from scratch is set as a baseline for this experiment, as shown in the first line in the table. In the sequence labeling setting, we face the challenge of lacking frame-level ground truth labels. To overcome this, we utilize hard labels as proxies for ground truth. As shown in the table, the fine-tuned models consistently outperform the one trained without pre-training, demonstrating the crucial role of the pre-training phase in enhancing the model’s effectiveness. The best-performing model with the sequence labeling task achieves an accuracy of 92.75% on the validation set and 91.88% on the test set, highlighting the model’s proficiency in giving frame-level prediction based on sequence inputs. However, regarding the impact of the proposed settings, although the model trained with hard labels without pre-sequence random-length masking achieves the best results, the performance differences among the various settings are marginal. Considering that the ground truth labels used for evaluation in this experiment are generated by the binary classification model that is not perfectly reliable, we conclude that the different combinations of settings yield comparable performance. Therefore, a comprehensive evaluation in a real-game scenario is necessary to determine the optimal setting.

4.3.4 Evaluation on Real Games

Similar to our frame-level model, we evaluate our sequence-level model in real-world scenarios, using the same game as in the frame-level evaluation—the game between Miami CF and CF Montreal played on February 26, 2023, referred to as MIA-MTL. Additionally, we evaluate

our models on another game, played between Seattle Sounders FC and Los Angeles FC on March 18, 2023, referred to as SEA-LAF. Evaluating models in real-game scenarios is considered the gold standard for assessing model performance, while comparing model performance using our collected dataset’s evaluation set can be biased because the dataset itself may not accurately reflect the distribution of actions in real games nor cover all potential real-world actions. Thus, a higher accuracy on the test set does not necessarily guarantee better real-world performance. Evaluating in real games is crucial for accurately assessing model capabilities, identifying model shortcomings, and proposing improvements. It also enables a fair comparison between our models that have been fine-tuned with the sequence classification and sequence labeling tasks, ensuring a more rigorous selection of the optimal model.

4.3.4.1 Model Fine-tuned with Sequence Classification Task

We evaluate the best-performing model fine-tuned with sequence classification task on real games, which corresponds to the model shown in Table 4.9. To apply the model to the real-game scenario, post-processing techniques introduced in Section 4.2.4 are needed. We use a fixed context length of 4 seconds to provide context information since the model is fine-tuned on a dataset with all sequences being 4 seconds long. That is, the prediction for each frame in the game is based on the previous 4-second referee skeleton sequence. The duration threshold is set at 0.5s, and the confidence threshold is set at 0.8 to effectively filter out potential false positives. The evaluation depends on human decision for an action being correctly recognized or not. The results are shown in Table 4.12.

	# In Game	Precision	Recall	F1-Score
DIRECT_FREE_KICK	24	0.2396	0.9583	0.3833
INDIRECT_FREE_KICK	6	0.2500	1.0000	0.4000
THROW_IN_HOME	18	0.8095	0.9444	0.8718
THROW_IN_AWAY	18	0.8095	0.9444	0.8718
GOAL_KICK	15	0.6364	0.9333	0.7568
CORNER_KICK	8	0.6667	1.0000	0.8000
GOAL	3	0.2727	1.0000	0.4286
OTHER_DECISION	110	0.5037	0.6296	0.5597
Overall	202	0.4985	0.7630	0.6030

(a) Evaluation results on game MIA-MTL.

	# In Game	Precision	Recall	F1-Score
DIRECT_FREE_KICK	26	0.5526	0.8077	0.6563
INDIRECT_FREE_KICK	3	0.2143	1.0000	0.3529
THROW_IN_HOME	20	0.8000	1.0000	0.8889
THROW_IN_AWAY	15	0.8235	0.9333	0.8750
GOAL_KICK	20	0.6296	0.8500	0.7234
CORNER_KICK	13	0.7333	0.8462	0.7857
GOAL	0	0.0000	-	-
OTHER_DECISION	86	0.5431	0.7326	0.6238
Overall	183	0.6383	0.8152	0.7160

(b) Evaluation results on game SEA-LAF.

Table 4.12: Performance of the model trained with sequence classification task on real games.

As shown in the tables, for game MIA-MTL in Table 4.12a, the recall for most classes, except for the class `OTHER_DECISION`, is high, indicating that the model generally does not miss or misclassify true referee signals. Specifically, for `GOAL`, `CORNER_KICK`, and `INDIRECT_FREE_KICK`, the recall reaches 1.0, meaning that every referee signal in these categories is correctly predicted. However, the precision for some classes is unsatisfactory. For instance, the precision for `DIRECT_FREE_KICK`, `INDIRECT_FREE_KICK`, and `GOAL` does not exceed 0.3, indicating a large number of false positives. Although the precision is higher for other classes, the model remains overly sensitive to noise, resulting in false positives. The recall for `OTHER_DECISION` is lower compared to other classes, and the precision is also unsatisfactory, likely due to insufficient

training data. For game SEA-LAF, we observe similar patterns. The recall for `INDIRECT_FREE_KICK`, `THROW_IN`, `GOAL_KICK`, and `CORNER_KICK` is high, indicating these actions can be recognized well from the game. However, the precisions are not satisfactory, especially for `DIRECT_FREE_KICK`, `INDIRECT_FREE_KICK` and `GOAL_KICK`, where the precision is relatively lower, suggesting many false positives. There are no goals in this game, but the model still outputs false positive `GOAL` predictions, resulting in a precision of zero for this action. Overall, the model achieves an F1 score of 0.6030 for game MIA-MTL and 0.7160 for game SEA-LAF. With the overall precisions not exceeding 0.7, the indicated prevalence of false positives limits the model's practical applicability.

4.3.4.2 Model Fine-tuned with Sequence Labeling Task

For the model fine-tuned with the sequence labeling task, we evaluate models trained with different combinations of supervision label types and the use of pre-sequence random-length masking, as shown in Table 4.11. We use a context length of 6 seconds for this evaluation, as the model was fine-tuned with a history length varying from 0 to 8 seconds. A 6-second context provides both relatively long context information and is commonly encountered during training. The duration threshold is set to 0.5s, and the confidence threshold is set to 0.8, consistent with the evaluation of the model trained with the sequence classification task in the previous section. The results are presented in Table 4.13.

First, compared to the evaluation results of the model trained with the sequence classification task, the model trained with the sequence labeling task achieves a best F1 score of 0.7121 and 0.7863, respectively, on the two games. This significantly outperforms the model trained with the sequence classification task, whose F1 score is 0.6030 and 0.7160, as shown in Table 4.12. While maintaining high recall for most classes, the model trained with the sequence labeling task achieves much higher precision, indicating significantly fewer false positives.

For `OTHER_DECISION`, the model trained with the sequence labeling task achieves a higher F1 score with significantly higher precision but some sacrifice in recall. This improvement is likely due to the sequence labeling task providing supervision signals on each frame rather than each sequence, thereby significantly enhancing the efficacy of leveraging labeled data. However, the supervision provided by a trained model could introduce errors. This might explain why the model tends to classify some actions with limited motion range as `NO_DECISION`, leading to a decrease in recall.

	# In Game	Hard w/o pre-masking			Hard w/ pre-masking		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
DIRECT_FREE_KICK	24	0.5946	0.9167	0.7213	0.5263	0.8333	0.6452
INDIRECT_FREE_KICK	6	0.3158	1.0000	0.4800	0.3750	1.0000	0.5455
THROW_IN_HOME	18	0.9444	0.9444	0.9444	0.7500	1.0000	0.8571
THROW_IN_AWAY	18	0.8182	1.0000	0.9000	0.8182	1.0000	0.9000
GOAL_KICK	15	1.0000	0.9333	0.9655	0.8333	1.0000	0.9000
CORNER_KICK	8	0.6667	1.0000	0.8000	0.5000	1.0000	0.6667
GOAL	3	0.7500	1.0000	0.8571	0.3750	1.0000	0.5455
OTHER_DECISION	110	0.7600	0.3455	0.4750	0.6173	0.4425	0.5155
Overall	202	0.7879	0.6161	0.6914	0.6796	0.6667	0.6731
	# In Game	Soft w/o pre-masking			Soft w/ pre-masking		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
DIRECT_FREE_KICK	24	0.5000	0.6667	0.5714	0.6129	0.7917	0.6909
INDIRECT_FREE_KICK	6	0.4615	1.0000	0.6316	0.3750	1.0000	0.5455
THROW_IN_HOME	18	0.8095	0.9444	0.8718	0.8947	0.9444	0.9189
THROW_IN_AWAY	18	0.6429	1.0000	0.7826	0.6667	1.0000	0.8000
GOAL_KICK	15	0.9375	1.0000	0.9677	0.7778	0.9333	0.8485
CORNER_KICK	8	0.5714	1.0000	0.7272	0.6667	1.0000	0.8000
GOAL	3	0.5000	1.0000	0.6667	0.5000	1.0000	0.6667
OTHER_DECISION	110	0.7385	0.4364	0.5486	0.7879	0.4771	0.5943
Overall	202	0.7598	0.6445	0.6974	0.7581	0.6714	0.7121

(a) Evaluation results on game MIA-MTL.

	# In Game	Hard w/o pre-masking			Hard w/ pre-masking		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
DIRECT_FREE_KICK	26	0.7619	0.6154	0.6809	0.6667	0.6154	0.6400
INDIRECT_FREE_KICK	3	0.3750	1.0000	0.5455	0.3333	1.0000	0.5000
THROW_IN_HOME	20	0.9091	1.0000	0.9524	0.8333	1.0000	0.9091
THROW_IN_AWAY	15	1.0000	0.9333	0.9655	1.0000	0.9333	0.9655
GOAL_KICK	20	0.9048	0.9500	0.9268	0.9048	0.9500	0.9268
CORNER_KICK	13	0.8000	0.9231	0.8571	0.6500	1.0000	0.7879
GOAL	0	0.0000	-	-	0.0000	-	-
OTHER_DECISION	86	0.7164	0.5517	0.6234	0.6265	0.5977	0.6118
Overall	183	0.8571	0.7097	0.7765	0.7874	0.7366	0.7611
	# In Game	Soft w/o pre-masking			Soft w/ pre-masking		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
DIRECT_FREE_KICK	26	0.8333	0.5769	0.6818	0.6957	0.6154	0.6531
INDIRECT_FREE_KICK	3	0.4286	1.0000	0.6000	0.4286	1.0000	0.6000
THROW_IN_HOME	20	0.9524	1.0000	0.9756	0.9091	1.0000	0.9524
THROW_IN_AWAY	15	0.9231	0.8000	0.8571	0.9333	0.9333	0.9333
GOAL_KICK	20	0.9000	0.9000	0.9000	0.9500	0.9500	0.9500
CORNER_KICK	13	0.8125	1.0000	0.8966	0.8000	0.9231	0.8571
GOAL	0	0.0000	-	-	0.0000	-	-
OTHER_DECISION	86	0.6548	0.6322	0.6433	0.6667	0.6207	0.6429
Overall	183	0.8293	0.7312	0.7771	0.8364	0.7419	0.7863

(b) Evaluation results on game SEA-LAF.

Table 4.13: Performance of the model trained with sequence labeling task on real games. 'Hard w/o pre-masking' means the model is fine-tuned with hard labels without pre-sequence random-length masking; 'Hard w pre-masking' means the model is fine-tuned with hard labels and pre-sequence random-length masking; 'Soft w/o pre-masking' means the model is fine-tuned with soft labels without pre-sequence random-length masking; 'Soft w/ pre-masking' means the model is fine-tuned with soft labels and pre-sequence random-length masking;

When comparing different fine-tuning settings for the model trained with the sequence labeling task, the best-performing model in the real-game evaluation is the model trained with soft labels and pre-sequence random-length masking, consistent across both games. This result doesn't align perfectly with the evaluation on the validation and test set in Table 4.11. The difference might be due to the collected evaluation set being biased and not accurately reflecting the real-world data distribution. We consider the real-game evaluation to be more reliable since it is closer to the practical application of the model. Therefore, the model trained with soft labels and pre-sequence random-length masking is considered the best model for referee action recognition in this study.

We can observe from the table that models trained with soft labels consistently outperform those trained with hard labels, particularly due to improved performance on the `OTHER_DECISION` class. The likely reason is that with soft labels, more frames are assigned a probability for the `OTHER_DECISION` label, which is originally zero in the hard label setting. Consequently, the model trained with soft labels is exposed to more frames with a probability of being `OTHER_DECISION`, resulting in more robust performance. When trained with soft labels, as shown in the lower part of each table, pre-sequence random-length masking further enhances model performance. This technique varies the history length for a particular token during training across different epochs, preventing potential overfitting and leading to better results.

In view of practicality, although the performance for the `OTHER_DECISION` class is not entirely satisfactory, referee actions in this class do not have as significant an impact on the game as other well-defined actions. Applications of the model might rely more on well-defined actions rather than those categorized under `OTHER_DECISION`. In such cases, our model demonstrates greater capability. For example, a potential application of the referee action recognition system is to assist in recognizing key events on the pitch, such as throw-ins, goal kicks, and corner kicks. Since our model consistently achieves an F1 score above 0.9 for the `THROW_IN` and `GOAL_KICK`, and above 0.8 for `CORNER_KICK` signals, the predictions for these labels are relatively reliable and can effectively contribute to real-time event recognition during a game.

4.3.4.3 Inference Time

To evaluate if our model is capable of real-time recognition, we assessed the inference time of our S-SPT_R on real games. An Nvidia RTX 3060 GPU is

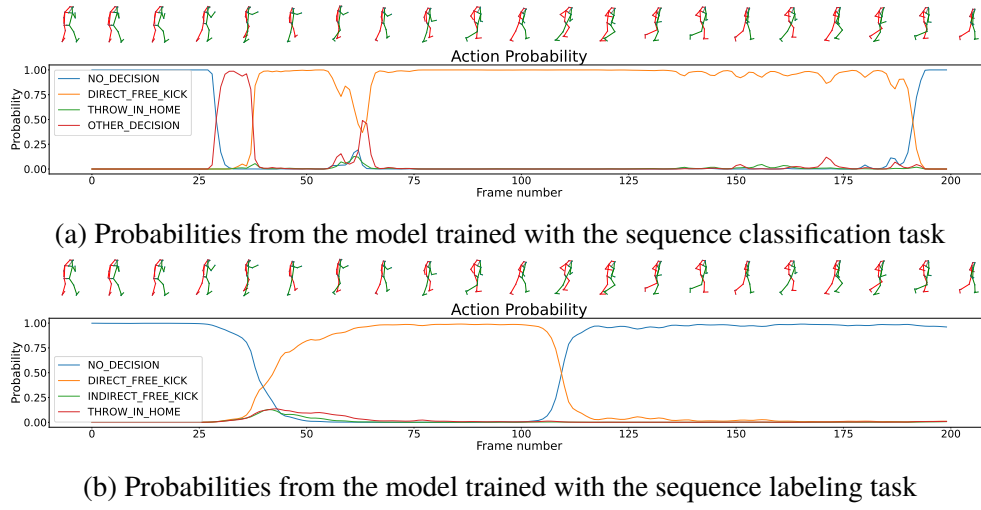
used for the evaluation. The evaluation is performed frame by frame, with the context sequence length of each frame being 6 seconds long. We run the inference for 1,000 frames and report the average inference time per frame. As shown in Table 4.14, the inference time per frame is no more than 6 ms, indicating that our model can provide real-time referee action recognition at a frequency of at least 150 Hz. While increasing the batch size generally improves inference time, a batch size of 32 causes a longer inference time. This may be due to the bottleneck shifting away from GPU computation. Nevertheless, the inference time remains sufficiently low to support real-time referee action recognition.

	Batch Size	GPU Memory Usage	Inference Time (ms/frame)
S-SPT _R	1	752M	5.4
	4	1154M	2.3
	16	2640M	2.1
	32	4664M	3.0

Table 4.14: Inference time of S-SPT_R on real games. Evaluations are performed on an Nvidia RTX 3060 GPU. Context sequences are 6 seconds long.

4.3.5 Case Study

Similar to the frame-level model, we observe predicted probabilities over frames to assess prediction stability. Moreover, we aim to observe the differences in delay caused by different models. We analyze 3 cases: a `DIRECT_FREE_KICK`, an `INDIRECT_FREE_KICK`, and a `GOAL`. The outputs of two models are plotted: one trained with the sequence classification task, as introduced in Table 4.9, and the other trained with the sequence labeling task, utilizing soft labels and pre-sequence random-length masking, as shown in the second line of Table 4.11. Following the settings used for frame-level models, we set the frame rate at 25 Hz and omit classes with probabilities consistently below 0.05 from the plot during our observations. The left side of the body is shown in red, and the right side is shown in green.

Figure 4.4: Output probabilities for a `DIRECT_FREE_KICK`

`DIRECT_FREE_KICK` We plot the output probabilities for a `DIRECT_FREE_KICK` in Figure 4.4. The upper subfigure (4.4a) shows the output probabilities from the model trained with the sequence classification task, while the lower subfigure (4.4b) presents the predictions from the model trained with the sequence labeling task. As illustrated in the figures, the predictions from the model trained with the sequence classification task are more chaotic, with a false positive `OTHER_DECISION` around frame 25. In contrast, the prediction probabilities from the model trained with the sequence labeling task for irrelevant labels are never high enough to dominate. This suggests that the model trained with the sequence labeling task offers better prediction accuracy and stability.

Moreover, there is a significant delay in the prediction in the upper subfigure. The actual action ends around frame 100, but the prediction for `DIRECT_FREE_KICK` remains positive for another 80 to 100 frames for the model trained with the sequence classification task. This delay is due to the prediction at each frame being based on the previous 4-second sequence, which is equivalent to 100 frames. In contrast, the model trained with the sequence labeling task stops predicting `DIRECT_FREE_KICK` labels just around one second after the action terminates, indicating that it effectively solves the problem of delay.

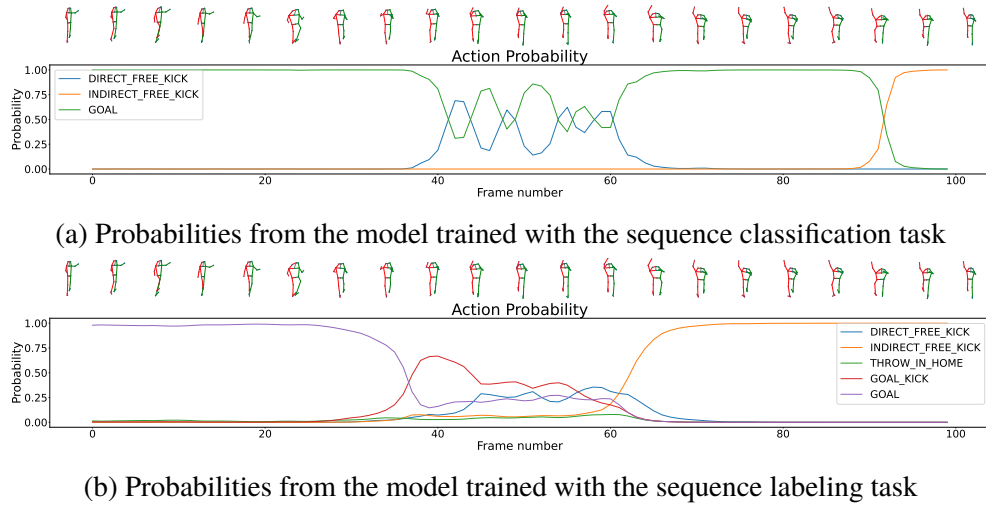


Figure 4.5: Output probabilities for an `INDIRECT_FREE_KICK`

INDIRECT_FREE_KICK Figure 4.5 shows the action probabilities predicted by the two models for an `INDIRECT_FREE_KICK`. In this case, the ball went into the net, and the referee initially signaled a `GOAL` (arm pointing to the center mark). However, the referee later realized that the assistant referee signaled for offside, so the referee disallowed the goal and signaled an `INDIRECT_FREE_KICK` (raising the hand vertically).

For the model trained with the sequence classification task, the prediction of each frame is based on the previous 4-second sequence. This sequence might include two different actions in this case, `GOAL` and `INDIRECT_FREE_KICK`, leading to confusion and suboptimal predictions. From the plot, we observe that the model experiences confusion around frame 50 and eventually predicts `GOAL` from frame 60 to frame 90, even though the referee is already signaling an `INDIRECT_FREE_KICK`. Due to this confusion and delay, the model doesn't identify the `INDIRECT_FREE_KICK` until around frame 100, almost 2 seconds after the action begins.

In contrast, the model trained with the sequence labeling task predicts `INDIRECT_FREE_KICK` around frame 70, almost immediately after the referee gives the signal. Although there is some confusion between frames 40 and 60, it does not affect the later prediction, as the prediction in this model is only responsible for the current frame rather than the previous 4-second sequence. This suggests that training with the sequence labeling task reduces interference between different actions and results in better real-game performance.

GOAL In Figure 4.6 and Figure 4.7, we show the variation of classification

probabilities predicted by the two models for a `GOAL`. In this case, both models perform well in predicting the `GOAL`. Figure 4.6 shows the beginning stage of the `GOAL` signal. We can see from the plot that the model trained with the sequence labeling task (lower subfigure 4.6b) responds earlier than the model trained with the sequence classification task (upper subfigure 4.6a). Figure 4.7 shows the probabilities during the entire `GOAL` signal. This demonstrates that the model trained with the sequence labeling task also responds more quickly to the termination of the action. This indicates that the model trained with the sequence labeling task has a shorter latency in responding to referee signals, with a quicker response not only at the end of the signal but also at the beginning.

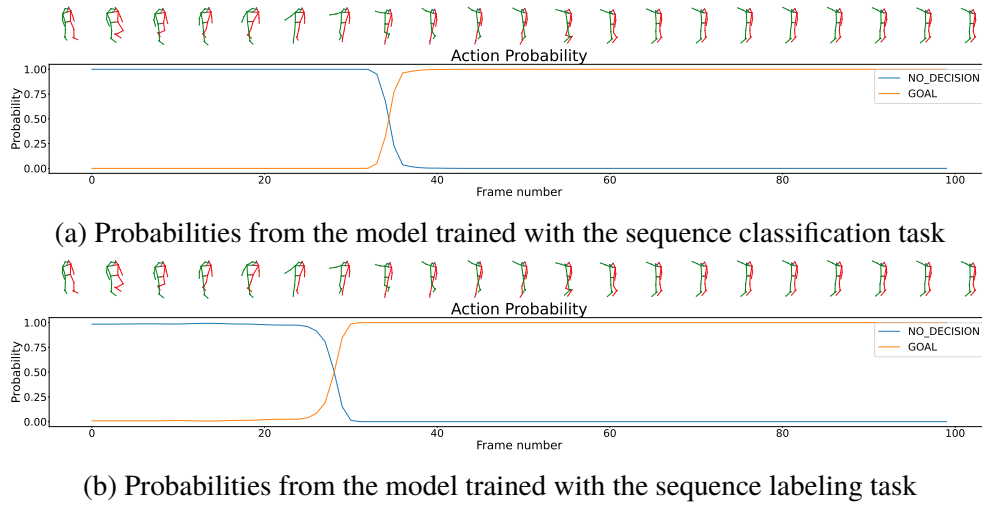


Figure 4.6: Output probabilities for a `GOAL` in the beginning

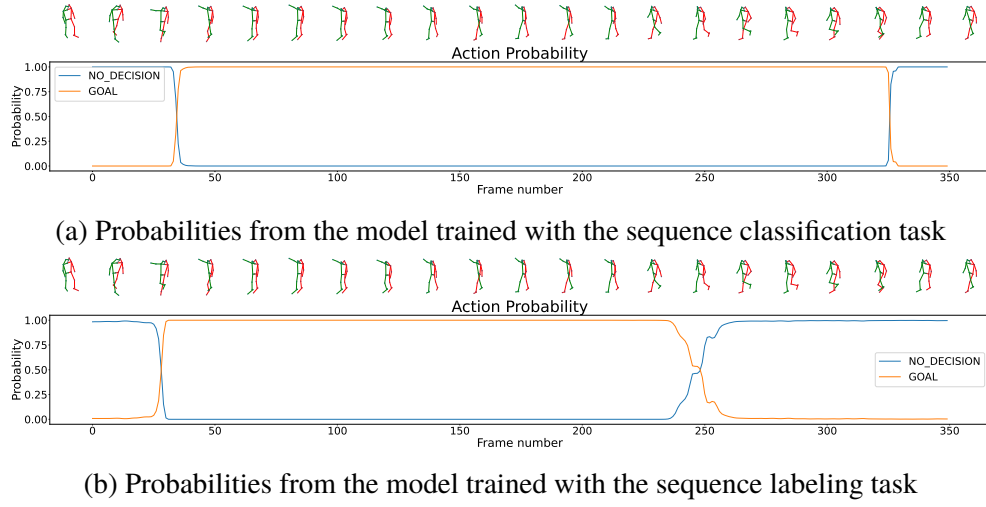


Figure 4.7: Output probabilities for the entire process of the GOAL signal

To summarize, our case studies confirm that the model trained with the sequence labeling task outperforms the model trained with the sequence classification task in several aspects. First, it solves the problem of delay after an action ends. Moreover, It demonstrates quicker responses to referee actions and is less influenced by the confusion between different actions within the same sequence, resulting in more accurate and timely predictions in real-game scenarios.

4.3.6 Summary

Due to the more complex nature of the main referee's actions, the incorporation of sequence-level context is essential. Our research confirms that pre-training is a fundamental step for this purpose. Specifically, our proposed multi-task pre-training objective proves effective in learning general patterns for referee movements. To further enhance the performance of action recognition, we integrate additional information, such as ball position and the actions of other referees, into the input. Experimental results indicate the efficacy of this approach. More importantly, We propose a novel fine-tuning strategy using a sequence-labeling task instead of the conventional sequence-classification task. Evaluations conducted on real-game scenarios, along with detailed case studies, demonstrate that this strategy effectively solves delays, significantly enhances the utilization of labeled data, and consequently results in satisfactory performance in real-game scenarios.

Chapter 5

Conclusions and Future work

5.1 Conclusions

In this work, we propose a pre-training and fine-tuning pipeline to develop skeleton-based action recognition models for football referees. We employ different strategies tailored for different types of referees, training a frame-level model for assistant referees and a sequence-level model for main referees. Various strategies are implemented to collect data that support robust pre-training and fine-tuning.

For the frame-level model, we design a simple yet effective pre-training method with Transformer-based autoencoders. The model, fine-tuned on an elaborately collected dataset, demonstrates excellent performance for assistant referee action recognition on both the evaluation set and in real-game scenarios, thereby proving its practical applicability in live sports events.

For the sequence-level model, we propose a novel multi-task pre-training approach that integrates two advanced pre-training techniques: Masked Motion Predictor (MAMP) [33] and data2vec [36]. Experimental results verify the effectiveness of this approach. Furthermore, in the fine-tuning stage, we fine-tune the model with a novel sequence-labeling task, which addresses the challenge of delay inherent in sequence-level models and utilizes labeled data more efficiently. Additionally, we implement strategies to incorporate additional inputs, thereby improving the model's comprehension of referee actions. Experimental results on the evaluation set and real game data show that these methods surpass baseline systems such as ST-GCN and produce satisfactory results.

Our results demonstrate the effectiveness of the pre-training stage in learning general knowledge of referee actions, significantly enhancing the

accuracy and performance of the models during the fine-tuning stage for the specific task. This approach ensures that both frame-level and sequence-level actions are accurately recognized, providing a robust solution for skeleton-based action recognition in football referees. It is worth noting that significant efforts were made to collect the dataset, and the experiments also suggest that the quality of the dataset is crucial for the final performance. Collecting more high-quality data remains the most efficient way to improve model performance.

5.2 Limitations

Despite the promising results, there are several limitations to this work:

- The target classes are not highly fine-grained, which limits the model's ability to predict specific classes. For example, referee signals for advantage are not a separate class but are included in the `OTHER_DECISION` class in this work, preventing the model from accurately predicting referees showing advantage signals.
- Although most assistant referee actions are static, some are dynamic. The frame-level model design restricts the ability to identify dynamic actions. For example, an assistant referee waving the flag to signal a foul might be incorrectly interpreted by our frame-level model as `RAISE_FLAG_VERTICALLY`.
- The performance of the fine-tuned model heavily depends on the availability of high-quality labeled data. Insufficient data can result in inadequate training and suboptimal performance. For instance, the amount of `DIRECT_FREE_KICK` and `OTHER_DECISION` data may not be sufficient for the model to train effectively.
- It is sometimes impossible to determine the meaning of referee signals solely by relying on skeleton data. Humans actually rely on all available information on the pitch, including the actions of all players and the position of the ball, to understand what is happening. Thus, depending solely on referee skeletons does not inherently guarantee the recognition of all referee actions.

5.3 Future work

Future research could address the limitations and extend the findings of this work in several ways:

- Implementing a more systematic strategy for collecting classification datasets, which includes human annotations by experts such as professional referees, to create a larger and higher-quality dataset. A more representative dataset is expected to lead to significant improvements.
- Leveraging few-shot learning methods to recognize rare classes, thereby improving recognition accuracy for classes that naturally lack sufficient training data.
- Developing strategies to combine assistant referee action predictions from the frame-level model and referee action predictions from the sequence-level model, which is crucial for applications such as event detection.
- Introducing finer-grained class divisions to perform more detailed classifications and provide more specific predictions about referee actions. For example, actions such as `ADVANTAGE`, `NO_FOUL`, and `SANCTION` should be classified into individual classes, provided there is sufficient data.
- Applying the sequence-level approaches designed for main referee actions to assistant referee action recognition, which could potentially yield better performance.
- Including additional contextual information to improve the understanding of events. For instance, the actions of players could be beneficial for identifying free kicks, as players typically slow down and prepare for the kick when it is called.
- Ensuring the models are free from bias, performing equally well for different games and different referees.

5.4 Reflections

With accurate recognition of referee actions from this work, there can be a significant reduction in the need for extensive human resources to manually

analyze and label referee actions in football games. This optimization can lead to cost savings associated with human trainers and manual annotation. It also results in a more efficient use of resources, contributing to operational efficiency. Although specific environmental benefits, such as reduced carbon footprints, are not directly addressed in the thesis, the reduction in resource use indirectly supports environmental sustainability.

The automated referee recognition system could also promote community engagement by enhancing transparent officiating. Referee actions are not as easy to understand as player actions by the community, including fans, players, and stakeholders. By providing clear and accurate interpretations of referee signals from the model, we could improve the real-time understanding of these referee decisions, thereby reducing controversies and making football more united for everyone.

One concern is ensuring that the models do not exhibit bias and that they perform equally well across different scenarios and for different referees. Ensuring fairness in the algorithm's decisions is essential to maintain the integrity of the sport. This aspect is not addressed in this thesis and should be a critical part of future work. This includes regular updating of algorithms to maintain fairness and accuracy, thereby maintaining the ethical standards of the technology and ensuring its widespread acceptance and credibility within the football community.

References

- [1] V. Veeriah, N. Zhuang, and G.-J. Qi, “Differential Recurrent Neural Networks for Action Recognition,” pp. 4041–4049, 2015. [Page 1.]
- [2] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. van Gool, “Temporal segment networks: Towards good practices for deep action recognition,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9912 LNCS, pp. 20–36, 2016. doi: 10.1007/978-3-319-46484-8_2/FIGURES/3. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-46484-8_2 [Page 1.]
- [3] S. Yan, Y. Xiong, and D. Lin, “Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 4 2018. doi: 10.1609/AAAI.V32I1.12328. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/12328> [Pages 1, 9, 10, 16, 72, and 73.]
- [4] L. Shi, Y. Zhang, J. Cheng, and H. Lu, “Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition,” pp. 12 026–12 035, 2019. [Online]. Available: <https://github.com/lshiwjx/2s-AGCN> [Pages 1, 9, and 16.]
- [5] P. Zhang, C. Lan, W. Zeng, J. Xing, J. Xue, and N. Zheng, “Semantics-Guided Neural Networks for Efficient Skeleton-Based Human Action Recognition,” pp. 1112–1121, 2020. [Pages 1 and 16.]
- [6] R. Yue, Z. Tian, and S. Du, “Action recognition based on RGB and skeleton data sets: A survey,” *Neurocomputing*, vol. 512, pp. 287–306, 11 2022. doi: 10.1016/J.NEUCOM.2022.09.071 [Page 1.]

- [7] L. Wang and P. Koniusz, “3Mformer: Multi-order Multi-mode Transformer for Skeletal Action Recognition,” pp. 5620–5631, 3 2023. doi: 10.1109/cvpr52729.2023.00544. [Online]. Available: <https://arxiv.org/abs/2303.14474v1> [Page 1.]
- [8] J. Lee, M. Lee, S. Cho, S. Woo, S. Jang, and S. Lee, “Leveraging Spatio-Temporal Dependency for Skeleton-Based Action Recognition,” pp. 10 255–10 264, 2023. [Online]. Available: <https://github.com/Jho-Yonsei/STC-Net>. [Pages 1 and 17.]
- [9] J. Lee, M. Lee, D. Lee, and S. Lee, “Hierarchically Decomposed Graph Convolutional Networks for Skeleton-Based Action Recognition,” pp. 10 444–10 453, 2022. [Online]. Available: <http://arxiv.org/abs/2208.10741> [Pages 1, 9, and 17.]
- [10] Y. Zhu, H. Han, Z. Yu, and G. Liu, “Modeling the Relative Visual Tempo for Self-supervised Skeleton-based Action Recognition,” pp. 13 913–13 922, 2023. [Online]. Available: <https://github.com/Zhuysheng/RVTCLR>. [Pages 1 and 19.]
- [11] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, “NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2684–2701, 5 2019. doi: 10.1109/TPAMI.2019.2916873. [Online]. Available: <http://arxiv.org/abs/1905.04757><http://dx.doi.org/10.1109/TPAMI.2019.2916873> [Pages 1, 4, 9, and 48.]
- [12] L. Lin, J. Zhang, and J. Liu, “Actionlet-Dependent Contrastive Learning for Unsupervised Skeleton-Based Action Recognition,” pp. 2363–2372, 2023. [Pages 1 and 19.]
- [13] J. Chung, C.-h. Wu, H.-r. Yang, Y.-W. Tai, and C.-K. Tang, “HAA500: Human-Centric Atomic Action Dataset With Curated Videos,” pp. 13 465–13 474, 2021. [Online]. Available: <https://www.cse.ust.hk/haa>. [Page 1.]
- [14] F. Wu, Q. Wang, J. Bian, N. Ding, F. Lu, J. Cheng, D. Dou, and H. Xiong, “A Survey on Video Action Recognition in Sports: Datasets, Methods and Applications,” *IEEE Transactions on Multimedia*, 6 2022. doi: 10.1109/TMM.2022.3232034. [Online]. Available: <https://arxiv.org/abs/2206.01038v1> [Page 1.]

- [15] Q. Zeng, J. Liu, D. Yang, Y. He, X. Sun, R. Li, and F. Wang, "Machine Learning Based Automatic Sport Event Detection and Counting," *Proceedings of 2021 7th IEEE International Conference on Network Intelligence and Digital Content, IC-NIDC 2021*, pp. 16–20, 2021. doi: 10.1109/IC-NIDC54101.2021.9660509 [Page 1.]
- [16] R. Theagarajan and B. Bhanu, "An Automated System for Generating Tactical Performance Statistics for Individual Soccer Players from Videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 2, pp. 632–646, 2 2021. doi: 10.1109/TCSVT.2020.2982580 [Pages 1 and 24.]
- [17] Z. Cai, H. Neher, K. Vats, D. A. Clausi, and J. Zelek, "Temporal Hockey Action Recognition via Pose and Optical Flows," pp. 0–0, 2019. [Page 1.]
- [18] R. Li and B. Bhanu, "Fine-Grained Visual Dribbling Style Analysis for Soccer Videos With Augmented Dribble Energy Image," pp. 0–0, 2019. [Page 1.]
- [19] S. Barbon Junior, A. Pinto, J. V. Barroso, F. G. Caetano, F. A. Moura, S. A. Cunha, and R. d. S. Torres, "Sport action mining: Dribbling recognition in soccer," *Multimedia Tools and Applications*, vol. 81, no. 3, pp. 4341–4364, 1 2022. doi: 10.1007/S11042-021-11784-1/METRICS. [Online]. Available: <https://link.springer.com/article/10.1007/s11042-021-11784-1> [Page 1.]
- [20] R. Li and B. Bhanu, "Energy-Motion Features Aggregation Network for Players' Fine-grained Action Analysis in Soccer Videos," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023. doi: 10.1109/TCSVT.2023.3288565 [Page 1.]
- [21] T. Tsunoda, Y. Komori, M. Matsugu, and T. Harada, "Football Action Recognition Using Hierarchical LSTM," pp. 99–107, 2017. [Pages 1 and 24.]
- [22] M. Fani, K. Vats, C. Dulhanty, D. A. Clausi, and J. Zelek, "Pose-Projected Action Recognition Hourglass Network (PARHN) in Soccer," *2019 16th Conference on Computer and Robot Vision (CRV)*, pp. 201–208, 5 2019. doi: 10.1109/CRV.2019.00035 [Page 1.]

- [23] A. G. Larsen and G. Papi, “Prediction of football actions and identification of optimal sensor placements using a semi-supervised learning approach,” 2023. [Page 1.]
- [24] B. Gerats, “Individual Action and Group Activity Recognition in Soccer Videos,” 2020. [Page 1.]
- [25] C. Song and C. Rasmussen, “Who Did It? Identifying Foul Subjects and Objects in Broadcast Soccer Videos,” 2022. doi: 10.5220/0010770600003124. [Online]. Available: <https://orcid.org/0000-0001-6775-5553> [Page 1.]
- [26] S. Giancola, M. Amine, T. Dghaily, and B. Ghanem, “SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos,” pp. 1711–1721, 2018. [Online]. Available: <https://silviogiancola.github.io/SoccerNet>. [Pages 1 and 24.]
- [27] A. Delì, A. Cioppa, S. Giancola, M. J. Seikavandi, J. V. Dueholm, K. Nasrollahi, B. Ghanem, K. Thomas, B. Moeslund, and M. V. Droogenbroeck, “SoccerNet-v2: A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos,” pp. 4508–4519, 2021. [Online]. Available: <https://soccer-net.org/>. [Pages 1 and 24.]
- [28] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang, “Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 140–149, 3 2020. doi: 10.1109/CVPR42600.2020.00022. [Online]. Available: <https://arxiv.org/abs/2003.14111v2> [Pages 1, 9, and 16.]
- [29] Y. B. Cheng, X. Chen, D. Zhang, and L. Lin, “Motion-transformer: Self-supervised pre-training for skeleton-based action recognition,” *Proceedings of the 2nd ACM International Conference on Multimedia in Asia, MMAsia 2020*, 3 2021. doi: 10.1145/3444685.3446289. [Online]. Available: <https://dl-acm-org.focus.lib.kth.se/doi/10.1145/3444685.3446289> [Pages 1 and 9.]
- [30] C. Plizzari, M. Cannici, and M. Matteucci, “Spatial Temporal Transformer Network for Skeleton-Based Action Recognition,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol.

- 12663 LNCS, pp. 694–701, 2021. doi: 10.1007/978-3-030-68796-0_50/TABLES/2. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-68796-0_50 [Pages 1, 10, and 18.]
- [31] H. Qiu, B. Hou, B. Ren, and X. Zhang, “Spatio-Temporal Tuples Transformer for Skeleton-Based Action Recognition,” 1 2022. [Online]. Available: <https://arxiv.org/abs/2201.02849v1> [Pages 1, 10, 18, 30, and 53.]
- [32] H. Duan, M. Xu, B. Shuai, D. Modolo, Z. Tu, J. Tighe, and A. Bergamo, “SkeleTR: Towards Skeleton-based Action Recognition in the Wild,” pp. 13 634–13 644, 2023. [Pages 1, 10, 18, and 30.]
- [33] Y. Mao, J. Deng, W. Zhou, Y. Fang, W. Ouyang, and H. Li, “Masked Motion Predictors are Strong 3D Action Representation Learners,” pp. 10 181–10 191, 2023. [Pages 1, 6, 10, 13, 19, 30, 31, 48, 50, 53, 54, 55, 56, and 87.]
- [34] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, “NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis,” pp. 1010–1019, 2016. [Pages 4, 9, 17, and 48.]
- [35] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, “The Kinetics Human Action Video Dataset,” 5 2017. [Online]. Available: <https://arxiv.org/abs/1705.06950v1> [Pages 4, 9, and 48.]
- [36] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, “data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language,” pp. 1298–1312, 6 2022. [Online]. Available: <https://proceedings.mlr.press/v162/baevski22a.html> [Pages 6, 10, 15, 48, 55, 56, and 87.]
- [37] I. F. A. B. (IFAB), “Laws of the Game,” 2024. [Online]. Available: <https://www.theifab.com/> [Pages xi, 7, 21, 22, 23, 27, 37, and 47.]
- [38] J. Wang, X. Nie, Y. Xia, Y. Wu, and S.-C. Zhu, “Cross-view Action Modeling, Learning and Recognition,” pp. 2649–2656, 2014. [Page 9.]
- [39] C. Liu, Y. Hu, Y. Li, S. Song, and J. Liu, “PKU-MMD: A Large Scale Benchmark for Continuous Multi-Modal Human Action

- Understanding,” 3 2017. [Online]. Available: <https://arxiv.org/abs/1703.07475v2> [Pages 9 and 48.]
- [40] Y. Du, W. Wang, and L. Wang, “Hierarchical Recurrent Neural Network for Skeleton Based Action Recognition,” pp. 1110–1118, 2015. [Pages 10 and 17.]
- [41] J. Liu, A. Shahroudy, D. Xu, and G. Wang, “Spatio-temporal LSTM with trust gates for 3D human action recognition,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9907 LNCS, pp. 816–833, 2016. doi: 10.1007/978-3-319-46487-9_50/TABLES/5. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-46487-9_50 [Pages 10 and 17.]
- [42] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, “View Adaptive Recurrent Neural Networks for High Performance Human Action Recognition From Skeleton Data,” pp. 2117–2126, 2017. [Pages 10 and 17.]
- [43] W. Zheng, L. Li, Z. Zhang, Y. Huang, and L. Wang, “Relational network for skeleton-based action recognition,” *Proceedings - IEEE International Conference on Multimedia and Expo*, vol. 2019-July, pp. 826–831, 7 2019. doi: 10.1109/ICME.2019.00147 [Pages 10 and 17.]
- [44] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, “A New Representation of Skeleton Sequences for 3D Action Recognition,” pp. 3288–3297, 2017. [Pages 10 and 17.]
- [45] C. Li, Q. Zhong, D. Xie, and S. Pu, “Co-occurrence Feature Learning from Skeleton Data for Action Recognition and Detection with Hierarchical Aggregation,” *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2018-July, pp. 786–792, 4 2018. doi: 10.24963/ijcai.2018/109. [Online]. Available: <https://arxiv.org/abs/1804.06055v1> [Pages 10 and 17.]
- [46] H. Duan, Y. Zhao, K. Chen, D. Lin, and B. Dai, “Revisiting Skeleton-Based Action Recognition,” pp. 2969–2978, 2022. [Online]. Available: <https://github.com/kennymckormick/pyskl>. [Pages 10 and 18.]

- [47] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A Simple Framework for Contrastive Learning of Visual Representations,” pp. 1597–1607, 11 2020. [Online]. Available: <https://proceedings.mlr.press/v119/chen20j.html> [Page 10.]
- [48] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum Contrast for Unsupervised Visual Representation Learning,” pp. 9729–9738, 2020. [Online]. Available: <https://github.com/facebookresearch/moco> [Page 10.]
- [49] J. B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, “Bootstrap your own latent: A new approach to self-supervised Learning,” *Advances in Neural Information Processing Systems*, vol. 2020-December, 6 2020. [Online]. Available: <https://arxiv.org/abs/2006.07733v3> [Pages 10 and 19.]
- [50] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, 10 2018. [Online]. Available: <https://arxiv.org/abs/1810.04805v2> [Pages 10, 18, and 30.]
- [51] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners.” [Online]. Available: <https://github.com/openai/gpt-2> [Pages 10, 18, 30, and 51.]
- [52] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked Autoencoders Are Scalable Vision Learners,” pp. 16 000–16 009, 2022. [Pages 10, 18, and 30.]
- [53] W. Wu, Y. Hua, C. Zheng, S. Wu, C. Chen, and A. Lu, “SkeletonMAE: Spatial-Temporal Masked Autoencoders for Self-supervised Skeleton Action Recognition,” *2023 IEEE International Conference on Multimedia and Expo Workshops*, 2023. [Pages 10 and 13.]

- [54] R. Xu, L. Huang, M. Wang, J. Hu, and W. Deng, “Skeleton2vec: A Self-supervised Learning Framework with Contextualized Target Representations for Skeleton Sequence,” 1 2024. [Online]. Available: <https://arxiv.org/abs/2401.00921v1> [Pages 10, 15, 19, 54, 55, and 56.]
- [55] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, “Attention is All you Need,” *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Pages 15 and 18.]
- [56] L. Shi, Y. Zhang, J. Cheng, and H. Lu, “Skeleton-Based Action Recognition With Directed Graph Neural Networks,” pp. 7912–7921, 2019. [Page 16.]
- [57] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, “Actional-Structural Graph Convolutional Networks for Skeleton-Based Action Recognition,” pp. 3595–3603, 2019. [Page 16.]
- [58] F. Ye, S. Pu, Q. Zhong, C. Li, D. Xie, and H. Tang, “Dynamic GCN: Context-enriched Topology Learning for Skeleton-based Action Recognition,” *MM 2020 - Proceedings of the 28th ACM International Conference on Multimedia*, pp. 55–63, 10 2020. doi: 10.1145/3394171.3413941. [Online]. Available: <https://dl.acm.org/doi/10.1145/3394171.3413941> [Page 16.]
- [59] K. Cheng, Y. Zhang, X. He, W. Chen, J. Cheng, and H. Lu, “Skeleton-Based Action Recognition With Shift Graph Convolutional Network,” pp. 183–192, 2020. [Online]. Available: <https://github.com/kchengiva/> [Page 16.]
- [60] K. Cheng, Y. Zhang, C. Cao, L. Shi, J. Cheng, and H. Lu, “Decoupling GCN with DropGraph Module for Skeleton-Based Action Recognition,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12369 LNCS, pp. 536–553, 2020. doi: 10.1007/978-3-030-58586-0_32/TABLES/6. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-58586-0_32 [Page 16.]
- [61] M. Korban and X. Li, “DDGCN: A Dynamic Directed Graph Convolutional Network for Action Recognition,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial*

- Intelligence and Lecture Notes in Bioinformatics*), vol. 12365 LNCS, pp. 761–776, 2020. doi: 10.1007/978-3-030-58565-5_45/TABLES/6. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-58565-5_45 [Page 16.]
- [62] Z. Chen, S. Li, B. Yang, Q. Li, and H. Liu, “Multi-Scale Spatial Temporal Graph Convolutional Network for Skeleton-Based Action Recognition,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, pp. 1113–1122, 5 2021. doi: 10.1609/AAAI.V35I2.16197. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/16197> [Page 16.]
- [63] Y. Chen, Z. Zhang, C. Yuan, B. Li, Y. Deng, and W. Hu, “Channel-wise Topology Refinement Graph Convolution for Skeleton-Based Action Recognition,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 13 339–13 348, 7 2021. doi: 10.1109/ICCV48922.2021.01311. [Online]. Available: <https://arxiv.org/abs/2107.12213v2> [Page 16.]
- [64] Y. F. Song, Z. Zhang, C. Shan, and L. Wang, “Constructing Stronger and Faster Baselines for Skeleton-based Action Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 1474–1488, 6 2021. doi: 10.1109/TPAMI.2022.3157033. [Online]. Available: <https://arxiv.org/abs/2106.15125v2> [Page 16.]
- [65] H.-g. Chi, M. H. Ha, S. Chi, S. W. Lee, Q. Huang, and K. Ramani, “InfoGCN: Representation Learning for Human Skeleton-Based Action Recognition,” pp. 20 186–20 196, 2022. [Page 16.]
- [66] L. Ke, K. C. Peng, and S. Lyu, “Towards To-a-T Spatio-Temporal Focus for Skeleton-Based Action Recognition,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, pp. 1131–1139, 6 2022. doi: 10.1609/AAAI.V36I1.19998. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/19998> [Page 16.]
- [67] W. Xiang, C. Li, Y. Zhou, B. Wang, and L. Zhang, “Generative Action Description Prompts for Skeleton-based Action Recognition,” pp. 10 276–10 285, 2023. [Online]. Available: <https://github.com/> [Page 17.]
- [68] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, “An End-to-End Spatio-Temporal Attention Model for Human Action Recognition

- from Skeleton Data,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, pp. 4263–4270, 2017. doi: 10.1609/AAAI.V31I1.11212. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/11212> [Page 17.]
- [69] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, “Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN,” pp. 5457–5466, 2018. [Page 17.]
- [70] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan, “An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition,” pp. 1227–1236, 2019. [Page 17.]
- [71] R. Zhao, K. Wang, H. Su, and Q. Ji, “Bayesian Graph Convolution LSTM for Skeleton Based Action Recognition,” pp. 6882–6892, 2019. [Page 17.]
- [72] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. Mccandlish, A. Radford, I. Sutskever, and D. Amodei, “Language Models are Few-Shot Learners,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020. [Online]. Available: <https://commoncrawl.org/the-data/> [Pages 18 and 30.]
- [73] OpenAI, :, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo,

- C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O’Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. d. A. B. Peres, M. Petrov, H. P. d. O. Pinto, Michael, Pokorny, M. Pokrass, V. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph, “GPT-4 Technical Report,” 3 2023. [Online]. Available: <https://arxiv.org/abs/2303.08774v4> [Page 18.]
- [74] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An Image is Worth 16x16 Words:

- Transformers for Image Recognition at Scale,” *ICLR 2021 - 9th International Conference on Learning Representations*, 10 2020. [Online]. Available: <https://arxiv.org/abs/2010.11929v2> [Pages 18 and 30.]
- [75] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows,” pp. 10 012–10 022, 2021. [Online]. Available: <https://github.com>. [Pages 18 and 30.]
- [76] L. Shi, Y. Zhang, J. Cheng, and H. Lu, “Decoupled Spatial-Temporal Attention Network for Skeleton-Based Action-Gesture Recognition,” 2020. [Online]. Available: <https://github.com/lshiwjx/DSTA-Net> [Pages 18 and 53.]
- [77] L. Li, M. Wang, B. Ni, H. Wang, J. Yang, and W. Zhang, “3D Human Action Representation Learning via Cross-View Consistency Pursuit,” 4 2021. [Online]. Available: <https://github.com/LinguoLi/CrosSCLR> <https://arxiv.org/abs/2104.14466v2> [Page 19.]
- [78] Y. Su, G. Lin, and Q. Wu, “Self-Supervised 3D Skeleton Action Representation Learning With Motion Consistency and Continuity,” pp. 13 328–13 338, 2021. [Page 19.]
- [79] F. M. Thoker, H. Doughty, and C. G. Snoek, “Skeleton-Contrastive 3D Action Representation Learning,” *MM 2021 - Proceedings of the 29th ACM International Conference on Multimedia*, pp. 1655–1663, 10 2021. doi: 10.1145/3474085.3475307. [Online]. Available: <https://dl.acm.org/doi/10.1145/3474085.3475307> <https://arxiv.org/abs/2108.03656v1> [Page 19.]
- [80] T. Guo, H. Liu, Z. Chen, M. Liu, T. Wang, and R. Ding, “Contrastive Learning from Extremely Augmented Skeleton Sequences for Self-Supervised Action Recognition,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, pp. 762–770, 6 2022. doi: 10.1609/AAAI.V36I1.19957. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/19957> [Page 19.]
- [81] H. Zhang, Y. Hou, W. Zhang, and W. Li, “Contrastive Positive Mining for Unsupervised 3D Action Representation Learning,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13664 LNCS,

- pp. 36–51, 2022. doi: 10.1007/978-3-031-19772-7_3/FIGURES/6. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-19772-7_3 [Page 19.]
- [82] Y. Mao, W. Zhou, Z. Lu, J. Deng, and H. Li, “CMD: Self-supervised 3D Action Representation Learning with Cross-Modal Mutual Distillation,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13663 LNCS, pp. 734–752, 2022. doi: 10.1007/978-3-031-20062-5_42/FIGURES/4. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-20062-5_42 [Page 19.]
- [83] O. Moliner, S. Huang, and K. K. Kalleåström, “Bootstrapped Representation Learning for Skeleton-Based Action Recognition,” pp. 4154–4164, 2022. [Page 19.]
- [84] Y. Hua, W. Wu, C. Zheng, A. Lu, M. Liu, C. Chen, and S. Wu, “Part Aware Contrastive Learning for Self-Supervised Action Recognition,” *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2023-Augus, pp. 855–863, 5 2023. doi: 10.24963/ijcai.2023/95. [Online]. Available: <https://arxiv.org/abs/2305.00666v2> [Page 19.]
- [85] Y. Zhou, H. Duan, A. Rao, B. Su, and J. Wang, “Self-supervised Action Representation Learning from Partial Spatio-Temporal Skeleton Sequences,” *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, vol. 37, pp. 3825–3833, 2 2023. doi: 10.1609/aaai.v37i3.25495. [Online]. Available: <http://arxiv.org/abs/2302.09018><http://dx.doi.org/10.1609/aaai.v37i3.25495> [Page 19.]
- [86] A. Shah, A. Roy, K. Shah, S. Mishra, D. Jacobs, A. Cherian, and R. Chellappa, “HaLP: Hallucinating Latent Positives for Skeleton-Based Self-Supervised Learning of Actions,” pp. 18 846–18 856, 2023. [Online]. Available: <https://github.com/anshulbshah/HaLP>. [Page 19.]
- [87] J. Dong, S. Sun, Z. Liu, S. Chen, B. Liu, and X. Wang, “Hierarchical Contrast for Unsupervised Skeleton-Based Action Representation Learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 1, pp. 525–533, 6 2023. doi: 10.1609/AAAI.V37I1.25127. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/25127> [Page 19.]

- [88] Q. Nie, Z. Liu, and Y. Liu, “Unsupervised 3D Human Pose Representation with Viewpoint and Pose Disentanglement,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12364 LNCS, pp. 102–118, 7 2020. [Online]. Available: https://arxiv.org/abs/2007.07053v2https://link.springer.com/chapter/10.1007/978-3-030-58529-7_7 [Page 19.]
- [89] S. Yang, J. Liu, S. Lu, M. H. Er, and A. C. Kot, “Skeleton Cloud Colorization for Unsupervised 3D Action Representation Learning,” pp. 13 423–13 433, 2021. [Page 19.]
- [90] Y. Chen, L. Zhao, J. Yuan, Y. Tian, Z. Xia, S. Geng, L. Han, and D. N. Metaxas, “Hierarchically Self-supervised Transformer for Human Skeleton Representation Learning,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13686 LNCS, pp. 185–202, 2022. doi: 10.1007/978-3-031-19809-0_11/TABLES/5. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-19809-0_11 [Page 20.]
- [91] B. Kim, H. J. Chang, J. Kim, and J. Y. Choi, “Global-Local Motion Transformer for Unsupervised Skeleton-Based Action Learning,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13664 LNCS, pp. 209–225, 2022. doi: 10.1007/978-3-031-19772-7_13/TABLES/5. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-19772-7_13 [Page 20.]
- [92] H. Yan, Y. Liu, Y. Wei, Z. Li, G. Li, and L. Lin, “SkeletonMAE: Graph-based Masked Autoencoder for Skeleton Sequence Pre-training,” pp. 5606–5618, 2023. [Page 20.]
- [93] J. Snell, K. Swersky, and T. R. Zemel, “Prototypical Networks for Few-shot Learning,” *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Page 20.]
- [94] M. Guo, E. Chou, D.-A. Huang, S. Song, S. Yeung, and L. Fei-Fei, “Neural Graph Matching Networks for Fewshot 3D Action Recognition,” pp. 653–669, 2018. [Page 20.]

- [95] L. Zhao and L. Akoglu, “PairNorm: Tackling Oversmoothing in GNNs,” *8th International Conference on Learning Representations, ICLR 2020*, 9 2019. [Online]. Available: <https://arxiv.org/abs/1909.12223v2> [Page 20.]
- [96] Y. Rong, W. Huang, T. Xu, and J. Huang, “DropEdge: Towards Deep Graph Convolutional Networks on Node Classification,” *8th International Conference on Learning Representations, ICLR 2020*, 7 2019. [Online]. Available: <https://arxiv.org/abs/1907.10903v4> [Page 20.]
- [97] N. Ma, H. Zhang, X. Li, S. Zhou, Z. Zhang, J. Wen, H. Li, J. Gu, and J. Bu, “Learning Spatial-Preserved Skeleton Representations for Few-Shot Action Recognition,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13664 LNCS, pp. 174–191, 2022. doi: 10.1007/978-3-031-19772-7_11/FIGURES/5. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-19772-7_11 [Page 20.]
- [98] L. Wang and P. Koniusz, “Temporal-Viewpoint Transportation Plan for Skeletal Few-shot Action Recognition,” pp. 4176–4193, 2022. [Page 20.]
- [99] X. Liu, S. Zhou, L. Wang, and G. Hua, “Parallel Attention Interaction Network for Few-Shot Skeleton-Based Action Recognition,” pp. 1379–1388, 2023. [Page 20.]
- [100] S. Yang, J. Liu, S. Lu, E. Meng Hwa, L. Fellow, and A. C. Kot, “One-Shot Action Recognition via Multi-Scale Spatial-Temporal Skeleton Matching,” 7 2023. [Online]. Available: <https://arxiv.org/abs/2307.07286v2> [Page 20.]
- [101] A. Zhu, Q. Ke, M. Gong, and J. Bailey, “Adaptive Local-Component-Aware Graph Convolutional Network for One-Shot Skeleton-Based Action Recognition,” pp. 6038–6047, 2023. [Page 20.]
- [102] C. Cuevas, D. Quilón, and N. García, “Techniques and applications for soccer video analysis: A survey,” *Multimedia Tools and Applications*, vol. 79, no. 39-40, pp. 29 685–29 721, 10 2020. [Online]. Available: <https://link.springer.com/article/10.1007/s11042-020-09409-0> [Page 24.]

- [103] R. Sanford, S. Gorji, L. G. Hafemann, B. Pourbabaei, and M. Javan, "Group Activity Detection From Trajectory and Video Data in Soccer," pp. 898–899, 2020. [Page 24.]
- [104] Y. Jiang, K. Cui, L. Chen, C. Wang, and C. Xu, "SoccerDB: A Large-Scale Database for Comprehensive Video Understanding," *MMSports 2020 - Proceedings of the 3rd International Workshop on Multimedia Content Analysis in Sports*, pp. 1–8, 10 2020. doi: 10.1145/3422844.3423051. [Online]. Available: <https://dl.acm.org/doi/10.1145/3422844.3423051> [Page 24.]
- [105] S. Hurault, C. Ballester, and G. Haro, "Self-Supervised Small Soccer Player Detection and Tracking," *MMSports 2020 - Proceedings of the 3rd International Workshop on Multimedia Content Analysis in Sports*, pp. 9–18, 10 2020. doi: 10.1145/3422844.3423054. [Online]. Available: <https://dl.acm.org/doi/10.1145/3422844.3423054> [Page 24.]
- [106] B. Vanderplaetse and S. Dupont, "Improved Soccer Action Spotting Using Both Audio and Video Streams," pp. 896–897, 2020. [Online]. Available: <https://github.com/DTaoo/VGGish> [Page 24.]
- [107] S. Giancola and B. Ghanem, "Temporally-Aware Feature Pooling for Action Spotting in Soccer Broadcasts," pp. 4490–4499, 2021. [Page 24.]
- [108] A. Cioppa, A. Deliege, F. Magera, S. Giancola, O. Barnich, B. Ghanem, and M. Van Droogenbroeck, "Camera Calibration and Player Localization in SoccerNet-v2 and Investigation of Their Representations for Action Spotting," pp. 4537–4546, 2021. [Online]. Available: <https://soccer-net.org/>. [Page 24.]
- [109] F. Yang, S. Odashima, S. Masui, and S. Jiang Fujitsu Research, "Hard To Track Objects With Irregular Motions and Similar Appearances? Make It Easier by Buffering the Matching Space," pp. 4799–4808, 2023. [Page 24.]
- [110] C. Plizzari, M. Cannici, and M. Matteucci, "Skeleton-based Action Recognition via Spatial and Temporal Transformer Networks," *Computer Vision and Image Understanding*, vol. 208-209, 8 2020. doi: 10.1016/j.cviu.2021.103219. [Online]. Available: <http://arxiv.org/abs/2008.07404><http://dx.doi.org/10.1016/j.cviu.2021.103219> [Page 53.]

- [111] Z. Tong, Y. Song, J. Wang, and L. Wang, “VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 10 078–10 093, 12 2022. [Online]. Available: <https://github.com/MCG-NJU/VideoMAE>. [Page 54.]

Appendix A

Details of Frame-level Data Preparation

A.1 Data Cleaning for Pre-training Data

Data cleaning is necessary to enhance the quality of the pre-training data. This process includes removing corrupted frames, as well as noisy frames in both spread and motion.

Removal of corrupted frames Missing frames and missing joints are not rare in skeleton sequences, often resulting in values being recorded as NaN (Not a Number). To ensure these NaN values do not adversely affect the pre-training process, we remove any frames that contain NaN values.

Removal of spread-noisy frames When a joint's position deviates significantly from its normal location, the frame is likely to contain noisy data. To address this, we calculate the spread of a frame in both the x and y directions and filter out noisy frames by setting a threshold on the ratio of the x-direction spread to the y-direction spread. Formally, the ratio r is derived by

$$\begin{aligned} s_x &= \max(\mathbf{x}_{:,0}) - \min(\mathbf{x}_{:,0}) \\ s_y &= \max(\mathbf{x}_{:,1}) - \min(\mathbf{x}_{:,1}) \\ r &= \min(s_x/s_y, s_y/s_x) \end{aligned} \tag{A.1}$$

The threshold is set to 0.15. Namely, we filter out frames with $r < 0.15$.

Removal of motion-noisy frames There is a physiological upper limit for human joint movement speed. Therefore, joints that move excessively fast are likely to be noisy data. Although our training only involves single frames, which don't involve joint movement, frames with noisy joint motion

often have structural inaccuracies. To filter out motion-noisy frames, we calculate the instantaneous speed of each joint within a frame. Frames in which any joint moves faster than 10 times the z-direction spread per second are excluded. Given that referees typically stand or run, the z-direction spread, approximately equal to the referee’s height, is usually more than 1.5 meters. Thus, the threshold normally exceeds 15 meters per second—a speed unreachable by humans.

$$\begin{aligned} s_z &= \max(\mathbf{x}_{:,2}^t) - \min(\mathbf{x}_{:,2}^t) \\ \mathbf{v} &= \mathbf{x}^{t+1} - \mathbf{x}^t \\ v_{max} &= \max_i(\|\mathbf{v}_i\|) \end{aligned} \tag{A.2}$$

We filter out frames with $v_{max} > 10 \cdot s_z$.

A.2 Classification Dataset for Stage 1

A.2.1 Manual Annotation

Manually annotating data is the most straightforward and reliable way to get labeled data, yet it is also inefficient. Acquiring a large-scale dataset through manual annotation alone is impractical. However, manual annotations can serve as valuable starting points for implementing the automatic and semi-automatic strategies outlined below.

A.2.2 Cosine Similarity Matching

Frames within the same class should exhibit similar characteristics in the latent space of the pre-trained model. Therefore, we propose to leverage cosine similarity within this latent space to automatically derive more labeled data from unlabeled datasets, leveraging a few manually annotated examples as a reference. To determine the similarity between an unlabeled frame and a known class, we calculate the minimum cosine distance between the unlabeled frame and all labeled frames within that class. We then apply two specific thresholds to this calculated similarity to label new frames:

- **Similarity threshold t_s :** A frame is only labeled if its minimum cosine distance to the class is less than t_s .
- **Duration threshold t_d :** A frame is only labeled if it consistently meets the similarity criterion across t_d consecutive frames.

To prevent duplication and maintain diversity within the dataset, only the center frame of a ten-second sequence is labeled if it satisfies both thresholds. The optimal values for these thresholds vary depending on the type of action being analyzed. Empirically, we find that the values listed in Table A.1 are effective for the target classes for both Stage 1 and Stage 2.

	Similarity Threshold	Duration Threshold
THROW_IN	0.01	5
GOAL_KICK	0.01	10
CORNER_KICK	0.03	10
SUBSTITUTION	0.02	20
RAISE_FLAG_VERTICALLY	0.02	10
FAR_SIDE_OFFSIDE	0.03	10
NEAR_SIDE_OFFSIDE	0.05	10

Table A.1: Empirical optimal values for cosine similarity matching for different action types. The frame rate is 25Hz.

Frames annotated using this strategy typically exhibit an error rate of approximately 5% to 20%, depending on the action type. Therefore, in practice, we employ a semi-automatic approach: initially, frames are annotated using the two thresholds, and then the auto-annotated frames are manually verified to ensure the dataset’s quality.

A.2.3 Auto-annotation with Binary Classification Models

Another approach to utilizing the pre-trained model involves specifically training a model based on it for automatic annotation. For each action class, we fine-tune the pre-trained model to get a binary classification model that can determine whether a frame belongs to the selected action type or not. The main advantage of using specialized models for annotation is that it allows more complex decision boundaries for action recognition, which are more expressive than simple spherical boundaries in cosine similarity matching. The detailed steps to annotate data for each action class with binary classification models are outlined below:

(1) Create a binary classification dataset for the selected action

For each class, we create a separate binary classification dataset. All existing labeled frames are categorized into positive and negative categories. Frames that belong to the selected class are labeled as

positive, while all those that do not are assigned a negative label. Typically, the number of negative samples substantially exceeds the number of positive samples. To balance this, we randomly select a subset of the negative samples to ensure a fixed positive-to-negative ratio. This ratio is maintained at 1/4 for all classes in our experiments, except for `RAISE_FLAG_VERTICALLY`, where it is set to 1/8. We construct the binary dataset after cosine similarity matching. So, the dataset produced by this process usually contains hundreds of frames, which is sufficient to train decent models.

(2) Fine-tune the pre-trained model on the collected dataset To get a capable binary classification model for data annotation, we fine-tuned the pre-trained model using the datasets specifically collected for binary classification for each class. As a result, a model is trained for each class to annotate frames. We follow the fine-tuning methods described in Section 3.2.2, without residual module and weighted resampling. The pre-trained model and the additional linear layer are trained for 10 epochs. We set the batch size to 32 and used a learning rate of $5e-4$. We use the pre-trained model trained on unlabelled data, which will be detailed in Section 3.3.2.

(3) Annotate unlabeled frames using the fine-tuned model with thresholding We utilize the fine-tuned model to predict the probability of each unlabeled frame being a positive example for each class. The thresholding strategy is similar to that described in Cosine Similarity Matching, with the modification that we now threshold based on classification probability. The thresholds used for annotation are:

- **Probability threshold t_p :** A frame is only annotated if the softmax probability predicted by the model exceeds t_p .
- **Duration threshold t_d :** A frame is only annotated if it consistently exceeds the probability threshold over t_d consecutive frames.

The duration threshold (t_d) is set at 20 for all classes. The probability threshold (t_p) is consistently set at 0.99 across all classes, with the exception of the `SUBSTITUTION` class, which has a probability threshold of 0.95.

Based on the above three steps, we collected a large-scale labeled frame-level classification dataset. As indicated in Table A.2, we trained five models corresponding to five action types. These fine-tuned models demonstrated

high accuracy on both the validation and test sets. The number of frames ultimately annotated was 1 to 5 times greater than the data initially used for training, showcasing the effectiveness of this approach.

Class	Binary Dataset Train/Valid/Test	Fine-tuned Model Valid/Test Acc.	# Frames Annotated
THROW_IN	685+2740/59+239/60+238	100.00/99.66	4415
GOAL_KICK	831+3324/67+241/56+252	100.00/99.35	2247
CORNER_KICK	429+1716/27+125/34+118	100.00/99.34	1119
SUBSTITUTION	311+1244/24+106/28+102	100.00/100.00	635
RAISE_FLAG_VERTICALLY	456+3648/34+322/45+311	100.00/99.44	353

Table A.2: Frames annotated automatically by binary classification models (the last column). The number before the '+' sign represents the count of positive samples, while the number after the '+' sign indicates the count of negative samples.

For NO_DECISION actions, we developed a sequence-level binary classification model using the ST-GCN architecture, as detailed in Section 2.1.1. We utilized existing event labels such as 'whistle' and 'ball dead', labeling 25,592 sequences, each 10 seconds in length. Sequences with event labels are labeled HAS_DECISION, while sequences without event labels are labeled NO_DECISION. These sequences were then split into training, validation, and test sets in an 8/1/1 ratio. A 9-layer ST-GCN with a hidden size of 64 was trained on this binary classification dataset using a learning rate of 3e-4 over 20 epochs. As shown in Table A.3, the performance of the ST-GCN model reached 95.4% on the validation set and 94.8% on the test set. This model was subsequently applied to predict 20,285 unlabeled 10-second long sequences, of which 18,889 were predicted as NO_DECISION sequences. We then randomly picked one frame from each sequence to be labeled as NO_DECISION.

Class	Binary Dataset Train/Valid/Test	Fine-tuned Model Valid/Test Acc.	# Frames Annotated
NO_DECISION	20,474/2,559/2,559	95.4/94.8	18,889

Table A.3: Frames annotated by the sequence-level binary ST-GCN model for NO_DECISION.

A.2.4 Throw-in Direction Division

To achieve a more fine-grained classification, we further divide the class `THROW_IN` into `THROW_IN_LEFT` and `THROW_IN_RIGHT`, as it can provide richer information for future applications, such as event detection. We employ similar strategies as previously described. Initially, we manually annotate a small set of samples for both `THROW_IN_LEFT` and `THROW_IN_RIGHT`. Subsequently, we utilize cosine similarity matching to expand our labeled dataset slightly. Following this, we train a binary classification model using the gathered data and then use this model to annotate all `THROW_IN` frames.

A.2.5 Data Cleaning Conditioned on Arm Angles and Joint Distances

To further improve the quality of the dataset, we implement a denoising process based on arm angle conditions and joint distance conditions. This strategy stems from observing that the initial versions of fine-tuned models sometimes performed poorly, particularly when the assistant referee was scratching or swiping sweat, and some no-action frames were misclassified as corner kicks and goal kicks. We tailor the denoising conditions to suit different types of actions, as follows:

`THROW_IN` We calculate the distance between the shoulder and the wrist for both limbs and establish a threshold of 0.4 meters to filter out noisy frames since actions with the wrist close to the shoulder are likely to be scratching. For `THROW_IN_LEFT`, the condition applies only to the left limbs, and for `THROW_IN_RIGHT`, it applies only to the right limbs. Frames where the wrist-to-shoulder distance is less than 0.4 meters are labeled as noisy.

`GOAL_KICK` To denoise goal kick frames, we compute two cosine values: $\cos Z$, which measures the cosine value of the angle between the right elbow-wrist vector and the z-axis, and $\cos Y$, which measures the cosine value of the angle between the right elbow-wrist vector and the y-axis. Ideally, during a goal kick signal, the arm should be horizontally oriented toward the goal area, aligning parallel with the goal line. Therefore, we set the threshold at 0.5 for $|\cos Z|$ and 0.7 for $|\cos Y|$. Frames with the absolute value of these cosine values below these thresholds are excluded as noisy data.

CORNER_KICK The model occasionally confuses the signal of a corner kick with no-action frames. To address this, we calculate the cosine value between the right elbow-wrist vector and the z-axis, setting a threshold at -0.8. Frames where the cosine value falls below this threshold are considered no-action, indicating that the arm is not sufficiently raised.

RAISE_FLAG_VERTICALLY Similarly, we calculate the cosine value between the elbow-wrist vector and the z-axis for both forearms, setting a threshold of 0.95. Frames with the cosine values of both arms smaller than 0.95 are considered not sufficiently raised vertically and are thus labeled as noisy.

Note that arm angles and joint distances are not golden criteria for identifying actions. Therefore, in practice we performed the aforementioned thresholding in a semi-automatic manner, meaning we manually review frames labeled as noisy and remove them upon manual confirmation. Eventually, 14 **THROW_IN** frames, 14 **GOAL_KICK** frames, 79 **CORNER_KICK** frames and 257 **RAISE_FLAG_VERTICALLY** frames are removed.

A.2.6 Data Supplementation

In addition to removing noisy frames such as scratching and wiping sweat, it is also crucial to categorize these frames under the **NO_DECISION** class to enhance model robustness. Therefore, we measure the distance from each wrist to the nose and the neck, setting a threshold of 0.2 meters. Frames, where the minimum distance from either the wrist to the nose or the neck is less than 0.2 meters, are regarded as scratching actions and annotated as **NO_DECISION**. We supplement the **NO_DECISION** data with an additional 1929 frames.

For the **RAISE_FLAG_VERTICALLY** class, we identify additional data by thresholding the angle of the forearms. Similar to our data cleaning process, we confirm frames where the cosine value exceeds 0.97. We specifically review frames within a 10-second window surrounding existing **RAISE_FLAG_VERTICALLY** frames, proceeding in 1-second increments.

	Manual	Cosine	Model	Supplement	Total
NO_DECISION	5		6695	1929	8722*
THROW_IN_LEFT	6	500	2628		3134
THROW_IN_RIGHT	9	277	935		2022
GOAL_KICK	15	935	2236		3186
CORNER_KICK	10	475	1044		1529
SUBSTITUTION	4	359	635		998
RAISE_FLAG_VERTICALLY	3	225	353	547	1128

Table A.4: Statistics of final collected data for assistant referee action recognition Stage 1. *NO_DECISION frames includes 93 frames labeled noisy from other classes in data cleaning.

With all the above strategies, the data we collected is presented in Table A.4. Note that the numbers are not exactly the same as those in Table A.2 due to data cleaning.

A.3 Classification Dataset for Stage 2

A.3.1 Cosine similarity matching

As shown in Table A.1, we employ the same collection strategy as in Stage 1 for the FAR_SIDE_OFFSIDE and NEAR_SIDE_OFFSIDE frames. This process is semi-automatic, as we manually verify the annotated results. Note that we did not apply similar matching for MIDDLE_SIDE_OFFSIDE, because the signal is identical to that of GOAL_KICK, except for the location of the assistant referee raising the flag.

A.3.2 Binary classification model

Since offside events occur less frequently than other events like goal kicks or corner kicks in football games, the collection of offside frames is relatively small. Training a model on such limited data is generally not ideal. However, in an effort to expand the dataset, particularly for the class with the fewest frames, NEAR_SIDE_OFFSIDE, we attempted to annotate frames using a fine-tuned binary classification model, following the strategy described in Section A.2.3. As shown in Table A.5, the binary classification dataset comprises 89 positive training frames and 712 negative frames, which is significantly smaller than the datasets listed in Table A.2. The validation set includes

17 positive samples and 51 negative samples, while the test set contains 13 positive samples and 55 negative samples. Despite achieving a classification accuracy of 100% on both the validation and test sets, the model lacks sufficient generalization for reliable auto-annotation. We set a probability threshold of 0.99 and a duration threshold of 20. Nevertheless, the quality of auto-annotated data remained not ideal, with most annotated frames being rejected in manual verification. In conclusion, the strategy of training a binary classification model is less effective with insufficient data.

	Train	Valid	Test
NEAR_SIDE_OFFSIDE	89	17	13
Non-NEAR_SIDE_OFFSIDE	712	51	55
Total	801	68	68

Table A.5: Dataset collected for training binary classification model for NEAR_SIDE_OFFSIDE

A.3.3 By-product of the goal kick binary model

MIDDLE_SIDE_OFFSIDE can be annotated as a by-product of the binary classification model for goal kicks in Stage 1. This is feasible because GOAL_KICK and MIDDLE_SIDE_OFFSIDE share identical gestures; however, GOAL_KICK only occurs around the goal area, whereas MIDDLE_SIDE_OFFSIDE can occur anywhere along the touchline. Therefore, if a GOAL_KICK is annotated far from the goal area, it is likely a MIDDLE_SIDE_OFFSIDE. We establish a distance threshold of 40 meters between the referee and the halfway line. If a GOAL_KICK is predicted by the binary classification model (meeting two thresholds) and the distance between the assistant referee and the halfway line is less than 40 meters, we label the frame as MIDDLE_SIDE_OFFSIDE.

A.3.4 Manual annotation from raising flags

Offside frames frequently occur immediately after a RAISE_FLAG_VERTICALLY action. Given that all automated annotation methods require manual verification, directly annotating frames following RAISE_FLAG_VERTICALLY is not only more efficient but also ensures higher data quality.

	Manual	Cosine	Model	Total
FAR_SIDE_OFFSIDE	268	64		332
MIDDLE_SIDE_OFFSIDE	261		164	425
NEAR_SIDE_OFFSIDE	88	16	60	164

Table A.6: Statistics of three extra classes for assistant referee action recognition Stage 2.

The final collected data for three extra offside classes are shown in Table A.6, with a significant portion of the data derived from manual annotations.

Appendix B

Details of Sequence-level Data Preparation

B.1 Data Cleaning for Pre-training Data

The quality of data is essential for effective pre-training. To enhance the quality of the pre-training data, we propose implementing denoising techniques to address three specific types of noise:

Spread-noisy We follow the definition of a frame being noisy in terms of spread as specified in Section 3.3.1.1, using the same threshold of 0.15. To determine if a skeleton sequence is noisy based on spread, we calculate the proportion of spread-noisy frames within the sequence and set a threshold of 0.2. If more than 20% of the frames in a sequence are spread-noisy, we exclude that sequence from our dataset.

Motion-noisy Similarly, we follow the motion-noisy definition as outlined in Section 3.3.1.1. Since motion is one of our prediction targets, noisy motions could significantly affect the training process. Therefore, we implement a strict rule, excluding any sequences from the dataset that contain even a single motion-noisy frame.

Pelvis-centered motion noisy Even when the motion of a skeleton sequence remains within the threshold, the relative motion of joints can still be excessively large, which may affect training stability. Given that our pre-training involves pelvis-centered preprocessing, we specifically calculate the motion relative to the pelvis and filter out noisy skeleton sequences. Following the calculation of joint motions, we calculate the pelvis-centered motion based on the pelvis-centered input sequence. Formally, for the frame at time step t

in the skeleton sequence, we compute the maximum pelvis-centered velocity:

$$\begin{aligned} s_z^{pelvis} &= \max(\mathbf{x}_{t::,2}^{pelvis}) - \min(\mathbf{x}_{t::,2}^{pelvis}) \\ \mathbf{v}^{pelvis} &= \mathbf{x}_{t+1}^{pelvis} - \mathbf{x}_t^{pelvis} \\ v_{max}^{pelvis} &= \max_i(\|\mathbf{v}_i^{pelvis}\|) \end{aligned} \quad (\text{B.1})$$

As in the pelvis-centered space, the maximum speed achievable by humans should be lower, we establish a noise criterion defined as $v_{max}^{pelvis} > 6 \cdot s_z^{pelvis}$. Sequences containing any frame that exceeds this threshold are filtered out.

B.2 Sequence Classification Dataset for Fine-tuning

B.2.1 Manual Annotation

Compared to frame-level assistant referee recognition, we rely more heavily on manual annotation to collect the dataset for the sequence classification task, particularly for specific classes such as `DIRECT_FREE_KICK` and `OTHER_DECISION`. Designing a reliable automatic annotation rule for these classes is extremely challenging. Even with the animation of the skeleton sequence, determining the action type can be challenging for humans and often requires viewing match videos for accurate identification. So, although manual annotation is less efficient, it provides a reliable way of collecting high-quality data. Therefore, in most of our subsequent strategies, we implement a semi-automated approach to data collection, combining automated detection methods with manual verification to balance efficiency and maintain the data quality.

B.2.2 Converting from Frame-level Data

Considering that for some certain classes, specifically `THROW_IN`, `GOAL_KICK`, and `CORNER_KICK`, the referee and the assistant referee often react to events almost simultaneously, we can leverage existing labeled data from the last chapter to annotate skeleton sequences. Since we already have labeled frames for assistant referees from these actions, as detailed in Section 3.3.1.2, we propose extracting skeleton sequences for referees around these labeled frames. This approach will allow us to efficiently gather a substantial amount of labeled referee skeleton sequences for these classes.

Specifically, for each frame labeled as `THROW_IN`, `GOAL_KICK`, or `CORNER_KICK` for assistant referees, we extract a 4-second sequence centered on the frame. Preliminary observations indicate that referees typically signal the same action type within this 4-second window, supporting the choice of the window length.

Collecting Data for `RAISE_HAND_VERTICALLY` In addition to `THROW_IN`, `GOAL_KICK`, and `CORNER_KICK`, we also collect `INDIRECT_FREE_KICK` sequences with a similar approach. The signal for an `INDIRECT_FREE_KICK` involves the referee raising one hand vertically. Unlike other signals, `INDIRECT_FREE_KICK` typically does not require a specific response from assistant referees. However, most `INDIRECT_FREE_KICK` calls are due to offsides, during which the assistant referee raises the flag, and we have existing data for these instances. Therefore, we design a strategy to initially collect frame-level data for main referees signaling with a raised hand and subsequently extracting sequence-level data using a similar methodology as for `THROW_IN`, `GOAL_KICK`, and `CORNER_KICK`.

To collect frame-level data for the main referee raising their hand, we use the same methodology used for collecting the frame-level dataset as described in Section 3.3.1.2. Initially, we manually annotate referee frames within a 10-second window following a `RAISE_FLAG_VERTICALLY` signal from assistant referees, resulting in 887 frames newly labeled as `RAISE_HAND_VERTICALLY`. We then use cosine similarity matching to identify an additional 159 labeled frames. Subsequently, we fine-tuned a binary classification model based on the frame-level pre-trained model. This model was used to annotate 3,317 more frames, with a confidence threshold of 0.99 and a duration threshold of 25 frames. In total, we collected 4,363 frames labeled `RAISE_HAND_VERTICALLY` for the main referee. Finally, we converted these collected frames into sequence data by extracting 4-second sequences centered on these frames.

Label Translation for Throw-ins There is a minor difference between the action types for assistant referees and the main referees concerning throw-ins. For assistant referees, the classifications are direction-based, labeled as `THROW_IN_LEFT` and `THROW_IN_RIGHT`, ensuring consistent predictions for both assistants. In contrast, for the main referee, we use `THROW_IN_HOME` and `THROW_IN_AWAY` to differentiate the subsequent attacking directions because the main referee may use either hand to signal the direction, and there are only two possible attacking directions.

In our pitch coordinate system, the origin is located at the center of the pitch, the x-axis runs along the line connecting the two goals, and the

y-axis runs along the halfway line. We define `THROW_IN_HOME` as the situation where the team attacking towards the positive x-direction possesses the ball and earns a throw-in, while `THROW_IN_AWAY` corresponds to the team attacking towards the negative x-direction earning a throw-in. To translate labels from assistant referees to the main referee, we consider the position of the assistant referee. Specifically, the first assistant referee, positioned with a y-coordinate less than 0, has their `THROW_IN_RIGHT` labeled as `THROW_IN_HOME` and `THROW_IN_LEFT` as `THROW_IN_AWAY` for the main referee. Conversely, the second assistant referee, positioned with a y-coordinate greater than 0, has their `THROW_IN_LEFT` labeled as `THROW_IN_HOME` and `THROW_IN_RIGHT` as `THROW_IN_AWAY` for the main referee.

Using Conditioning to Exclude Non-Action Sequences Another challenge is that, unlike assistant referees, the main referee does not necessarily respond to every throw-in. At times, for clear throw-ins, the referee may not show any signal. To avoid including these instances in our data, we use rules conditioned on arm angles to filter out sequences where the referee does not signal anything. The rules are based on the following indicators:

- Minimum Absolute Cosine Angle of the Forearm** $\min(|\cos \theta^{forearm}|)$
 The forearm vector is defined as the vector extending from the elbow to the wrist. We compute the absolute cosine value between the forearm vector and the unit z vector, determining the minimum value for both forearms throughout the sequence. If the arms remain downward, they are almost parallel to the z-axis, resulting in an absolute cosine value close to 1. Conversely, if the forearm is raised horizontally at any point, this indicator will approach zero.
- Minimum Absolute Cosine Angle of the Upper Arm** $\min(|\cos \theta^{upperarm}|)$
 The upper arm vector connects the shoulder to the elbow. Similarly, we compute the absolute cosine value between the upper arm vector and the unit z vector and calculate the minimum value for both upper arms throughout the sequence. This indicator functions similarly to the forearm indicator, offering us more precise control to filter out non-action sequences.
- Maximum Elbow-shoulder Difference along z-direction** $\max(d_z^{elbow} - d_z^{shoulder})$
 We calculate the difference in the z-direction between the elbow and the shoulder and then determine the maximum z-direction difference for both arms throughout the sequence. This indicator represents the highest position reached by the arms in the sequence.

We implement specific conditions for different action types to exclude non-action sequences based on the characteristics of each action type. The conditions for each action type are detailed as follows:

THROW_IN For both `THROW_IN_HOME` and `THROW_IN_AWAY` sequences, we filter out sequences with the arm never raised horizontally, implemented with conditions

- $\min(|\cos \theta^{forearm}|) > 0.2$
- $\min(|\cos \theta^{upperarm}|) > 0.2$

GOAL_KICK For `GOAL_KICK` we filter out sequences with the arm never raised up as well as the arm raising too high, implemented with conditions

- $\min(|\cos \theta^{forearm}|) > 0.2$
- $\min(|\cos \theta^{upperarm}|) > 0.5$
- $\max(d_z^{elbow} - d_z^{shoulder}) > 0.1$

CORNER_KICK For `CORNER_KICK` we filter out inappropriate sequences with the arm not raised over the shoulder, given by the condition

- $\max(d_z^{elbow} - d_z^{shoulder}) < 0$

INDIRECT_FREE_KICK For the `INDIRECT_FREE_KICK` we filter out unqualified sequences with the arm not raising high enough, given by the condition

- $\max(d_z^{elbow} - d_z^{shoulder}) < 0.2$

After implementing all the aforementioned collecting and cleaning strategies, we converted frame data into a substantial number of 4-second referee skeleton sequences. The statistics for these sequences are shown in Table B.1.

	# 4s Sequences
THROW_IN_HOME	1062
THROW_IN_AWAY	1116
GOAL_KICK	1676
CORNER_KICK	937
INDIRECT_FREE_KICK	2058

Table B.1: Labeled skeleton sequences collected by converting frame data

B.2.3 Inlier Detection with Pre-training Loss

Collecting skeleton sequences for `NO_DECISION` poses a unique challenge. Simply extracting sequences around `NO_DECISION` frames from assistant referees is suboptimal, as it does not necessarily mean the main referee is inactive. Relying on arm angles to ensure they are not raised could also misrepresent `NO_DECISION` instances since referees often swing their arms while running. To more effectively collect representative skeleton sequences for `NO_DECISION`, we suggest utilizing the pre-training loss from our sequence-level pre-trained model to identify inlier sequences, which are likely to be `NO_DECISION` actions.

As our pre-trained model calculates pre-training loss only on masked joints, we propose masking all arm joints—namely both elbows and both wrists—from the input skeleton sequence, and then calculating the pre-training loss specifically for these masked arm joints. Given that most referee signals are conveyed through arm movements and such signals are relatively rare compared to non-signal sequences in football games, the pre-trained model should be more likely to predict the arm joints at positions where there are no signals. Consequently, sequences that contain referee signals are likely to have a higher pre-training loss, whereas sequences with a smaller pre-training loss are more common sequences, known as inliers, and thus more likely to represent `NO_DECISION`. We utilize the model pre-trained with a multi-task pre-training objective as introduced in Section 4.2.1.6 and employ the `data2vec` loss as the metric for detecting inliers. We also experimented with motion prediction loss, but found that `data2vec` loss yields more reliable results.

Based on initial observations, we established a threshold of 0.022 for the `data2vec` loss on masked arm joints. To maintain consistency with the sequences collected previously, we segment unlabeled data into 4-second sequences and use the pre-trained model to compute the pre-training loss. Sequences with a loss below the threshold are identified as inliers and labeled as `NO_DECISION`. These labeled sequences undergo further manual verification to ensure data quality. Eventually, we collected 2,097 4-second `NO_DECISION` sequences.

B.2.4 Binary Whistle Classification Model

Another significant challenge is collecting skeleton sequences with label `DIRECT_FREE_KICK`. Given that `DIRECT_FREE_KICK` actions are typically preceded by a whistle-blow, identifying whistles could be beneficial for finding

these actions. Fortunately, our data source already includes annotations for whistle events, which allows us to train a binary classification model to identify sequences with whistle blows from unlabeled data. After gathering potential whistle events, we apply some rules to filter out improbable sequences and manually verify them to confirm whether they correspond to a `DIRECT_FREE_KICK`.

Training a Binary Classification Model for Whistle vs. Non-Whistle Detection To train a binary classification model for detecting whistle events, we first collected a sequence classification dataset, shown in Table B.2. We utilized an ST-GCN model for this binary whistle classification, equipped with 9 hidden layers with hidden sizes (128, 128, 128, 256, 256, 256, 512, 512, 512). The model was trained for 20 epochs using an AdamW optimizer with a weight decay of $5e-4$. The learning rate follows an inverse square root decay after a 2,000-step linear warmup to $3e-4$. The batch size is set at 64. The trained binary whistle model achieves an accuracy of 95.62% on the validation set and 96.87% on the test set.

	Train	Valid	Test
WHISTLE	607	114	122
NON_WHISTLE	2448	479	453
Total	3055	593	575

Table B.2: Dataset collected for training binary classification model for whistle events

However, in football games, referees blow the whistle for various reasons, but only a fraction of these instances are `DIRECT_FREE_KICK` signals. Given that we already have plenty of data for actions that involve assistant referee actions such as `THROW_IN`, `GOAL_KICK`, or `CORNER_KICK`, we aim to exclude these from our current focus. To this end, we plan to use the frame-level assistant referee recognition model derived in Section 3.3.3.1 to identify sequences where a whistle is blown but no assistant referee action is recorded. By eliminating sequences in these action types, we enhance the likelihood that the remaining sequences belong to `DIRECT_FREE_KICK`.

Specifically, because the whistle model doesn't work perfectly - and sometimes misinterprets actions like scratching or wiping sweat as the whistle blows — we set a very high confidence threshold of 0.9999 to ensure the data quality. We segment all available unlabeled data into multiple 2-second sequences and use the whistle model to assess the likelihood of each

involving a whistle blow. Only sequences with a probability exceeding 0.9999 are retained. Subsequently, we examine a 10-second window around each sequence and employ the frame-level assistant referee model to predict actions for both assistant referees. With a frame rate of 25 Hz, this generates 500 labels. If fewer than 20 frames of the 500 are labeled anything other than `NO_DECISION`, we confirm the 2-second sequence as an action involving a whistle blow without assistant referee actions. To be consistent with the previously collected data, we extend each 2-second sequence by one second on both ends to make it 4 seconds.

Note that this strategy only identifies sequences with a higher likelihood of being a `DIRECT_FREE_KICK`. To definitively include these sequences in the classification dataset, we extract corresponding video clips for manual review to confirm or reject each sequence. Ultimately, this process resulted in the collection of 156 `DIRECT_FREE_KICK` sequences.

B.2.5 Combination of Rules

Since our assistant referee action recognition does not cover signals for `GOAL`, it is crucial to include them for the main referee to ensure that referees' goal signals are not overlooked. The signal for a goal by the main referee involves pointing an arm towards the center mark. However, identifying these signals poses a challenge because the action can be easily confused with a throw-in or direct free kick. To address this, we propose using a set of rules grounded in common football knowledge to detect referee actions for `GOAL`.

In football games, when a goal is scored, the conceding team is going to kick off. Leveraging this, we first detect all kick-offs by checking for a stationary ball at the center mark. We then trace back to locate the main referee's signals preceding the kick-off. Using the arm angle indicators described in Section B.2.2, we set conditions to identify the nearest instance of the referee's action when the arm is raised.

Formally, the strategy to find `GOAL` actions involves the following steps:

- 1 Find kick-offs by identifying a stationary ball at the center mark:
 - The ball's speed is less than 0.5 m/s for over one second, and it stays within a 0.5m radius of the center mark
- 2 Trace back the 2 minutes referee sequences prior to the kick-off, segmenting into 4-second sequences
- 3 Retaining only those sequences where the ball is near the goal area:

- The absolute x value of the ball position is greater than 36
- 4 Applying the arm angle and distance indicators as introduced in Section B.2.2 to only keep sequences that the arm is raised:

- $\min(|\cos \theta^{forearm}|) < 0.2$
- $\min(|\cos \theta^{upperarm}|) < 0.7$
- $\max(d_z^{elbow} - d_z^{shoulder}) > -0.15$

5 Manually reviewing and confirming the filtered sequences

Eventually, using this rule-based strategy, we collected 513 GOAL sequences, each 4 seconds in length.

B.2.6 Collecting OTHER_DECISION

In addition to the action types for which we have already collected data, there are more various referee signals on the pitch. Some of these actions are standardized and outlined in the Laws of the Game, such as issuing yellow/red cards and signaling advantage. However, due to the sparsity of data, it is impractical to allocate a separate class for each of these actions. Other actions, more similar to body language and not standardized, such as signaling no foul, asking to hold on, or calling the doctor, should be distinguished from NO_DECISION. Sometimes, the referee's actions are not clearly interpretable, yet it is evident they signify more than nothing. For practicality and simplicity, we propose merging all these into a single class, OTHER_DECISION. Clearly, there is no fixed pattern associated with the OTHER_DECISION class. The purpose of establishing this class is to differentiate these signals from the main action types we are focused on without categorizing them as NO_DECISION.

Specifically, the OTHER_DECISION class includes following action types: SANCTION, ADVANTAGE, NO_FOUL, HOLD_ON, CALL_DOCTOR, HURRY_UP, SET_PIECE_ALLOWED, SET_PIECE_MISPLACED, SUBSTITUTION, VAR_REVIEW, DROP_BALL, INTERRUPT. Some skeleton sequences are labeled directly as OTHER_DECISION when the referee's actions do not fit into any of the previously defined classes.

To collect skeleton sequences for OTHER_DECISION, we utilize preliminarily fine-tuned models for sequence action recognition, trained with a partially collected dataset. Given the lack of OTHER_DECISION data within the dataset, these actions frequently feature confusion among several classes. Leveraging this, we use the model to predict classification probabilities

for segmented 4-second unlabeled sequences and select those that exhibit uncertainty between classes. Specifically, we retain sequences where the probability of `NO_DECISION` ranges from 0.1 to 0.9. These sequences are then proceeded to manual confirmation. This approach not only aids in gathering `OTHER_DECISION` data but also helps the data collection for previous classes. Eventually, we have collected 271 4-second sequences for `OTHER_DECISION`, as detailed in Table B.3.

	# 4s Sequence
<code>OTHER_DECISION</code>	112
<code>SANCTION</code>	21
<code>ADVANTAGE</code>	17
<code>NO_FOUL</code>	12
<code>HOLD_ON</code>	13
<code>CALL_DOCTOR</code>	5
<code>HURRY_UP</code>	8
<code>SET_PIECE_ALLOWED</code>	35
<code>SET_PIECE_MISPLACED</code>	17
<code>SUBSTITUTION</code>	22
<code>VAR_REVIEW</code>	1
<code>DROP_BALL</code>	1
<code>INTERRUPT</code>	7
Total	271

Table B.3: Composition of `OTHER_DECISION` class

B.2.7 Expansion for the Sequence-labeling Task

When trained with the sequence-labeling task, as discussed in Section 4.2.3.4, the model becomes less sensitive to the sequence length. This allows more contextual information to be incorporated to enhance performance. We propose extending the existing 4-second sequences to 8 seconds for this purpose. Given the supervision from the action non-action binary classification model, the risk of introducing noisy data through this expansion is minimal because the frames before the action starts or after the action ends will be labeled `NO_DECISION` automatically by the model supervision. For all sequence types except `DIRECT_FREE_KICK`, we extend the sequence by adding 4 seconds before the existing sequence, resulting in 8-second sequences. For `DIRECT_FREE_KICK` sequences, we extend 2 seconds on both sides of the sequence, also resulting in 8-second sequences. The reason for

this difference is that extending sequences after it could introduce noise when there is an extra action different from the sequence's label, whereas it is less probable that other signals will immediately follow a `DIRECT_FREE_KICK` signal.

