Skeleton-Based Modeling in Badminton

Unlocking Insights for Stroke Recognition and Forecasting

by

Magnus Andreas Petersen Ibh

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY
COMPUTER SCIENCE
IT UNIVERSITY OF COPENHAGEN

OCTOBER, 2024

Abstract

This thesis uses skeleton poses as the primary modality for analyzing badminton video sequences. The topic is approached through three central computer vision tasks: Action recognition, action forecasting, and 3D shuttle reconstruction.
One part of the research explores and develops models that best capture informative motion representations from skeleton sequences to infer the performed stroke. Additional features such as shuttlecock position and player court location are integrated to improve recognition performance.
Another focus is on identifying limitations in stroke recognition, such as limited and inconsistent annotation conventions and type imbalance, and exploring potential solutions. Pretraining, especially self-supervised learning, is explored. The models are pretrained using masked autoencoders, reconstructing parts of the skeleton sequences hidden in the input. The findings show that these approaches improve model performance on downstream tasks like stroke recognition.

Achieving the best model performance is not the only goal of the thesis. Part of the research tries to identify which elements of a player's stroke motion carry the most descriptive information on a particular action. Through model inspection, the different phases of the stroke motion and various modalities are examined using ablation and qualitative attention studies to determine which offers the most relevant information to the model. The results are compared to the human perspective of analysts and coaches to gauge how the findings could benefit coaches and players in match preparation.
Expanding on this, the thesis investigates the model forecasting capabilities for predicting the next strokes. Usually, in sequence modeling, the next stroke in a rally would be predicted based on the sequence of stroke exchanges up to that point. Here, instead, a model architecture is proposed that learns a stroke representation from the player's skeleton motion, identity, and shuttle position to condition the prediction probability of the next stroke. The model design reflects the turn-based nature of badminton to capture a basic understanding of the game to make informed predictions.

Using 3D information can extract valuable physical and shot statistics while eliminat-

ing ambiguities found in 2D image representations. However, the limited availability of 3D badminton data inhibits the development of effective reconstruction models. A physics-based model trained on synthetic 3D shuttlecock trajectories is proposed to overcome this challenge. The developed model, TrajTrans, predicts the initial 3D conditions based on 2D image projections. The results generalize well to real data by implementing shot filtering criteria of the synthetic data that ensure realistic trajectories.

Ultimately, the findings contribute to the field of sports analytics by providing foundational knowledge and guidelines that can advance the development of future tools for analysts and coaches.

Resume

Denne afhandling fokuserer på at anvende skelet-positurer som den primære modalitet til at analysere badminton-videosekvenser ved at adressere tre centrale computer vision-opgaver: Slag-genkendelse, slag-forudsigelse og 3D rekonstruktion af data fra 2D.
En del af forskningen udforsker og udvikler modeller, der bedst indfanger informative bevægelses-repræsentationer fra skelet-sekvenser til at afgøre, hvilket slag der udføres. Yderligere input, som fjerboldens position og spillerens placering på banen, integreres for at forbedre modellerne og deres genkendelses-performance.
Et andet fokus er at identificere begrænsninger indenfor slag-genkendelse, herunder begrænset og inkonsekvent annoteringspraksis samt ubalance mellem slagtyper, og at undersøge mulige løsningsstrategier. Fortræning, især selv-superviseret fortræning, af modeller undersøges og resultaterne viser, at disse tilgange forbedrer modelpræstationen på efterfølgende opgaver, som slag-genkendelse.

At opnå den bedst mulige modelpræstation er dog ikke det eneste mål med afhandlingen. En del af forskningen søger at identificere, hvilke elementer af spillerens slagbevægelse der rummer den mest beskrivende information om en given handling. Gennem modelinspektion undersøges de forskellige faser af slagbevægelsen samt forskellige modaliteter, for at forstå, hvilke der giver den mest relevante information til modellen. Resultaterne sammenlignes med analytikeres og træneres menneskelige perspektiv for at vurdere, hvordan disse indsigter kan gavne trænere og spillere i deres kampforberedelse.
Et andet centralt punkt i afhandlingen, er udviklingen af en model der kan forudsige fremtidige slag i en badminton duel. Normalt i sekvens-modellering vil man forsøge at forudsige det næste slag i en duel kun baseret baseret på sekvensen af slag i en duel frem til det pågældende tidspunkt. Her foreslås i stedet en modelarkitektur, som lærer en slag-repræsentation ud fra spillerens skeletbevægelse, identitet og fjerboldens banekurve, og medregner det i forudsigelsen af det næste slag i duellen.

Anvendelsen af 3D-information fra både spiller-skeletter og fjerboldbevægelse giver mulighed for at udlede værdifulde fysiske og slagrelaterede statistikker og kan potentielt udrede tvetydigheder, der findes i 2D billedrepræsentationer. Dog vanskel-

liggør den begrænsede tilgængelighed af 3D-data for badminton udviklingen af effektive rekonstruktionsmodeller. Som en mulig løsning til denne udfordring foreslås en fysikbaseret model, der er trænet på syntetisk genererede 3D fjerbold-data. Modellen forudsiger de indledende 3D startbetingelser baseret på 2D video-sekvenser. Den udviklede model TrajTrans, trænet på syntetiske data, viser lovende resultater anvendt på badminton-data fra virkelige kampe.

Samlet set bidrager resultaterne i afhandlingen til machine learning i sportsanalyse ved at etablere et fundament af viden og retningslinjer, der kan fremme udviklingen af værktøjer til analytikere og trænere i fremtiden.

Contents

Abstract **3**

Resume **4**

Abbrivations **8**

Acknowledgments **11**

1 Introduction **12**
- 1.1 Motivation . . . 12
- 1.2 Thesis Goals . . . 13
- 1.3 Thesis overview . . . 14
- 1.4 Badminton . . . 19

2 Neural components **24**
- 2.1 Graph Convolutional Networks . . . 24
- 2.2 Temporal Convolutional Networks . . . 32
- 2.3 Transformer . . . 35
- 2.4 Chapter Conclusion . . . 41

3 Human Pose Estimation **42**
- 3.1 Introduction . . . 42
- 3.2 Methods of Human Pose Estimation . . . 43
- 3.3 Quality of Pose Estimation on Badminton videos . . . 49
- 3.4 Chapter Conclusion . . . 53

4 Data pipeline & Datasets **54**
- 4.1 Homogenous Coordinates and Perspective Transfomation . . . 55
- 4.2 Data Features . . . 57
- 4.3 Datasets . . . 63
- 4.4 Chapter Conclusion . . . 71

5 Skeleton-based Stroke Recognition **76**
5.1 Raw Video Frames vs. Skeleton Features for Stroke Recognition in Badminton . . . 77
5.2 Related Work . . . 80
5.3 TemPose: A Multimodal Factorized Transformer for Fine-Grained Stroke Recognition in Badminton . . . 82
5.4 Experiments . . . 88
5.5 Skeleton-based Pretraining Methods . . . 101
5.6 Chapter Conclusion . . . 107

6 Stroke Forecasting **109**
6.1 Related Work . . . 111
6.2 Forecating task formulation . . . 114
6.3 RallyTemPose . . . 115
6.4 Experiments . . . 121
6.5 Discussion . . . 125
6.6 Chapter Conclusion . . . 131

7 3D Reconstruction of Shuttlecock Trajectories **133**
7.1 Camera models . . . 135
7.2 3D shuttle-trajectory estimation with Physics-based modeling . . . 139
7.3 Shuttle Experiments . . . 153
7.4 Discussion . . . 165
7.5 Chapter Conclusion . . . 167

8 Conclusion **168**
8.1 Main Takeaways . . . 168
8.2 Limitations & Short-comings . . . 169
8.3 Future Prospects . . . 170
8.4 Key Insights for Collaboration . . . 171
8.5 Overall Message . . . 172

References **185**

Appendix Paper 1: TemPose: A new approach skeleton-based action recognition **186**

Appendix Paper 2: A stroke of Genius: Predicting the next move in badminton **197**

Appendix Paper 3: SynthNet: 3D trajectory reconstruction from synthetic training data **208**

Abbrivations

HPE Human Pose Estimation

MLP Multi-Layer Perceptron (used synonymously with FCNN and FC)

GCN Graph Convolutional Network

TCN Temporal Convolutional Network

MHSA Multi-head Self-Attention

PE Positional Encoding

RNN Recurrent Neural Network

LSTM Long Short-Term Memory

GRU Gate Recurrent Unit

CNN Convolutional Neural Network

ReLU Rectified Linear Unit

GELU Gaussian Error Linear Unit

MSE Mean Square Error

PCK Percentage of Correct Keypoints

OKS Object Keypoint Similarity

TP True Positives

FP False Positives

FN False Negatives

PoT Pose Transformer

RGB Red-Green-Blue (Synonym for color video footage)

ViViT Video Vision Transformer

ViT Vision Transformer

ST SpatioTemporal

TS temporospatial

NLP Natural Language Processing

BadPL Badminton Placement

BadOL Badminton Olympics

Sset Shuttleset

Sset22 Shuttleset 22

STB Spatial Transformer Block

TTB Temporal Transformer Block

Acknowledgments

First and foremost, I would like to express my heartfelt gratitude to my girlfriend, Cécile, and family, especially my parents, for their unwavering support and encouragement throughout my academic journey. Their belief in me has been a constant source of motivation and strength.
I would also like to extend my deepest thanks to my supervisor, Dan Witzner Hansen, for his invaluable guidance and insights, which have significantly shaped this thesis. I sincerely thank Stella Grasshof and Pascal Madeleine for their support and constructive feedback, which was instrumental in refining my work. I am grateful to my fellow PhD colleagues, whose camaraderie and intellectual discussions have made this journey both enriching and enjoyable.
Furthermore, I wish to express my appreciation to Team Danmark and Badminton Danmark for their collaboration and the insights they provided regarding the analysis of professional badminton. Their expertise and willingness to share knowledge have been crucial to the success of this thesis.
Additionally, I am grateful for the financial support from Novo Nordisk Fonden, which partially funded this project as part of TeamSPORTek.
Thank you to all those who have supported and inspired me along the way.

Copenhagen, Dec. 2024

Magnus Ibh

Chapter 1

Introduction

Technological advances have led to the development of analysis tools to refine athletes' performance, enhance coaching strategies, and enrich viewers' experiences. In this context, human pose estimation and action recognition technologies have emerged as viable fields, especially in sports such as badminton, where minor differences in posture and technique can significantly impact performance outcomes. This thesis explores the application of skeleton-based action recognition and forecasting for stroke analysis in badminton. In this context, the term "skeleton" refers to a human body model consisting of connected joints indicating key points on the human body. In this capacity, the potential of human skeleton data to provide meaningful representations of athletic motion without the distractions presented by lighting, backgrounds, and other redundant information is examined.
Despite the wealth of possibilities computer vision offers in sports analytics, the scarcity of annotated sports data poses a significant challenge. Annotated datasets are crucial for training accurate and reliable models; however, they are often limited, especially for sports that do not enjoy as wide coverage as others. This limitation inhibits the development of models that capture the essence of sport-specific movements and predict future actions with high precision.

1.1 Motivation

The motivation for focusing on skeleton data arises from its potential to condense detailed athlete movements into a simplified yet informative format. Unlike traditional

video data, which require substantial processing to filter out irrelevant background information, skeleton data provide a distilled representation of motion, focusing solely on the athletes' poses. This abstraction facilitates a more focused analysis of movements and techniques and significantly reduces the computational resources required for processing, making it an efficient choice for real-time applications.
The sparse nature of annotated sports data necessitates innovative approaches to model training. This thesis explores the use of skeleton/joint data of both players and the shuttle fused with self-supervised learning techniques to utilize the vast amounts of unlabeled data, thereby circumventing the limitations of dataset sparsity. This approach enhances the model's understanding of complex motions and broadens its applicability across different athletes and conditions without requiring extensive annotated datasets. This thesis aims to address the challenges presented by sparse data using skeleton data and deep learning techniques. The focus is badminton, a sport in which precision and technique play a critical role.
Deep learning applications in sports analytics have increased significantly in popularity. Despite this growth, many specific use cases for athletes and coaches have yet to be established. This thesis addresses this gap by focusing on the practical applications of deep learning for badminton, emphasizing the analysis of skeleton-extracted motions during matches.

1.2 Thesis Goals

Stroke Recognition This research aims to investigate specific downstream tasks that utilize skeleton-based data to capture essential properties of badminton matches. The thesis employs machine learning architectures tailored for stroke recognition, conducting several experiments to determine the most effective input modalities and features for accurate stroke recognition. It will also explore the temporal dynamics of stroke sequences and tackle the challenge posed by the scarcity of annotated stroke data. The approach includes exploring a self-supervised method for pretraining, leveraging the abundant unannotated badminton video material available to improve the robustness and generalizability of recognition algorithms.

Stroke Forecasting The thesis extends beyond basic recognition tasks to explore in-depth the analytical capabilities of deep learning models. Although automatic an-

notation of video events is beneficial, it often does not interact directly with real-time match analysis. The thesis progresses into action forecasting, using previous stroke data to predict an opponent's next actions. This requires the integration of motion representation and sequence analysis to inform predictions about future strokes, necessitating that the model comprehends underlying tactical and positional dynamics.

3D Data Reconstruction Another aspect of this thesis investigates the potential for reconstructing 3D information from 2D match footage. The transition from image to real-world data significantly enhances the information inferred from the extracted data, such as player movements and shuttle trajectories, facilitating a more comprehensive analysis of matches. Comparing the benefits of 3D over traditional 2D data, this research seeks to show new insights and improve the predictive capabilities of sports analytics models. The work in this thesis is limited to 3D reconstruction of the shuttle, but in future work, extracting accurate 3D player motion should be of high priority.

Main Takeaway The findings of this thesis are intended to provide a current overview of potentially useful applications of neural machine learning methods in badminton. Furthermore, the proposed methods highlighted issues and challenges that are supposed to provide direction for the future development of concrete tools and applications for analysts, coaches, and athletes.

1.3 Thesis overview

The following section summarizes the main focus of each chapter of the thesis:

- **Neural Components:** In this chapter, relevant fundamental components for neural network models are introduced. The chapter covers the following model types: Temporal convolution Networks (TCN), Graph Convolutional Networks (GCN), and Transformer models.
- **Human Pose Estimation:** This chapter introduces the core focus of the thesis, human keypoint/skeleton data that is used as the main feature for many computer vision tasks in the remaining part of the thesis. Finally, this section

uses a self-annotated badminton dataset to evaluate different pre-training human pose estimation (HPE) models.

- **Data Pipeline and Datasets:** This chapter defines the main features and modalities used throughout the model. The modalities refer to. Skeleton joint data (S) and skeleton bone data (B). Shuttlecock coordinates (U) and player court position (G). Followed by a description of the procedure and methods to extract skeleton (pose) and 2D shuttle trajectories from match videos, along with preprocessing to refine for use as model input features. Last, the chapter introduces the annotated badminton datasets used for the experiments in the thesis.

- **Stroke Recognition:** This chapter is comprised of the main contributions from the Paper 1 *"TemPose: a new skeleton-based transformer model designed for fine-grained motion recognition in badminton"*. It includes prior related work on skeleton-based action recognition in sports and the TemPose model architecture, along with results on fine-grained badminton datasets. Additional experiments into more complex transformer architectures and self-supervised pre-training methods are also included in the chapter.

- **Stroke Forecasting:** In this chapter, skeleton condition forecasting of badminton strokes is explored based on the main contribution of the Paper 2 *"A Stroke of Genius: Predicting the Next Move in Badminton"* an extension of the TemPose model RallyTemPose is proposed badminton tailored stroke forecasting. Furthermore, application for the forecasting model is explored, such as player similarity and player style comparisons. One addition to the paper includes an experiment redefining the prediction focus from the player's entire stroke motion to the motion during the shuttle flight.

- **3D Reconstruction of Shuttlecock Trajectories:** This chapter builds upon our Paper 3 *"SynthNet: 3D trajectory reconstruction from synthetic training data"*, which proposes a synthetic model training method shuttlecock 3D reconstruction. In the paper, the method is applied to tennis matches, but in this chapter, the method is improved and repurposed for broadcasted badminton matches. Additional content includes a comparison of 3D data and 2D data for badminton shot recognition.

- **Discussion and Conclusion** Discussion of the current landscape of deep-learning applications for sports. Addressing limitations and future work for successfully integrating deep-learning-based analysis in Badminton and sport in general.

The raw data used throughout the thesis are segmented – or annotated – video clips of broadcasted badminton matches. The actual segmentation of raw badminton videos into segmented strokes/actions is not part of the work in this thesis, which entirely uses available annotated data. However, during the course of the project, several student projects have looked into this particular area, and while not in this thesis, it is certainly an essential part of badminton-related deep-learning tasks, which could be integrated as pre-processing for the models developed in this thesis and similar.

1.3.1 Reuse of Published Work

This thesis incorporates material from the following previously published works by the author:

- Paper 1
  M. Ibh, S. Grasshof, D. Witzner, and P. Madeleine, "TemPose: a new skeleton-based transformer model designed for fine-grained motion recognition in badminton," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 5199–5208, 2023. © 2023 IEEE. Reprinted, with permission, from [57].

- Paper 2
  M. Ibh, S. Graßhof, and D. Witzner, "A Stroke of Genius: Predicting the Next Move in Badminton," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3376–3385, June 2024. © 2024 IEEE. Reprinted, with permission, from [56].

- Paper 3
  M. Holck Ertner, S. S. Konglevoll, M. Ibh, and S. Graßhof, "SynthNet: Leveraging Synthetic Data for 3D Trajectory Estimation from Monocular Video," *Proceedings of the 7th ACM International Workshop on Multimedia Content Analysis in Sports (MMSports '24)*, pp. 51–58, Melbourne VIC, Australia, 2024. © 2024 ACM. Reprinted, with permission, from [38].

1.3.2 Badminton Danmark & Team Danmark

Badminton Danmark and Team Danmark deserve gratitude and acknowledgment for their participation, including data collection and expert insights into badminton. Most badminton domain knowledge presented in this thesis is obtained through dialog, conversation, and meetings with Badminton Danmark.

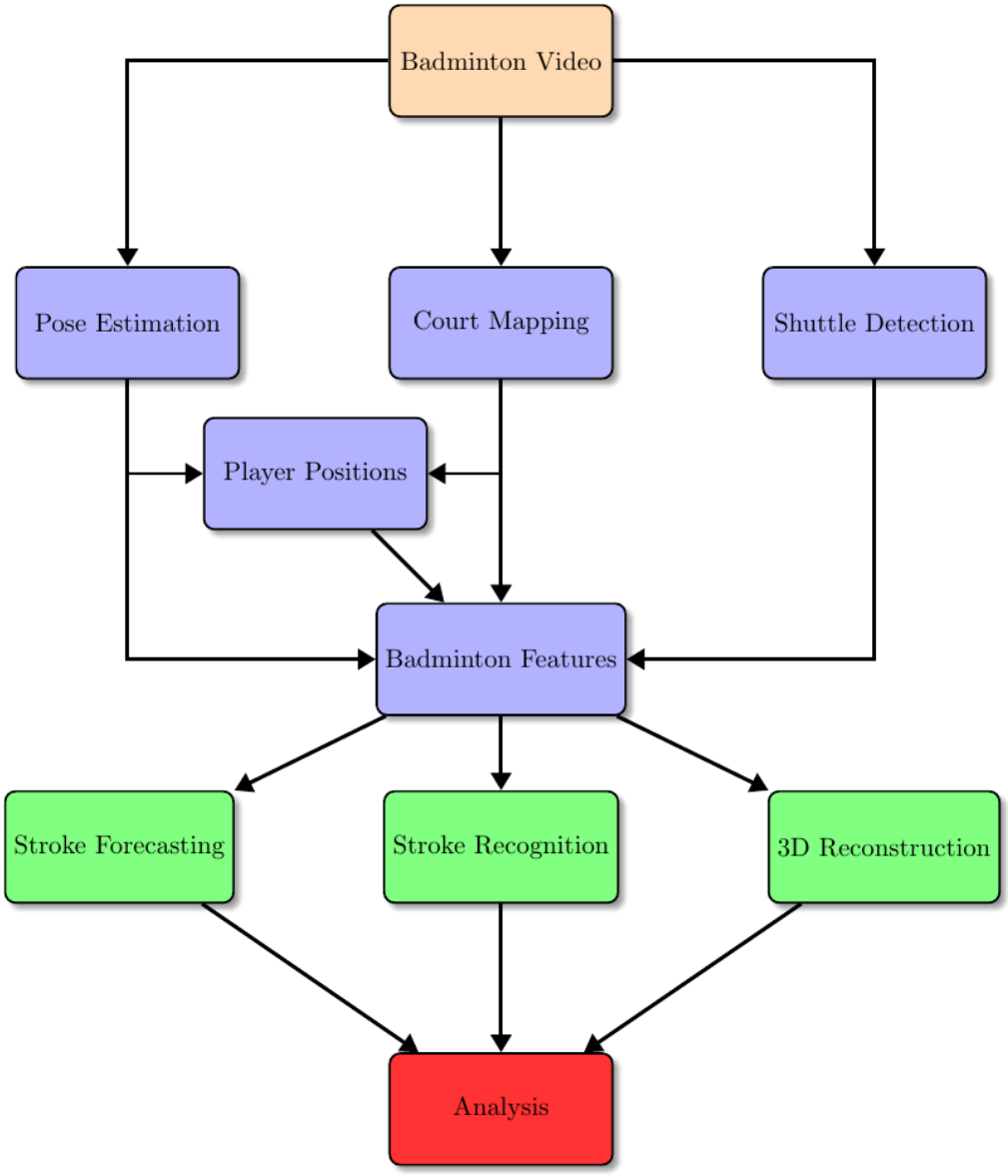


Figure 1.1: Flowchart that shows the thesis structure. The badminton videos (orange) are the raw material used in the thesis. Pose Estimation, Shuttle Detection, Player Position, and Court Mapping (blue) comprise the extracted and processed features for deep-learning-based badminton tasks. Stroke Recognition, Stroke Forecasting, and 3D Reconstruction (green) are computer vision research areas tackled in the thesis for application in badminton. The experimental results in each research area are discussed, emphasizing their implication for Analysis (red) in badminton.

1.4 Badminton

Badminton, a popular racket sport, can be played in singles (one-on-one) or doubles (two-on-two). The ”ball” in badminton is a so-called shuttlecock (or just shuttle) made from a rounded peace of cork with a conical arrangement of feathers providing just the right amount of air resistance to make the shuttle ”playable”. The main objective is to score points by hitting a shuttle with a racket so that it passes over a net and lands within the opponent's court area without being successfully returned. The racket is lightweight and stiff, typically made from carbon composite material and strung with thin high-tension polymer strings. This construction reflects the comparatively low mass of the shuttle (e.g. compared to tennis or squash balls) and enables the player to impact and launch the shuttle with significant speed when so desired. In Figure 1.2, a picture of a typical racket and shuttlecock is shown.
Each rally in the game starts with the serve, which must be hit underhand with the racket pointing downwards from horizontal and below the waist of the server. The shuttle must be hit diagonally across the net into the designated service area. Points are scored when the shuttlecock lands in the opponent's court or if the opponent commits a fault, such as hitting the shuttlecock into the net or out of bounds.
A standard badminton match is typically structured as a best-of-three sets game, with each set played to 21 points. A player or team must lead by at least two points to win the set; if the score reaches 20-20, play continues until one side leads by two points. If the score reaches 29-29, the next point determines the winner of that set.

1.4.1 Court Dimensions

The dimensions of a badminton court vary depending on whether it's a singles or doubles match:

- The court is rectangular and divided into halves by a net. Courts are marked for both singles and doubles play. The doubles court is wider.
- The full width of the court is 6.1 meters, and in singles, this width is reduced to 5.18 meters.
- The full length of the court is 13.4 meters.
- The net is 1.55 meters high at the edges and 1.524 meters high at the center.

(a) Racket, the tool players use to hit the shuttlecock over the net.

(b) Shuttlecock or shuttle, the object players try to hit over the net.

Figure 1.2: The racket and the shuttlecock, the two central objects used in badminton

- The service courts are marked by a center line dividing the width of the court, by a short service line at a distance of 1.98 meters from the net, and by the outer side and back boundaries. The service court is also marked by a long service line 0.76 meters from the back boundary.

The badminton court – with dimensions – is depicted in Figure 1.3.

1.4.2 Stroke definition

In badminton, a ”stroke” refers to the act of hitting the shuttlecock with the racket, which is the fundamental element of the game. Specifically, the stroke refers to the motion a player performs with their body and, by extension, the racket to hit a shuttle across the net. The shuttle motion produced by the player's stroke is commonly referred to as a shot. The terms stroke and shot can sometimes be intertwined, but in this work, the convention is applied: A stroke is centered around the player's motion, while a shot centers around the shuttlecock motion. Detailed biomechanics aspects are considered when teaching proper technique to badminton players to maximize speed, precision, and efficiency when hitting the shuttle. This work will deal with the image-based skeleton motion of the players. Thus, detailed biomechanical considerations will not be directly extracted or analyzed. However, the deep-learning models may still implicitly learn them. Depending on the player motion and racket orientation, many different shots can be produced, ranging from an offensive smash and a neutral clear to a defensive block. The basic categories a stroke can be divided into

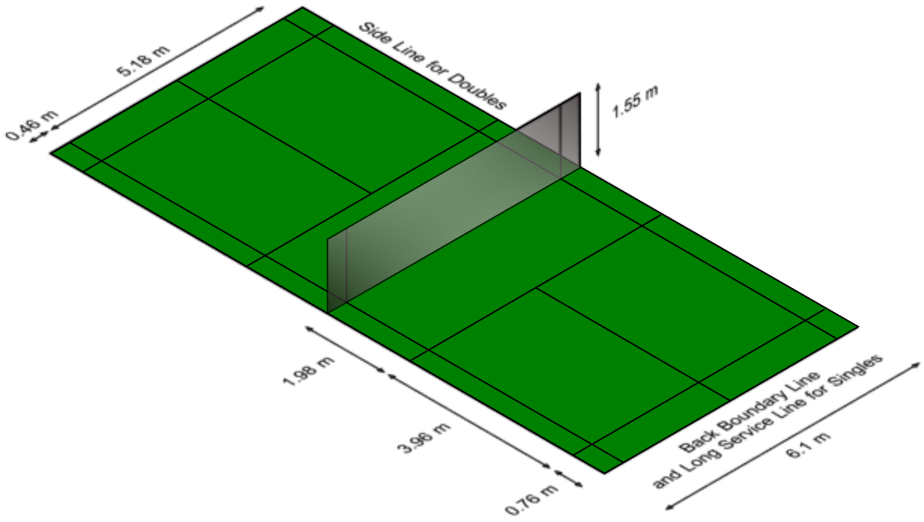


Figure 1.3: The badminton court with respective dimensions.

are listed in Table 1.1. The placement of each stroke holds relevance since a specific stroke might have a different value depending on the placement.
This paper explores two primary ways to define and analyze badminton strokes, focusing on different aspects relevant to the sport's action recognition and anticipation strategies.
The first definition, called Split-M, emphasizes the stroke motion itself, which is divided into several phases: preparation, forward acceleration, contact with the shuttle, and follow-through and recovery. The different phases are shown in Figure 1.4. This approach examines the player's movements to analyze racket speed and shuttle trajectory during the hit. By understanding these phases, we can gain insight into the player-specific motions crucial for recognizing and categorizing different strokes.
The second definition, Split-T, focuses on the shot, i.e., the shuttle's trajectory as influenced by the stroke. This perspective is particularly useful for predicting and anticipating in-game actions, as it provides a pragmatic view of how matches are analyzed in real-time. Players use this information to predict their opponent's next moves, allowing them to position themselves more effectively on the court.
Both definitions overlap and contribute to our understanding of badminton dynamics, but also serve distinct purposes. Split-M is more about the technical execution

[1]Stroke types are based on https://development.bwfbadminton.com/coaches/level-1

Table 1.1: Summary of Different Types of Badminton Strokes, according to the BWF coaches manual [1]

Stroke	Description
Serve	Start of the rally. Two serves are the most commonly used. A forehand serve hits the shuttle too high to the opponent's backcourt. A short backhand serve chipped low over the net, not allowing the opponent to attack.
Clear	A high, deep shot hit overhead from the backcourt directed to the back of the opponent's court. It has a long preparation time and serves as a transport stroke.
Drop Shot	A soft, precise shot that falls just over the net, forcing the opponent to move forward. The drop shot is played from the backcourt.
Smash	A powerful, downward shot aimed at the opponent's court, intended to end the rally.
Drive	A fast, flat shot that travels parallel to the floor, usually aimed at the opponent's mid court to maintain pressure.
Net Shot	A delicate shot played close to the net, often with spin to make it difficult for the opponent to return. Includes variations like the tumbling or brush net shot.
Lift	A defensive shot that sends the shuttle high and deep into the opponent's court, typically used when under pressure.
Push	A controlled, gentle shot with minimal backswing, pushing the shuttle into the opponent's mid or backcourt to maintain or change the pace of the rally.
Block	A defensive stroke used to return a smash, gently blocking the shuttle back over the net to neutralize the attack.

of strokes, ideal for training and technique refinement. In contrast, Split-T is geared toward strategic gameplay, helping players and coaches to simulate and respond to competitive scenarios.

With this basic introduction to badminton concluded, the focus will now be shifted to the fundamental neural network components constituting the basis of the more advanced network architectures presented in the thesis.

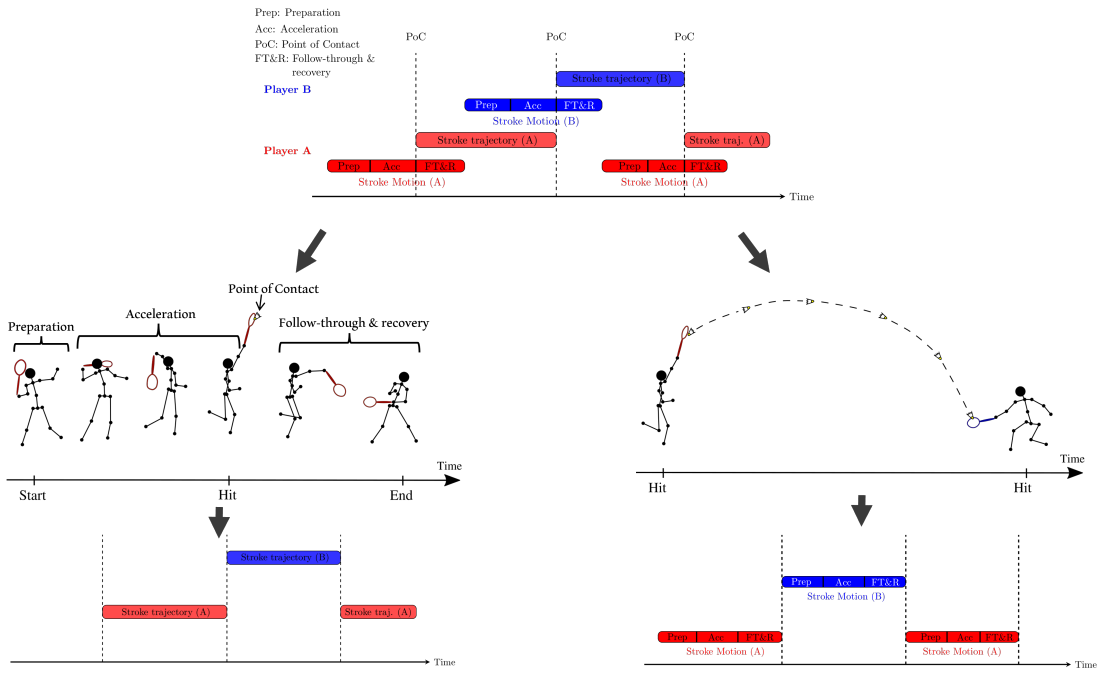


Figure 1.4: Badminton strokes can be divided into two different groups, motion-focused (left) or trajectory-focused (right).

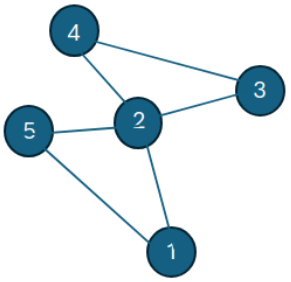
Chapter 2

Neural components

This chapter will cover the necessary but fundamental concepts in neural learning components present throughout the thesis. This section will introduce the Graph Convolutional Network (GCN), Temporal Convolutional Network (TCN) block, and transformer block.
Graph Convolutional Networks (GCNs) and Temporal Convolutional Networks (TCNs) are specialized neural network architectures for graph-structured and sequential data using specialized convolutional operations.

2.1 Graph Convolutional Networks

Graph Convolutional Networks (GCNs), first proposed in [63], are neural networks designed to operate on graph-structured data by aggregating information from neighboring nodes akin to convolutional networks. GCNs extend the concept of convolution from regular grids (such as images) to graphs, making them effective for tasks involving relational data, such as social networks, and, notably in this context, modeling of human skeleton data.
An undirected graph $G = (V, E)$ consists of a finite set of vertices (or nodes) V and a set of edges $E \subseteq V \times V$, where each edge (u, v) connects two vertices u and v. In an undirected graph, edges have no direction, so $(u, v) = (v, u)$. Each node v_i has an associated feature vector $\mathbf{x}_i \in \mathbb{R}^C$, where C is the dimensionality of the node features. The structure of a graph can be represented by the **adjacency matrix** $\mathbf{A} \in \mathbb{R}^{n \times n}$,



Adjacency matrix A

0	1	0	0	1
1	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	1	0	0	0

Degree matrix D

2	0	0	0	0
0	4	0	0	0
0	0	2	0	0
0	0	0	2	0
0	0	0	0	2

Figure 2.1: Illustration of a simple undirected graph with five nodes and five edges (excluding self-connections). Each node v_i is connected to others as indicated by the edges, demonstrating the basic structure of a graph used in GCNs.

where $n = |V|$ is the number of nodes. The element A_{ij} is defined as:

$$A_{ij} = \begin{cases} 1, & \text{if there is an edge between nodes } v_i \text{ and } v_j, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

The **degree matrix** $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix where each diagonal element D_{ii} is the degree of node v_i, defined as the number of edges connected to it:

$$D_{ii} = \sum_{j=1}^{n} A_{ij}. \tag{2.2}$$

These components are sufficient to construct the local Graph convolutional update law from [63], which modifies the value of each node $\mathbf{x}_i$ features based on a weighted sum of the features of nodes connected to it:

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right), \tag{2.3}$$

where $\mathbf{x} \equiv H^{(l)} \in \mathbb{R}^{n \times C_l}$ is the feature matrix at layer l, $W^{(l)} \in \mathbb{R}^{C_l \times C_{l+1}}$ is the learnable weight matrix at layer l, $\hat{A}$ is the renormalized adjacency matrix[1]. Information can be propagated throughout a graph by stacking layers of the GCN-update law, where the information in each node is adjusted based on the representation in the neighboring nodes[2], expanding the receptive field for each repeated layer. However, the conceptualization of GCNs is based on graph spectral convolutions from spectral graph theory (SGT) [27]. Following the key theoretical concepts of SGT leading to the development of GCNs provides a more thorough understanding of the benefits and use cases of GCNs.

A central concept in spectral graph theory is the **graph Laplacian**. The *combinatorial Laplacian* L is defined as:

$$L = D - A. \tag{2.4}$$

Alternatively, the *normalized Laplacian* L_{Norm} is defined as:

$$L_{\text{Norm}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}, \tag{2.5}$$

where I is the identity matrix. The Laplacian operator reflects the connectivity of the graph and measures the rate at which a signal at a vertex changes relative to its neighbors.

The Laplacian matrices L and L_{Norm} are symmetric and positive semi-definite, and their eigenvalues λ_i and eigenvectors $\mathbf{u}_i \in \mathbb{R}^n$ capture important properties of the graph's structure. Specifically, the eigenvalues represent graph frequencies, i.e., the transmission of information in the graph. The eigenvectors form an orthogonal basis representing these graph variations. The eigen-decomposition of the normalized Laplacian L_{Norm} is given by:

$$L_{\text{Norm}} = U \Lambda U^{\top}, \tag{2.6}$$

where $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n]$ is the matrix of eigenvectors, $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues.

[1] Will be introduced down below

[2] And the node itself through self-loops

2.1.1 Graph Fourier Transform

In spectral graph theory, equivalently to the classical Fourier transform, the **Graph Fourier Transform** (GFT) can be used to represent signals on a graph in the spectral domain. Given a graph signal $\mathbf{x} \in \mathbb{R}^n$, where f_i is the signal value at node v_i, the $\mathcal{GFT}$ and its inverse $\mathcal{IGFT}$ are defined as:

$$\hat{\mathbf{x}} = U^\top \mathbf{x}, \quad \mathbf{x} = U\hat{\mathbf{x}}. \tag{2.7}$$

with $\hat{\mathbf{x}} = (\hat{x}(\lambda_1), \hat{x}(\lambda_2), \ldots, \hat{x}(\lambda_m))^\top$ being the signal expressed in the Fourier basis. The transformation enables the analysis of graph signals expressed through their frequency components since any signal can be expressed through the Laplace eigenvector basis:

$$f = \hat{x}(\lambda_1)u_1 + \hat{x}(\lambda_2)u_2 + \cdots + \hat{f}(\lambda_n)u_n = \sum_{i=1}^{n} \hat{f}(\lambda_i)u_i, \tag{2.8}$$

2.1.2 Spectral Graph Convolution

The core idea of spectral graph convolution is to perform convolution operations in the spectral domain by leveraging the eigenvalues and eigenvectors of the Laplacian. The convolution theorem states [80] that the convolution operation between two signals in the Fourier domain is simply the point-wise product of their respective Fourier transformers. The convolution of a signal $\mathbf{x}$ with a filter g_θ is defined as:

$$g_\theta \star \mathbf{x} = \mathcal{IGFT}(\mathcal{GFT}(g_\theta) \odot \mathcal{GFT}(\mathbf{x})) \tag{2.9}$$

$$= U(U^\top g_\theta \odot U^\top \mathbf{x}) \tag{2.10}$$

$$= U g_\theta(\Lambda) U^\top \mathbf{x}, \tag{2.11}$$

where $\odot$ is the Hadaman product, and $g_\theta(\Lambda)$ is a diagonal matrix of spectral filter coefficients parameterized by θ, i.e.,

$$g_\theta(\Lambda) = \begin{bmatrix} g_\theta(\lambda_1) & & & \\ & g_\theta(\lambda_2) & & \\ & & \ddots & \\ & & & g_\theta(\lambda_n) \end{bmatrix} \tag{2.12}$$

The operation $U^\top \mathbf{x}$ transforms the signal to the spectral domain, $g_\theta(\Lambda)$ applies the filter, and U transforms the result back to the spatial domain.
While the concepts are presented for scalar signals $\mathbf{x} \in \mathbb{R}^n$ for clarity, they naturally extend to signals with multiple features $\mathbf{X} \in \mathbb{R}^{n \times C}$ by applying the operations to each feature channel independently:

$$g_\theta \star \mathbf{X} = U g_\theta(\Lambda) U^\top \mathbf{X}. \tag{2.13}$$

This allows Graph Convolutional Networks to process complex node features in practical applications, which is how they are used in practice.
Computing the eigen-decomposition U is computationally expensive for large graphs (complexity $O(n^3)$). To address this issue, Defferrard et al. [29] proposed approximating the spectral filter $g_\theta(\lambda)$ using a truncated expansion in terms of Chebyshev polynomials $T_k(\tilde{\lambda})$, which allows efficient recursive computation without the need for explicit eigen-decomposition. The filter $g_\theta(\lambda)$ is approximated as:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), \tag{2.14}$$

$$= \begin{bmatrix} \sum_{k=0}^{K-1} \theta_k T_k(\lambda_1) & & & \\ & \sum_{k=0}^{K-1} \theta_k T_k(\lambda_2) & & \\ & & \ddots & \\ & & & \sum_{k=0}^{K-1} \theta_k T_k(\lambda_n) \end{bmatrix}, \tag{2.15}$$

where $\tilde{\lambda} = \frac{2\lambda}{\lambda_{\max}} - 1$ scales the eigenvalues to the range $[-1, 1]$, $\lambda_{\max}$ is the largest eigenvalue of L_{Norm} (for the normalized Laplacian, $\lambda_{\max} \leq 2$, so $\lambda_{\max}$ can be set to $\lambda_{\max} \approx 2$), θ_k are the Chebyshev coefficients (learnable parameters), $T_k(\tilde{\lambda})$ are the

Chebyshev polynomials, defined recursively as:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x). \tag{2.16}$$

Substituting this approximation into Equation (2.11), the convolution becomes:

$$g_\theta \star \mathbf{x} \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{L}_{\text{Norm}})\mathbf{x}, \tag{2.17}$$

where $\tilde{L}_{\text{Norm}} = \frac{2}{\lambda_{\max}} L_{\text{Norm}} - I$. This formulation avoids explicit computation of U and leverages the sparsity of L_{Norm} for efficient computation. Moreover, the filters are localized, capturing information within K hops of each node.

2.1.3 Simplified Graph Convolutional Network

For further simplification, [63] explored setting $K = 1$ and approximating $\lambda_{\max} \approx 2$, which is reasonable for the normalized Laplacian. The Chebyshev polynomials simplify to:

$$T_0(\tilde{L}_{\text{Norm}}) = I, \quad T_1(\tilde{L}_{\text{Norm}}) = \tilde{L}_{\text{Norm}}. \tag{2.18}$$

The convolution operation becomes:

$$g_\theta \star \mathbf{x} \approx \theta_0 \mathbf{x} + \theta_1 \tilde{L}_{\text{Norm}} \mathbf{x}. \tag{2.19}$$

By setting $\theta = \theta_0 = -\theta_1$, this simplifies to:

$$g_\theta \star \mathbf{x} \approx \theta(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})\mathbf{x}. \tag{2.20}$$

To prevent numerical instabilities and ensure consistent scaling, a **renormalization trick** is applied by adding self-connections to the graph and modifying the adjacency matrix:

$$\tilde{A} = A + I, \tag{2.21}$$

and updating the degree matrix accordingly:

$$\tilde{D}_{ii} = \sum_{j=1}^{n} \tilde{A}_{ij}. \tag{2.22}$$

This leads to the normalized adjacency matrix:

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}. \tag{2.23}$$

The convolution operation simplifies to:

$$g_\theta \star \mathbf{x} \approx \theta \hat{A} \mathbf{x}. \tag{2.24}$$

This relation is for the convolution of a single feature vector $\mathbf{x}$. It can be generalized to multiple features and multiple filters. For node features $X \in \mathbb{R}^{n \times C}$ with C feature channels, and filters represented by the weight matrix $\Theta \in \mathbb{R}^{C \times F}$, the convolution becomes:

$$Z = \hat{A} X \Theta, \tag{2.25}$$

where $Z \in \mathbb{R}^{n \times F}$ is the output feature matrix, and F is the number of output feature channels.

2.1.4 Layer-wise Propagation in GCNs

Using a non-linear activation function σ such as ReLU[2] or GELU [53]), the layer-wise propagation rule for a GCN becomes:

$$H^{(l+1)} = \sigma\left(\hat{A} H^{(l)} W^{(l)}\right), \tag{2.26}$$

where $H^{(l)} \in \mathbb{R}^{n \times C_l}$ is the feature matrix at layer l, $W^{(l)} \in \mathbb{R}^{C_l \times C_{l+1}}$ is the learnable weight matrix at layer l, $\hat{A}$ is the normalized adjacency matrix.
Equation (2.26) shows how the graph convolution operation aggregates information from neighboring nodes, effectively performing a form of **Laplacian smoothing**. The operation $\hat{A} H^{(l)}$ updates each node's feature vector by combining its features with those of its neighbors, weighted by the renormalized adjacency matrix. The com-

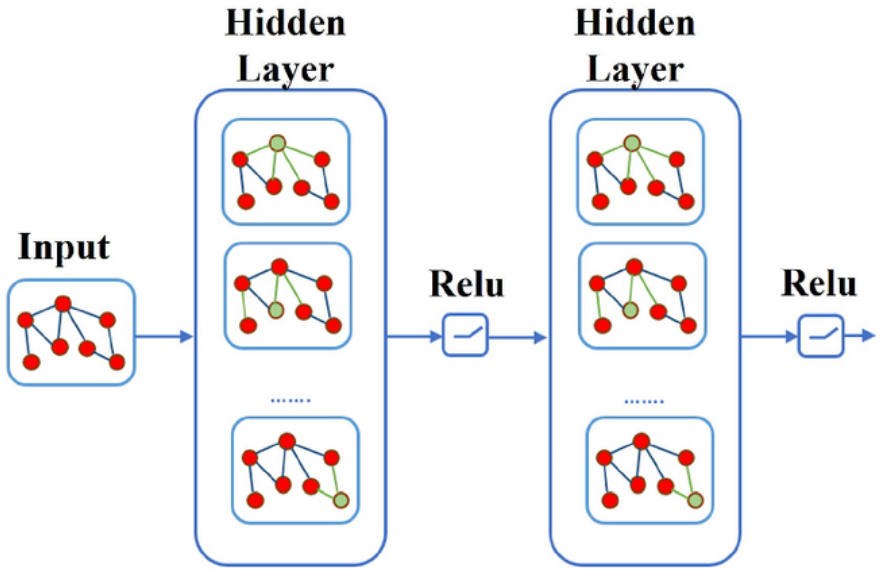


Figure 2.2: Visualization of a two-layer GCN network. Each layer performs feature transformation and neighborhood aggregation, expanding the receptive field to capture local and global graph structures. Adapted from [91].

plete GCN architecture consists of stacking multiple layers of the propagation rule in Equation (2.26). As stated, when the GCN update law was introduced, the receptive field expands by stacking layers, allowing the network to capture information from nodes multiple hops away. For example, a two-layer GCN can be expressed as:

$$H^{(2)} = \sigma_2\left(\hat{A}\,\sigma_1\left(\hat{A}H^{(0)}W^{(0)}\right)W^{(1)}\right), \tag{2.27}$$

where $H^{(0)} = X$ is the input feature matrix,$H^{(2)} = X_{\text{out}}$ is the output feature matrix, $W^{(0)}$ and $W^{(1)}$ are the learnable weight matrices, σ_1 and σ_2 are activation functions. By aggregating each node's weighted node features in the second layer, each node will be influenced by (at most) its neighbors' neighbors. The information would, therefore, be localized to 2 jumps on the graph in this case, and thus, for nodes to obtain global information from a GCN, the layers must be stacked multiple times.

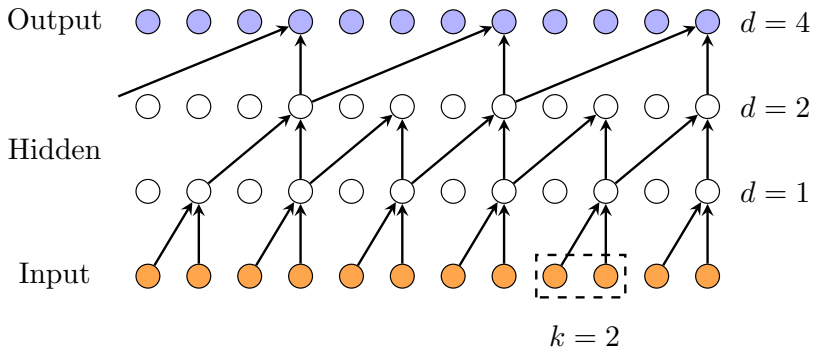


Figure 2.3: Visualization of a TCN with a kernel of 2, stride of 1, and dilations of 1, 2, and 4, respectively. Note that, for clarity, not all connections between layers are included.

2.2 Temporal Convolutional Networks

Temporal Convolutional Networks (TCNs) are a subcategory of CNNs designed to model 1D sequential input, in contrast to general CNNs typically applied to 2D data structures like images. TCNs apply convolutional operations on temporal sequences to capture temporal dependencies. They offer several benefits compared to sequential models like Recurrent Neural Networks (RNNs), primarily because TCNs are feed-forward architectures. This feed-forward nature allows for parallel processing of kernel convolutions on the sequence elements. Thus, it alleviates issues with vanishing or exploding gradients commonly encountered in RNNs, as gradients in TCNs do not need to pass through recurrent connections over many time steps. Another trait of TCNs (and convolutional networks in general) is the inductive bias of locality, which is achieved by utilizing local kernels to process a sequence iteratively. The inductive bias is a trade-off differentiating TCNs from the Transformer models with no inductive biases. As a result, TCNs are often faster and more efficient to train than Transformer models, which tend to require more data to learn the general properties of a task.

Although the operation is identical, TCNs are distinguished by performing causal or acausal convolutions. In causal TCNs, the convolution is designed such that the output at time t depends only on inputs from time t and earlier, ensuring no information leakage from the future. This is achieved by shifting the convolution operation, which

is described by the following equation:

$$y(t) = \sum_{i=0}^{k-1} W^{(i)} \cdot x_{t-i} \tag{2.28}$$

Here, $W^{(i)} \in \mathbb{R}^{D \times C}$ is the element-specific filter/weight matrix that $y(t) \in \mathbb{R}^D$ is the output at time t, and the inputs $x_{t-i} \in \mathbb{R}^C$ are restricted to previous time steps (including t), but in tasks like time-series forecasting $y(t)$ is trying to predict future events (which it only has "historic" knowledge about.
The kernel size k is a hyperparameter describing the number of temporal inputs processed by 1D filters at each step. Similarly, the stride s controls the step size with which the filter moves across the input sequence.[3] Since increasing the stride reduces the dimensionality of the output, it is often used for sequence downsampling but as a trade-off, reducing the temporal resolution. [4]
In contrast, acausal TCNs use past and future information by symmetrically centering the convolutional filter around the current time step. The formula for acausal convolution is:

$$y(t) = \sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} W^{(i)} \cdot x_{t+i}, \tag{2.29}$$

where variables and parameters are the same as in Equation 2.28 but now $y(t)$ depends on inputs from past and future time steps, seen by the shift of indexing on x relative to the kernel (receptive field). In this thesis, Acausal TCN variants will mainly be used in the thesis, but bidirectional causal TCNs are also explored [130]. Bidirectional TCNs (BiTCN) operate causally from 0 to T in the sequence. Subsequently, a new TCN operates causally from T to 0 in the sequences, allowing the model to look at the signal in both temporal directions. Lastly, like the 2D version, TCN architectures also use dilated convolutions to expand the receptive field with-

[3] Note that the stride does not appear Equation 2.28equation since it is the temporal convolution at a single timestep t. But is introduced when iterating over the entire sequence.

[4] Note that compared to the typical formulation of 2D convolutions that relies on the cross-correlation operation, $W^{(i)} \cdot x_{t-i}$ is here a matrix/inner product, which is because the equation has a sum over the time-steps in the receptive field instead of the cross-correlation product.

out increasing the parameters (increasing filter size k) or the depth of the network by adding layers. The dilation factor, d, determines the spacing between elements in the convolutional kernel. Increasing the dilation factor $d > 1$ does not lower the output dimensionality. The formula for dilated convolutions is:

$$y(t) = \sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} W^{(i)} \cdot x_{t+d \cdot i} \tag{2.30}$$

In this equation, the dilation factor d modifies the indexing of the input sequence $x \in \mathbb{R}^{T \times C}$. A visualization of a three-layer causal TCN with different dilation factors in each layer is shown in Figure 2.3.
The training of deeper TCNs often benefits from including residual connections. Where the residual blocks can be expressed as:

$$O = \sigma(X + F(X)), \tag{2.31}$$

where $F(x) \in \mathbb{R}^{T \times D}$ represents the output from the convolutional layers, $X\mathbb{R}^{T \times C}$ is the input to the block, and $O \in \mathbb{R}^{T \times D}$ is the layer output. A non-linear activation function, σ (e.g., ReLU/GELU), is applied after aggregating the input and the processed signal. If the input channel size changes during the convolutional layer, i.e., $D \neq C$, a 1×1 convolution is added to X in Equation 2.31 to match the dimensions. For the computation of single output elements, the stride s does not play a role in the output value at a specific timestep t. However, s affects the overall output sequence length. If the input sequence X has length T, the length of the output sequence O with stride s can be calculated as:

$$\text{Length}(y) = \left\lceil \frac{T - k + 2p + 1}{s} \right\rceil, \tag{2.32}$$

where p is the amount of padding applied to one side of the sequence (total padding is $2p$). Padding (typically zero-padding) preserves the length of the input sequence with $2p = k - 1$ when $s = 1$. This TCN introduction considers single-batch input. Otherwise, the input $X \in \mathbb{R}^{N \times T \times C}$ would be a 3-tensor instead, with N being the batch size.

2.3 Transformer

The transformer neural architecture, introduced by Vaswani et al. [108], greatly changed the handling of sequential data by relying entirely on self-attention mechanisms without relying on recurrence. Transformers are designed to handle sequential data efficiently, and their core self-attention mechanism allows for increased parallelization during training compared to previous approaches like RNNs. After Dosovitskiy et al. [34] demonstrated that a Vision Transformer (ViT) achieved remarkable results for image classification tasks, transformers have since been used for many vision-related tasks. The subsequent introduction to transformers takes inspiration from [34] and [108].

Before being passed to the transformer block, the input data is converted to a sequence of tokens $\mathbf{x}$ with

$$\mathbf{x} = [W_{\text{emb}}x_1,\ \ldots,\ W_{\text{emb}}x_n,\ \ldots,\ W_{\text{emb}}x_N]^{\text{T}} \in \mathbb{R}^{N\times D_L}, \tag{2.33}$$

where each token $x_n \in \mathbb{R}^C$ is a vector representation of token n in the sequence, and $W_{\text{emb}} \in \mathbb{R}^{C\times D_L}$ is a learned weight matrix that maps each token to the embedded feature space with dimension D_L. Converting raw input data into tokens is called *tokenization*. For 2D data (e.g., images or skeleton sequences), tokenization is done by dividing the signal into N vectors,e.g., image patches or individual skeleton poses, which are then flattened to dimension C and mapped to the embedded space, resulting in a transformer-ready input signal $\mathbf{x}$. For 1D signals like text strings, an embedding table is most commonly used to create the token sequence.

The embedded tokens defined in (2.33) are then passed to the transformer block, which consists of L transformer layers, where L is the transformer depth. To distinguish between the tokens at different layers, they are defined as $\mathbf{x}^{(l)}$ after having passed through layer l, where $\mathbf{x}^{(0)}$ is the embedded input sequence, and $\mathbf{x}^{(L)}$ is the output of the transformer, which is still of size $N \times D_L$. Each layer is composed of a multi-head self-attention (MHSA) module, layer normalization (LN) [6], and a multi-layer perceptron (MLP), which consists of two linear projections separated by a GELU activation [53] and dropout. The design of a single transformer layer is illustrated in Figure 2.4 on the left.

The following equations describe the transformer layer:

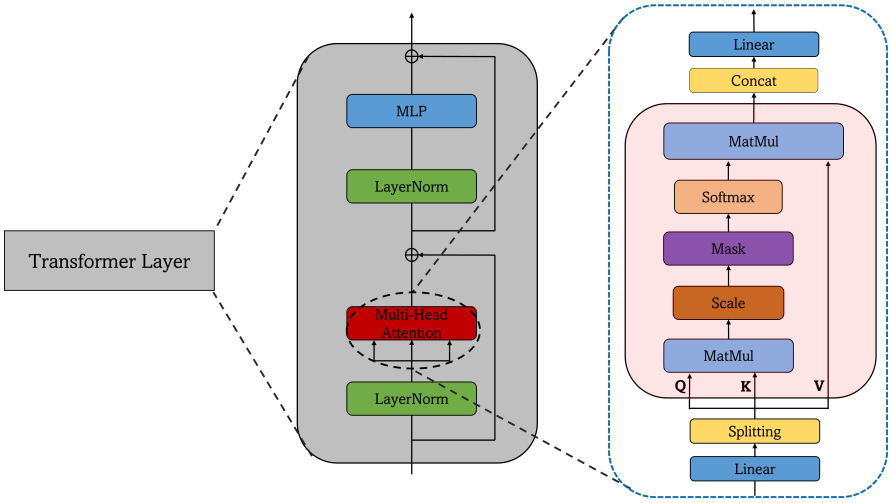


Figure 2.4: Components in a transformer layer are shown in the block to the left. The right block shows the composition of a single-head self-attention module. The Mask leaves out the (zero) padded tokens in the attention map, which allows the model to handle sequences of varying lengths proficiently.

$$\tilde{\mathbf{x}}^{(l+1)} = \mathbf{x}^{(l)} + \mathrm{MHA}(\mathrm{LN}(\mathbf{x}^{(l)})) \tag{2.34}$$

$$\mathbf{x}^{(l+1)} = \tilde{\mathbf{x}}^{(l+1)} + \mathrm{MLP}(\mathrm{LN}(\tilde{\mathbf{x}}^{(l+1)})), \tag{2.35}$$

where $\tilde{\mathbf{x}}^{(l+1)}$ is the intermediate representation obtained after the self-attention module.

Self-Attention Mechanism The self-attention mechanism computes attention scores between all pairs of tokens in the sequence, allowing the model to weigh the relevance of each token when encoding a particular token. The attention mechanism is defined as:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{D_k}}\right)V, \tag{2.36}$$

where Q, K, and V are the queries, keys, and values, which are linear projections of the input tokens $\mathbf{x}^{(l)}$:

$$Q = \mathbf{x}^{(l)}W_q \in \mathbb{R}^{N\times D_k},\ K \quad = \mathbf{x}^{(l)}W_k \in \mathbb{R}^{N\times D_k},\ V = \mathbf{x}^{(l)}W_v \in \mathbb{R}^{N\times D_v}, \tag{2.37}$$

where $W_q \in \mathbb{R}^{D_L \times D_k}$, $W_k \in \mathbb{R}^{D_L \times D_k}$, and $W_v \in \mathbb{R}^{D_L \times D_v}$ are learnable weight matrices, and D_k and D_v are the dimensions of the queries/keys and values, respectively. After scaling and applying the softmax function, (2.36) serves as an attention map that provides context to the value array V. This mechanism enables the model to focus on relevant parts of the sequence when encoding each token.

Multi-Head Attention Transformers use multi-head attention to capture information from different representation subspaces at different positions. The multi-head attention mechanism is defined as:

$$\text{MHA}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.38}$$

$$\text{where head}_i = \text{Attention}(QWq, i,\ KWk, i,\ VWv, i), \tag{2.39}$$

where h is the number of attention heads, $W_{q,i} \in \mathbb{R}^{D_L \times D_k}$, $W_{k,i} \in \mathbb{R}^{D_L \times D_k}$, and $W_{v,i} \in \mathbb{R}^{D_L \times D_v}$ are the weight matrices for queries, keys, and values for head i, and $W^O \in \mathbb{R}^{h \cdot D_v \times D_L}$ is the output projection matrix. While the latent dimensions of Q, K and V i.e.,D_k and D_v are customizable hyperparameters, it is standard, as proposed in [108], to set $D_k = D_v = D_L/h$. Such that the entire concatenated array of h single-head representation has the same embedded dimension as a single-head self-attention representation with $D_k = D_L$[5]. This has been shown to produce a more efficient representation with less computational cost.

Cross-Attention Mechanism In addition to self-attention, transformers utilize cross-attention mechanisms, especially in decoder layers, to allow the model to focus on relevant parts of the input sequence when generating outputs. In cross-attention, the queries (Q) come from the previous decoder layer (or decoder input), while the keys (K) and values (V) come from the encoder's output. This mechanism enables the decoder to attend to the encoder's representations, effectively integrating information from the input sequence into the output generation process.
The cross-attention mechanism is mathematically similar to self-attention but with

[5]Unless explicitly stated otherwise $D_k = D_L/h$ can be assumed.

different inputs:

$$Q_{\text{dec}} = \mathbf{y}^{(l)} W_q^{\text{dec}} \in \mathbb{R}^{M \times D_k},\ K_{\text{enc}} \ = \mathbf{x}^{(L)} W_k^{\text{enc}} \in \mathbb{R}^{N \times D_k},\ V_{\text{enc}} = \mathbf{x}^{(L)} W_v^{\text{enc}} \in \mathbb{R}^{N \times D_v}, \tag{2.40}$$

where $\mathbf{y}^{(l)}$ is the decoder's input at layer l, and $\mathbf{x}^{(L)}$ is the final output of the encoder. The cross-attention is then computed as:

$$\text{Cross-Attention}(Q_{\text{dec}}, K_{\text{enc}}, V_{\text{enc}}) = \text{softmax}\left(\frac{Q_{\text{dec}} K_{\text{enc}}^T}{\sqrt{D_k}}\right) V_{\text{enc}} \tag{2.41}$$

2.3.1 Encoder and Decoder Blocks

The transformer architectures exist in two forms: Encoder-only and encoder-decoder models. The encoder-only model simply consists of a transformer block, which creates an embedded representation of the input sequence. On the other hand, the encoder-decoder model consists of two components: The encoder and the decoder. The **encoder block** processes the input sequence and transforms it into a continuous representation that captures the relevant information from the input. It uses self-attention mechanisms to weigh the importance of different tokens relative to each other. Each encoder layer comprises a multi-head self-attention module and a feed-forward network wrapped with layer normalization and residual connections.
In an encoder-decoder model, the **decoder block**, however, generates the output sequence by predicting each token step-by-step. It employs self-attention to consider the generated tokens and cross-attention to incorporate information from the encoder's output, ensuring that the predictions are contextually relevant to the input. Each decoder layer includes a masked multi-head self-attention module (to prevent access to future tokens), a cross-attention module and a feed-forward network, all with layer normalization and residual connections.

Output Representations for Downstream Tasks Encoder-decoder architectures such as masked autoencoders [51] can be used for self-supervised representation learning of vision tasks. After pretraining, the final output representation of the encoder module $\mathbf{x}^{(L)}$ is suitable for numerous downstream tasks such as classification. One way to do this is by prepending a class token x_{cls} on the token sequence of the input before passing it to the transformer block:

$$\mathbf{x} = [x_{\text{cls}},\ x_1,\ \ldots,\ x_n,\ \ldots,\ x_N]^{\mathrm{T}} \in \mathbb{R}^{(N+1)\times D_L}, \tag{2.42}$$

where $x_{\text{cls}} \in \mathbb{R}^{D_L}$ is a learned class token. The representation of x_{cls} at the final transformer layer $\mathbf{x}_{\text{cls}}^{(L)}$ is used by the MLP head to make predictions. Alternatively, global pooling of $\mathbf{x}^{(L)}$ can be used:

$$O = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n^{(L)}, \tag{2.43}$$

where the aggregated output O is fed to the prediction head.

2.3.2 Masking Techniques

Masking techniques in transformers ensure that the attention mechanism operates correctly for different contexts, including handling sequences of different lengths (padding), preserving autoregressive properties (causal mask), and respecting graph structures (adjacency mask).

Padding Masking : To handle variable-length sequences within a batch, sequences are often padded to a uniform length. Padding tokens should not influence the attention mechanism. This is achieved by setting the attention scores of padding tokens to a large negative value, effectively ignoring them in the softmax computation. If A is the attention score matrix and M is the padding mask, the modified scores $\tilde{A}$ are:

$$\tilde{A}ij = Aij + M_{ij}, \tag{2.44}$$

where M_{ij} is $-\infty$ for padding positions and 0 elsewhere.

Causal Masking For autoregressive tasks, causal masking ensures that each token can only attend to previous tokens in the sequence, preventing information leakage from future tokens. This is achieved by masking out future positions in the attention computation:

$$\tilde{A}ij = \begin{cases} Aij, & \text{if } j \leq i \\ -\infty, & \text{if } j > i. \end{cases} \tag{2.45}$$

Adjacency Masking In graph transformers, masking retains the graph's topology. The adjacency matrix A_{ij} can be used to enforce that each node only attends to its neighbors. The attention scores between unconnected nodes are masked out:

$$\tilde{A}_{ij} = \begin{cases} A_{ij}, & \text{if } A_{ij} = 1, \\ -\infty, & \text{if } A_{ij} = 0. \end{cases} \tag{2.46}$$

2.3.3 Positional Encoding

Transformers treat each position in the input sequence equally, lacking an inherent notion of sequential order. To provide positional information, positional encodings are added to the input embeddings.

Absolute Positional Encoding A learnable positional embedding matrix[6] $E_{\text{pos}} \in \mathbb{R}^{(N)\times D_L}$ is added to the token embeddings:

$$\mathbf{x}^{(0)} = \mathbf{x} + E_{\text{pos}}, \tag{2.47}$$

where $\mathbf{x}$ is the embedded input sequence from (2.47) (including the class token if used).
Alternatively, sinusoidal positional encodings use sine and cosine functions of different frequencies, as first proposed in [108]:

$$PE(n, 2i) = \sin\left(\frac{n}{10000^{2i/D_L}}\right), \; PE(n, 2i+1) \qquad = \cos\left(\frac{n}{10000^{2i/D_L}}\right), \tag{2.48}$$

where n is the position index (from 0 to N), and i is the dimension index (from 0 to $D_L/2 - 1$). The positional encoding vector for position n is constructed by concate-

[6] $E_{\text{pos}} \in \mathbb{R}^{(N+1)\times D_L}$ including the x_{cls} token.

nating these values:

$$E_{\text{pos},n} = [PE(n,0),\ PE(n,1),\ \ldots,\ PE(n,D_L-1)] \in \mathbb{R}^{D_L}. \tag{2.49}$$

2.3.4 Graph Attention and Graph Transformers

Transformers can be viewed as a generalization of graph-based models by interpreting each token as a node and the attention mechanism as a learned adjacency structure. Previously, attention was introduced as a means of computing context-dependent weights between sequence elements. When applied to graph data, attention operates on node embeddings and dynamically determines which nodes to highlight. In Graph Attention Networks (GATs) [110], the model learns to focus on informative neighbors rather than relying solely on predefined edges. GATs adaptively refine graph connectivity based on node features and the specific prediction task by treating node-to-node interactions as a learned attention pattern. Graph Transformers [37] extends these concepts further. Rather than restricting their receptive field to local neighborhoods, they leverage global self-attention, potentially allowing any node to attend to any other node. Though still guided by the underlying graph structure through masking or positional encodings, these models can more naturally capture long-range dependencies and complex topological relationships than standard GCNs. The resulting architecture combines the structural awareness of graph methods with the powerful representational flexibility of transformers, enabling more effective learning from graph-structured data.

2.4 Chapter Conclusion

This chapter introduced GCNs, TCNs, and Transformers, which will serve as the primary building blocks for the more complex model architectures used throughout the thesis.

Chapter 3

Human Pose Estimation

3.1 Introduction

This chapter introduces human pose estimation (HPE). HPE's objective is to locate key body joints and capture the posture and movements of subjects within an image or video frame. HPE is central to various applications such as motion analysis, surveillance, action recognition, etc., and plays an important role in this thesis, as skeleton data are the main input feature in most studies.
Skeleton features represent the human body using key joint positions, offering a distilled but highly informative view of human motion. They allow models to efficiently process and analyze the core aspects of physical actions without the computational overhead and potential redundant information presented by full video data such as coloring, background clutter, lighting differences, etc. Using the lower-dimensional data enhances the efficiency and improves the robustness of action recognition systems, improving generalization across different settings and environments, since all nuisances of the environment are removed with skeleton features.

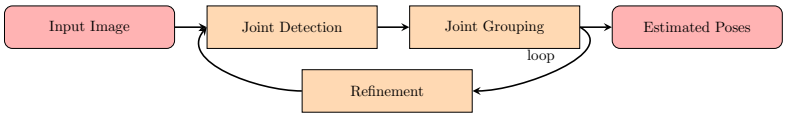
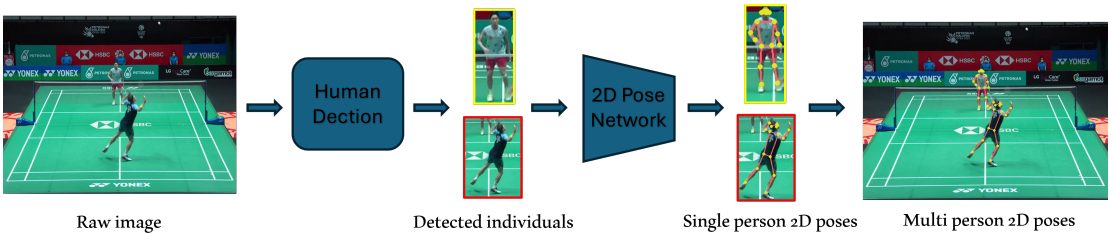


Figure 3.1: The procedure of Human Pose Estimation



(a) Top-Down Pose Estimation Pipeline

(b) Bottom-Up Pose Estimation Pipeline

Figure 3.2: Comparison of top-down and bottom-up pose estimation pipelines.

3.2 Methods of Human Pose Estimation

Human pose estimation can be divided into two approaches: Bottom-up and Top-down. Below is a description of the two approaches' main ideas, similarities, and differences. OpenPose [13] (Bottom-up) and HRNet [100] (Top-down). The figure also illustrates the relevant features of HPE models.

3.2.1 Bottom-Up Methods

The bottom-up approach first detects all key joint point candidates across an image with a deep learning architecture, commonly CNNs [13, 23, 84], that outputs confidence maps for each joint type, indicating the likelihood of each joint's presence at a specific pixel coordinate. For bottom-up HPE models, the next step involves grouping these joints into individual human poses, which is often difficult in crowded scenes. Techniques such as Part Affinity Fields (PAFs), which predict vector fields to represent the association and orientation between joint pairs, are used by [13, 84]. [81] uses hierarchical clustering, which creates high-dimensional associative embeddings of

joints to place joints belonging to the same person close together such that a clustering algorithm can be used to group the embedded joints into individual poses. They also use graph-based matching methods to select appropriate joint detections for each individual [84]. These methods construct a graph where joints are nodes and potential limb connections are edges, using graph partitioning algorithms to delineate individual poses. This grouping may be refined through iterative processes or additional network layers to improve accuracy, particularly in complex scenarios like occlusions or closely interacting individuals, as illustrated in Figure 3.1. OpenPose [13] is a well-known bottom-up method used as an inference model due to decent generalization on various datasets.

OpenPose

OpenPose architecture utilizes sequences of convolutional blocks to estimate keypoint candidates first, followed by estimates of their proposed PAFs for keypoint grouping. The model scales well for multi-person pose estimation using a parsing algorithm that splits the grouping of detected body parts into the matching of multiple bipartite (unconnected) graph line integrals of the PAFs between candidates.
PAFs are a set of 2D vector fields representing limbs of the human body (e.g., a forearm from the elbow to the wrist) to encode both the position and orientation of limbs between potentially associated joint candidates in the image. For each pixel in the image, the PAF encodes the direction of the limb in that pixel and the degree of association between connected body parts. OpenPose has a unique solution for creating ground truth data. If p is a coordinate in a PAF, $L_c(p)$, for limb c, The ground truth PAF, $L_c^*(p)$, is defined as a unit vector pointing from $\mathbf{x_{j_1}}$ to $\mathbf{x_{j_2}}$, where $\mathbf{x_{j_2}}$ and $\mathbf{x_{j_2}}$ are the two joints connected by the limb. This vector is nonzero only along the limb and is zero elsewhere. The ground truth PAF, L_c^*, at a point p on the limb of person k is given by:

$$L_{c,k}^*(p) = \begin{cases} \mathbf{v} & \text{if } p \text{ on limb } c,k \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

where $\mathbf{v}$ is the unit vector in the direction of the limb from one joint to another. OpenPose uses confidence maps to indicate the probability of a specific joint at each location in the image. For each body part j, a confidence map S_j is created where

each pixel value represents the likelihood of that part appearing at that location. Given the ground truth position $x_{j,k}$ of part j for person k, the confidence value at location p is given by:

$$S^*_{j,k}(p) = \exp\left(-\frac{||p - x_{j,k}||^2}{\sigma^2}\right), \tag{3.2}$$

where σ is the standard deviation, a hyperparameter, that dictates confidence spread around the part location.

The final pose estimate from the PAFs and confidence maps is handled by a so-called greedy parsing algorithm. The algorithm evaluates the degree of alignment between the detected body parts and the vector fields in the PAFs by computing a line integral (in practice, a sum of uniformly spread points) along the line segment connecting two candidate body part locations. The integral of the vector field along the line segment between two points dj_1 and dj_2 (part detections) is given by:

$$E = \int_{u=0}^{1} L_c(p(u)) \cdot \frac{dj_2 - dj_1}{||dj_2 - dj_1||} du, \tag{3.3}$$

where $p(u) = (1 - u)dj_1 + udj2$ interpolates between the two body parts. Using the above measure, candidate limbs are scored and assembled into complete poses using a method akin to bipartite graph matching [120], where the objective is to maximize the total association score.

Through these implementations, OpenPose can robustly detect and assemble human poses in real-time, even in complex multi-person scenes, by effectively balancing detail capture (via PAFs) and computational efficiency (via greedy parsing).

3.2.2 Top-Down Methods

The top-down approach to human pose estimation operates in two distinct phases: Individual (object) detection and subsequent pose estimation, as shown in Figure 3.2. Before doing pose-estimation, top-down models employ object detection models such as Faster R-CNN [93], YoloX [45] or a ResNet detection model [52] to scan the image or video frame to identify each individual. A bounding box is provided for each proposed individual, isolating them from the background and other individuals in the scene (Occlusion and overlapping individual bounding boxes are a persisting chal-

lenge).
Once individuals are detected, the next phase focuses exclusively on analyzing the contained regions to estimate the pose of the single individuals. The main model performs single-person pose estimation using a deep neural architecture. Many CNN (Residual) backbones [100, 82, 21, 121] have shown good results. They excel at multi-scale, capturing relevant information at different image resolutions. Recently, attention-based [123, 69, 23] models utilize a hybrid of attention mechanisms (Transformer blocks) and CNNs to improve spatial relationship handling and scene contextual understanding.
Although less common, top-down approaches also use GNN-based architecture, such as [10] to provide spatial awareness between keypoint representations. However, joint matching techniques are less needed since top-down approaches are trained for single-person pose estimation. In principle, top-down models can be trained for bottom-up pose estimation and vice versa. The models' architecture choice affects the performance in top-down/bottom-up approaches, but most of the difference lies in the task (training procedure) rather than the model structure.
Additionally, other research have suggested advancements in training strategies. In adversarial training [20, 106] uses a discriminator to improve model robustness against occlusion and other challenges. In [123] they use pretraining and [55] semi-self-supervised learning to improve performance and generalization of their respective pose estimation models by pretraining on ImageNet [30]. The pretraining benefits especially the data-hungry transformer HPE models.
Despite the extra computation step associated with the initial detection of each individual, top-down approaches are more accurate than multi-person / bottom-up models and more valued in applications such as sports analytics, where understanding the detailed dynamics of individual movements is crucial. HRNet [100] [23] has been a reliable off-the-shelf / pretrained model for doing HPE on unseen datasets.[1]

HRNet

HRNet can predict human pose accurately by keeping high-resolution information throughout the network. HRNet consists of 3 main components: Residual convolu-

[1] As noted in the last paragraph HRNet is capable of both top-down and bottom-up but is used predominantly as a top-down HPE.

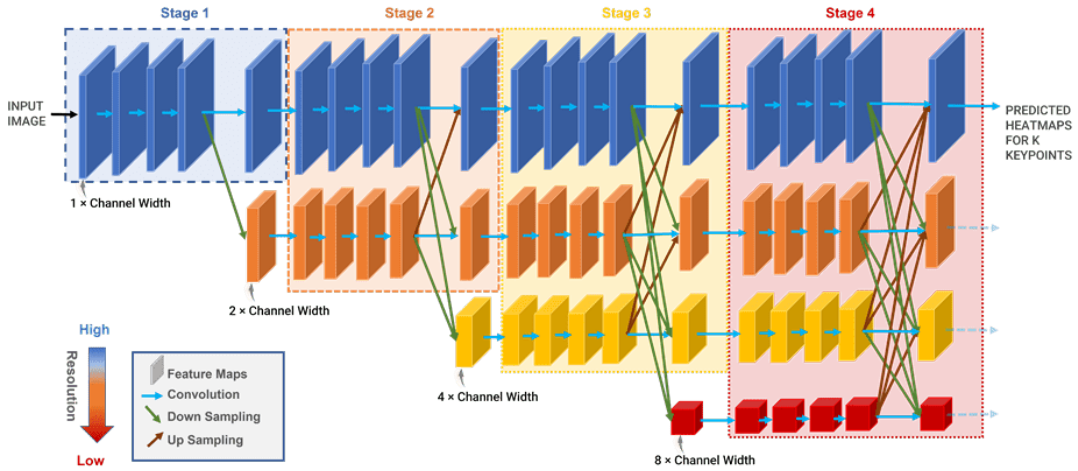


Figure 3.3: Overview of the HRNet architecture. The figure is adapted from a Matlab tutorial, Matlab HRNet.

tional units, adapted from Resnet-50 [52], parallel multi-resolution subnetworks, and repeated exchange blocks. The full architecture is shown in Figure 3.3 for the full architecture.

The output $\mathbf{y}$ of the bottleneck residual unit, assuming matching in/out dimensions, is given by:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}, \tag{3.4}$$

where the function $\mathcal{F}(\mathbf{x}, \{W_i\})$ consists of three convolutional layers, giving the complete equation:

$$\mathbf{y} = \mathrm{BN}(\mathrm{Conv}_{1\times1}(W_3, \sigma(\mathrm{BN}(\mathrm{Conv}_{3\times3}(W_2, \sigma(\mathrm{BN}(\mathrm{Conv}_{1\times1}(W_1, \mathbf{x}))))))))) + \mathbf{x}, \tag{3.5}$$

where $\mathrm{Conv}_{k\times k}(W, \mathbf{x})$ refer to 2D convolution operations with kernel size $k \times k$ and weights W, applied to input $\mathbf{x}$. $\mathrm{BN}(\cdot)$ denote batch normalization, and σ is the activation function, in this case ReLU (Rectified Linear Unit) activation function, applied element-wise.

HRNet starts with a high-resolution subnetwork (stage 1) and gradually adds lower-

resolution subnetworks at each stage while maintaining the connections in parallel. Specifically, HRNet utilizes four subnetworks in parallel across four stages (see Figure 3.3).

Another central part of the HRNet is exchanging information between different subnetworks. This is achieved using exchange units that share feature information across resolution subnetworks. The exchange unit uses strided convolutions and nearest-neighbor interpolation for up and down sampling. All subnets then receive the aggregated contribution from the other subnetworks (and identity residual connection). The exchange unit $\mathcal{E}_i^s$ is potentially repeated i times between residual units to facilitate the exchange between subnetworks effectively. The repeated exchange units make up an exchange block $\mathbf{E}$. Below, a 2-unit exchange block is depicted.

$$\mathbf{E} = \begin{array}{ccccccccc} & \mathcal{C}_{31}^1 & \searrow & & \nearrow & \mathcal{C}_{31}^2 & \searrow & & \nearrow \\ & \mathcal{C}_{32}^1 & \rightarrow & \mathcal{E}_3^1 & \rightarrow & \mathcal{C}_{32}^2 & \rightarrow & \mathcal{E}_3^2 & \rightarrow \\ & \mathcal{C}_{33}^1 & \nearrow & & \searrow & \mathcal{C}_{33}^2 & \nearrow & & \searrow \end{array}, \tag{3.6}$$

where C_{sr}^i denotes the residual unit at resolution r and unit i. The number of exchange units in a block is different for each stage of the main network. The entire network is shown in Figure 3.3.

Ultimately, the heatmaps are estimated from the high-resolution outputs at the last stage of the subnetworks by an MLP head. The model is optimized by minimizing the Mean Squared Error (MSE) between predicted and ground truth keypoint/heatmaps, which similarly to [13] generates a heatmap for each annotated keypoint with mean of the image coordinates and std of 1. The MSE loss is computed as follows:

$$L = \frac{1}{N} \sum_{i=1}^{N} \|H_{pred,i} - H_{gt,i}\|^2, \tag{3.7}$$

where $H_{pred,i}$ is the predicted heatmap, $H_{gt,i}$ is the ground truth heatmap, and N is the number of keypoints.

Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle

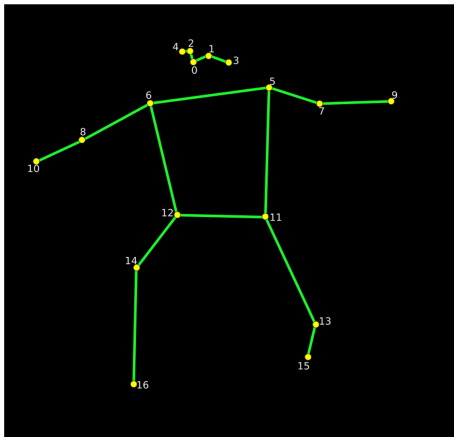


Figure 3.4: Coco Keypoints visualization for a badminton example. Credit for the original foto to the left goes to **@Badmintonphoto**

3.3 Quality of Pose Estimation on Badminton videos

This section evaluates the pretrained HPE models, HRNet[100] and OpenPose [13] on a badminton-specific dataset constructed for images of badminton broadcasted matches.

3.3.1 Pretrained on MS COCO dataset.

The weights of both models are from training on the Microsoft COCO dataset [70] with 250,000 individuals labeled with keypoints. The annotation format has 17 possible keypoints on the human body. They include ears, eyes, nose, shoulders, elbows, hands, hips, knees, and ankles. The complete skeleton with the respective keypoints is shown in Figure 3.4.

3.3.2 Metrics

First, a brief introduction to the two metrics used to evaluate the quality of predicted poses in an image: The Percentage of Correct Keypoints (PCK) and the precision of the Object Keypoint Similarity (OKS). The fundamental metrics of recall, precision, and F1-score are also introduced in this section.

Percentage of Correct Keypoints (PCK)

PCK measures the proportion of keypoints predicted within a specified threshold distance from the true keypoint positions.

$$\text{PCK} = \frac{100}{N} \sum_{i=1}^{N} \left[d(p_i, g_i) < \theta \right], \tag{3.8}$$

where N is the total number of keypoints, p_i is the predicted position of the i-th keypointg_i is the ground truth position of the i-th keypoint, and θ is the predefined threshold distance for correctness. Finally, $[\cdot]$ denotes the Iverson bracket, which is 1 if the condition is true and 0 otherwise.

Object Keypoint Similarity (OKS)

OKS calculates the similarity between the predicted keypoints (of a whole individual) and the ground truth keypoints, accounting for keypoint visibility and importance.

$$\text{OKS} = \exp\left(-\frac{\sum_{i=1}^{K} \frac{d_i^2}{2s^2 k_i^2}}{\sum_{i=1}^{K} [v_i > 0]} \right), \tag{3.9}$$

where K is the number of keypoints per individual, d_i is the Euclidean distance between the predicted and actual positions of the i-th keypoint. s is the scale of the individual (the area of the bounding box is the default value used for s), and k_i are per-keypoint constants modulating the impact of the distance. Last, v_i indicates whether the i-th key point is visible (1 if visible, 0 otherwise). The OKS value of an individual $\in [0, 1]$.

Recall and Precision with OKS

Precision and recall are fundamental metrics in evaluating a model's performance, including the task of HPE. Precision is the ratio of true positive predictions to the total number of predicted positives (true and false positives), measuring the accuracy of the positive predictions. On the other hand, recall is the ratio of true positive predictions to the total number of actual positives (true positives and false negatives).

Thus, recall measures the completeness of the positive predictions. Given by the following equations:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.10}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{3.11}$$

where TP are true positives, FP are false positives, TN are true negatives, and FN are false negatives. The F1-score is closely connected to precision and recall, defined as the harmonic mean of precision and recall. The F1 performance metric is especially useful when classes are imbalanced. The F1-score is given by the formula:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.12}$$

In evaluating HPE models, precision (and recall) use OKS as the threshold for categorizing true and false predictions of ground truth key points. These metrics are computed based on how well the predicted key points match the ground truth key points across varying OKS thresholds.[2]. The practice from COCO evaluation, which uses thresholds from 0.50 to 0.95 in steps of 0.05, will also be used in the badminton experiment. First, the OKS between the ground truth and predicted keypoints of each detected individual in the images are computed. For each OKS threshold, a prediction (keypoints of an individual) can be a TP, an FP, or an FN. A prediction is a **true positive** if its OKS score with a ground truth keypoint set exceeds the current threshold. A prediction is a **false positive** if it does not match any ground truth keypoint set/individual with an OKS score exceeding the threshold. Last, a **false negative** is counted when any prediction does not match a ground truth keypoint set with an OKS score exceeding the threshold. The number of TP, FP, and FN are counted for each OKS threshold t of the test data such that the precision can be calculated:

$$\text{Precision}(t) = \frac{\text{TP}(t)}{\text{TP}(t) + \text{FP}(t)} \tag{3.13}$$

The **average precision** (AP) evaluates the average of the precision values at differ-

[2]The evaluation procedure follows the recommended coco evaluation metrics:https://cocodataset.org/#keypoints-eval

ent OKS thresholds.

$$\text{AP} = \frac{1}{T} \sum_{t=1}^{T} \text{Precision}(t), \tag{3.14}$$

where T is the total number of OKS thresholds considered.

3.3.3 Badminton poses

To gauge the quality of both off-the-shelf HPE models, OpenPose [13] and HRNet [100], estimated poses on broadcasted badminton videos, the primary data video material for the application, a badminton-specific keypoint dataset is annotated. The keypoint style follows Coco [70], 62 badminton player images have been annotated across different settings and challenging human poses with occlusion and motion blur. Spectators and other non-players might sometimes be detected by pose inference. Still, the non-player detections outside the court are filtered out during evaluation using the ground plane position of the player retrieved from a homography [3] mapping of the image coordinates of the detected individual's feet. The performance is compared to the COCO validation data[4] to estimate the quality of the off-the-self HPE models on badminton data. The evaluation of the annotated keypoints for the badminton test data follows the [70] evaluation pipeline https://cocodataset.org/#keypoints-eval. The only difference between the COCO and badminton data is that in COCO validation data, all individuals in the images are evaluated, but in badminton data, spectators are filtered out, and only player keypoints are considered in the evaluation. A natural choice since only the player keypoints are used in the downstream recognition and forecasting tasks. Additionally, even occluded (non-visible) keypoints are annotated and evaluated for the badminton data. Occluded or blurry keypoints can be annotated accurately based on previous and future frames. The intention is to assess how well the HPE models can estimate even difficult poses with motion blur or occlusion and thus, how well-suited the HPE models are to create quality input data for the intended tasks.

[3] The preprocessing pipeline will be explained in greater detail in chapter 4

[4] Note that the performance score for the COCO dataset is the reported results in their respective papers.

Table 3.1: Performance Metrics for OpenPose and HRNet on Different Datasets

Dataset	Model	PCK	AP	AP50	AP75
COCO Validation	OpenPose	-	64.2%	86.2%	70.1%
	HRNet	-	81.4%	88.3%	77.7%
Badminton Data	OpenPose	57.8%	57.7%	93.6%	64.5%
	HRNet	80.7%	86.5%	98.4%	93.6%

3.3.4 Results

Table 3.1 below compares the performance of the two pretrained human pose estimation models, OpenPose and HRNet, for PCK and AP (based on OKS thresholds) on the badminton keypoint testset. The experiments show that HRnet is the superior architecture for badminton data, as it outperforms OpenPose on all evaluation metrics. Similarly, HRnet also appears to generalize quite well to the badminton scenarios, as there is an increase in performance compared to the COCO validation data. However, given that all pose candidates outside the court have been removed, the increase in performance is quite sensible.

3.4 Chapter Conclusion

In summary, this chapter introduces the two main HPE approaches, and two central model architectures, which are tested on broadcasted badminton matches with self-annotated keypoints of the players. HRNet generalized to badminton data performs better than OpenPose and will serve as the primary HPE model in the skeleton data extraction pipeline used in this work.

Chapter 4

Data pipeline & Datasets

An often overlooked aspect when assessing the capabilities of deep-learning-based models is the quality of the available data. That includes extraction and preprocessing of input features, i.e. in the case of this work, skeleton-data and shuttlecock trajectories and inspection annotated training. If these aspects are lacking, it will affect the performance of the models developed from the data. Thus, identifying limiting factors in the available training data, such as small data volume, unbalanced class distribution, or inconsistent annotations, allows to focus on methods that will mitigate the challenges and, if necessary, provide guidelines on refining data collection for future works.
This chapter focuses on these considerations and outlines the data processing pipeline for curating skeleton data and shuttlecock information used for model development. This includes the extraction process and preprocessing steps. Furthermore, the different datasets used for experiments in the remaining part of the thesis are presented. The statistics and properties of each dataset, such as the class/stroke distribution, are outlined. The datasets are finally compared to the definitions of the badminton strokes in chapter 1 to address the relevancy and potential of each dataset.
To refine and filter the skeleton poses effectively, the feet joints are mapped to the real-world court position using a homography map. A homography is a collineation that can map points from one projective space onto another and, therefore, can be used to map from the image-to-ground plane and reverse. The main principles required for projective geometry are introduced in the following section.

4.1 Homogenous Coordinates and Perspective Transfomation

Mapping between the world and the image is a necessary part of this thesis's filtering of skeleton data, and later camera models are central in the training pipeline for 3D reconstruction of skeleton and shuttle data. Specifically, the homography perspective transform basics will be covered, following [50], and in chapter 7, the pin-hole camera model is introduced.
Homogeneous coordinates is a representation used in projective geometry, which allows affine transformations to be expressed linearly. A vector, here in Cartesian coordinates, $\mathbf{P}$ can be converted to a homogeneous vector $\mathbf{P_h}$ by adding an extra dimension with the value 1. The reverse process going from $\mathbf{P_h} \rightarrow \mathbf{P}$ "removes" the last dimension of the vector and scales the other dimensions with the removed one. Below, the conversion is shown for a 3D Cartesian vector and the reverse process for a 2D homogeneous vector:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \rightarrow \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad , \quad \begin{pmatrix} X \\ Y \\ W \end{pmatrix} \rightarrow \begin{pmatrix} X/W \\ Y/W \end{pmatrix} \quad \text{for} \quad W \neq 0 \tag{4.1}$$

$$\mathbf{P} \rightarrow \mathbf{P_h} \qquad\qquad \mathbf{p}_h \rightarrow \mathbf{p}, \tag{4.2}$$

When working with homogenous coordinates, uppercase vectors are 3D, lowercase vectors are 2D, and in this case homogeneous vectors are given **h** subscript for clarity.

4.1.1 Homography

A homography, or projective transformation, is derived from the principles of projective geometry and describes how points in one plane can be transformed into points in another plane through a projective transformation. Given a point in one image plane (x, y) and its corresponding point in another image plane or surface (x', y'), the ho-

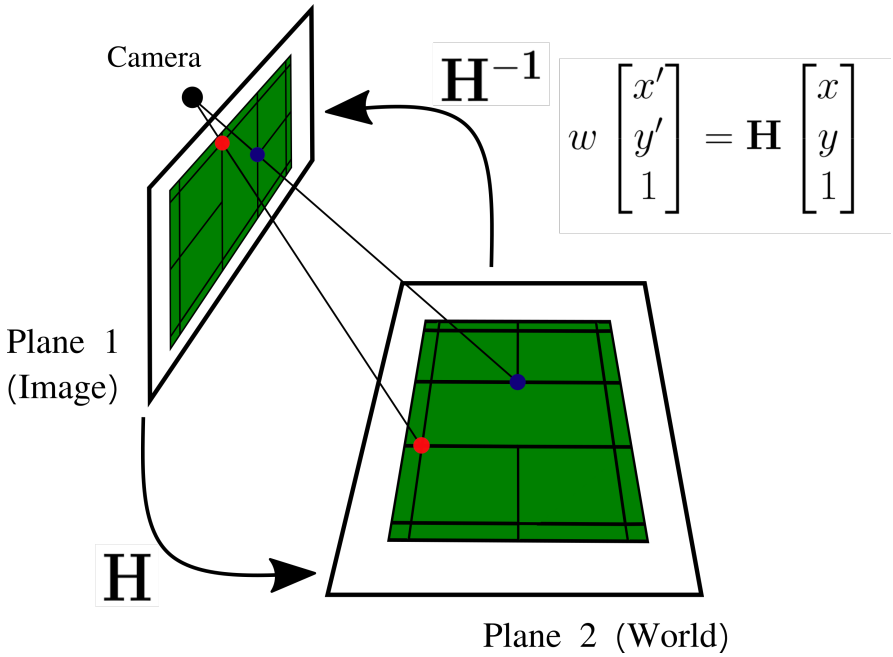


Figure 4.1: The figure visually illustrates how a homography can map points between the image and the world ground plane.

mography H maps points from one plane to another as follows:

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = \mathbf{H} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4.3}$$

where $\mathbf{H}$ is the 3x3 homography matrix with the following structure:

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{4.4}$$

The process is shown in Figure 4.1. The Cartesian coordinates of the corresponding points in the target plane can be obtained from homogeneous coordinates by scaling with w:

$$x' = \frac{x'}{w} \qquad y' = \frac{y'}{w} \tag{4.5}$$

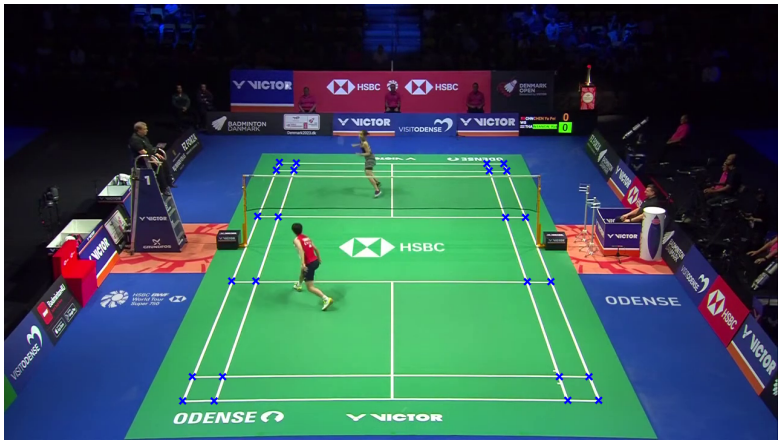


Figure 4.2: Using the known dimensions of a badminton court, a homography transformation is calculated to map image coordinates to court coordinates, facilitating the estimation of court positions and camera parameters. Corresponding points between the image and the court are used to compute the transformation.

The parameters of $\mathbf{H}$ can be estimated from several pairs of corresponding points in the two planes. A minimum of four point pairs are needed to solve for the eight unknowns in H. Specifically for badminton videos, the court's dimensions are known and can be exploited to calculate the homography for each setting. The autonomous identification of courts in badminton (among others) has been proposed in [95, 72]. The quality of estimated points is quite high, even across challenging settings, and can be used in practice. However, manually annotated corresponding points on the courts are still more accurate and reliable. The number of matches and, by extension, unique camera settings used in the thesis is $\sim$ 100, where corresponding points have manually been annotated for each match, as seen in Figure 4.2. Since the datasets used in this thesis are fully annotated broadcasted matches, the number of matches to calibrate camera parameters is manageable.

4.2 Data Features

A visual representation of the data retrieval and feature extraction pipeline is shown in Figure 4.3.

4.2.1 Overview of Input Features

Our action recognition framework utilizes multiple features extracted from video sequences to predict strokes in badminton. The primary features include:

- **Skeleton Data:** Capturing the player's body poses over time, consisting of joints, bones, and motion features.
- **Player Court Position** (G): Representing the player's location on the court in each frame.
- **Shuttlecock Position** (U): Tracking the shuttlecock's trajectory throughout the sequence.

These features collectively provide spatial and temporal information crucial for understanding player movements and actions.

4.2.2 Skeleton Data Definition

In a video sequence with T frames, the skeleton data captures the player's body pose in each frame.

Joint Positions

The pose at each frame t is represented by J keypoints (joints), each with 2D coordinates:

$$\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_T]^\top \in \mathbb{R}^{T \times J \times 2}, \tag{4.6}$$

where $\mathbf{P}_t \in \mathbb{R}^{J \times 2}$ is the set of joint positions at frame t:

$$\mathbf{P}_t = \left[\mathbf{j}_1^{(t)}, \mathbf{j}_2^{(t)}, \ldots, \mathbf{j}_J^{(t)}\right]^\top, \tag{4.7}$$

and $\mathbf{j}_i^{(t)} = \left(x_i^{(t)}, y_i^{(t)}\right)$ represents the 2D coordinates of joint i at time t. The sequence $\mathbf{P}$ captures the spatial configuration of the player's body over time.

Bone Data

Bones are vectors connecting specific pairs of joints, representing the limbs of the skeleton. Let $\mathcal{B}$ denote the set of bones, with each bone defined by a pair of joint indices (i, j). The bone vectors at frame t are computed as:

$$\mathbf{B}_t = \left[\mathbf{b}_1^{(t)}, \mathbf{b}_2^{(t)}, \ldots, \mathbf{b}_K^{(t)}\right]^\top \in \mathbb{R}^{K\times 2}, \tag{4.8}$$

where K is the number of bones, and each bone vector $\mathbf{b}_k^{(t)}$ is:

$$\mathbf{b}_k^{(t)} = \mathbf{j}_{i_k}^{(t)} - \mathbf{j}_{j_k}^{(t)} = \begin{pmatrix} x_{i_k}^{(t)} - x_{j_k}^{(t)} \\ y_{i_k}^{(t)} - y_{j_k}^{(t)} \end{pmatrix}, \quad \text{for } (i_k, j_k) \in \mathcal{B}. \tag{4.9}$$

By convention, bones are directed vectors, and their direction models the kinetic chain of a badminton stroke, starting from the feet and moving upwards, ending at the hands and head. This approach attempts to model biomechanical aspects relevant to stroke production in the Bone data (see Figure 4.4).

4.2.3 Player Court Position (G)

The player's court position represents their location on the court in each frame. It is computed as the midpoint between the left foot and right foot joints, mapped to the ground plane using a homography transformation $\mathbf{H}$:

$$\mathbf{G}_t = \mathbf{H}\left(\frac{\mathbf{j}\mathrm{lf}^{(t)} + \mathbf{j}\mathrm{rf}^{(t)}}{2}\right), \tag{4.10}$$

where $\mathbf{j}\mathrm{lf}^{(t)}$ and $\mathbf{j}\mathrm{rf}^{(t)}$ are the positions of the left and right foot joints at time t. The sequence of court positions is then:

$$\mathbf{G} = [\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_T]^\top \in \mathbb{R}^{T\times 2}. \tag{4.11}$$

4.2.4 Shuttlecock Position (U)

The shuttlecock's position provides essential context for stroke prediction. For each frame t, the shuttlecock's position is represented as:

$$\mathbf{U}_t = \left(x^{(t)}, y^{(t)}, c^{(t)}\right), \tag{4.12}$$

where $(x^{(t)}, y^{(t)})$ are the image coordinates, and $c^{(t)}$ is the confidence score of the detection. Only predictions with a confidence score above a threshold (e.g., $c^{(t)} \geq 0.75$) are considered. Positions with low confidence are padded with zeros. The shuttlecock trajectory sequence is then:

$$\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_T]^\top \in \mathbb{R}^{T\times 3}. \tag{4.13}$$

4.2.5 Feature Sequence

The comprehensive feature sequence for the model input combines the skeleton data (joints, bones), court positions, and shuttlecock positions:

$$\mathcal{F} = [\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_T]^\top, \tag{4.14}$$

where each frame's feature vector $\mathbf{F}_t$ is constructed by concatenating the following components:

$$\mathbf{F}_t = [\mathbf{P}_t, \mathbf{B}_t, \mathbf{G}_t, \mathbf{U}_t]. \tag{4.15}$$

The overall feature sequence $\mathcal{F}$ has dimensions:

$$\mathcal{F} \in \mathbb{R}^{T\times D}, \tag{4.16}$$

where D is the total dimensionality of the concatenated features for each frame.

4.2.6 Data Extraction and Preprocessing

A systematic extraction and preprocessing pipeline is used to obtain skeleton, court position, and shuttle features from raw video data.

Skeleton Pose and Court Position Extraction

A two-stage pipeline is employed for the extraction of skeleton data and court positions, using tools from [28, 18]:

1. **Human Detection:** Detect players in each frame using object detection techniques.
2. **Pose Estimation:** Estimate 2D poses of detected players using HRNet [101].

To focus on the relevant players and exclude non-participants (e.g., spectators):

- **Homography Transformation:** Calculate a homography **H** using the known dimensions of the badminton court (again see Figure 4.2 visualization of corresponding points).
- **Court Mapping:** Map detected individuals' feet positions to the ground plane to determine if they are within court boundaries.
- **Player Identification:** Identify the top and bottom players based on their court positions.

In cases where a player's feet are missing in a frame, it is replaced with the pose from the previous frame to maintain sequence continuity. The pseudo-code for the extraction process is depicted in algorithm 1.

Shuttlecock Detection

The shuttlecock's image trajectories are extracted using the pretrained models TrackNet [102] and WASB [105]. Both models detect the shuttle image coordinates in each frame but use different backbones.

TrackNet TrackNet is a deep learning model designed to detect and track high-speed, small objects – such as the shuttle – in sports videos. Utilizing a U-Net [94] CNN architecture, TrackNet processes multiple consecutive frames to generate heatmaps that pinpoint the shuttle position in each frame. This approach enables the model to learn both the visual characteristics of the shuttle and its motion patterns, retaining detection accuracy even when the ball is blurry or momentarily occluded.

WASB WASB (Widely Applicable Strong Baseline) is a model developed for ball detection and tracking across various sports, including soccer, tennis, badminton, volleyball, and basketball. WASB performs high-resolution feature extraction utilizing modules from an HRNet backbone to capture detailed spatial information. Additionally, inference incorporates multiple frames information for temporal consistency to maintain consistent ball tracking across frames, improving detection accuracy during occlusions or rapid movements.

Inference By collecting the shuttle image positions for each video frame – later split into the segmented badminton shots – the shuttle's trajectory can be determined throughout the video.
Only shuttle predictions with a confidence score above 0.75 are kept. Failed or low-confidence predictions are padded with zeros.
TrackNet detected the shuttle position in the Badminton Olympics (BadOL) and Badminton Placement (BadPL) datasets, see section 4.3. WASB outperforms TrackNet on their test set, and was hence used for shuttle detection in the more recent shuttleset and shuttleset22 datasets. WASB has a much shorter inference time than TrackNet. However, it is not apparent from visual inspection that the detection quality for WASB is superior to TrackNet for inference on in-the-wild broadcasted badminton sequences.

Normalization and Centering

To ensure consistency across different sequences and reduce variability due to camera zoom or player distance:

- **Centering:** Center each pose by subtracting the center of the player's bounding box.
- **Scaling:** Scale poses by dividing by the diagonal length of the largest player bounding box in the sequence.

Feature Computation

From the preprocessed sequences, all input features are calculated as previously defined:

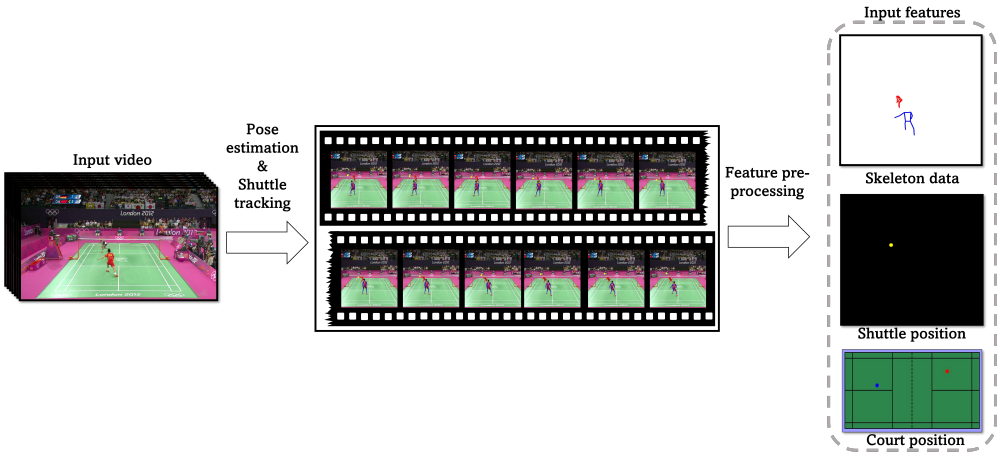


Figure 4.3: Illustration of the input data utilized by the proposed action recognition framework, *TemPose*. The framework takes in centered and normalized skeleton data of the badminton players, along with their court positions and the scaled positions of the shuttlecock, all extracted from RGB video input. Specifically, HRNet [101] estimates the players' poses, while TrackNet [102] estimates the shuttlecock's position.

1. **Bones:** Compute bone vectors using the joint positions and the defined bone pairs, ensuring bone directions align with the kinetic chain of badminton strokes.
2. **Court Positions:** Compute the player's court position using the homography transformation.
3. **Shuttlecock Positions:** Use the extracted shuttlecock positions from the pre-trained model.

Feature Concatenation

Finally, all computed features are concatenated to form the comprehensive feature sequence $\mathcal{F}$, which serves as input to our models.

4.3 Datasets

This section presents the different badminton datasets utilized for experiments during the project.

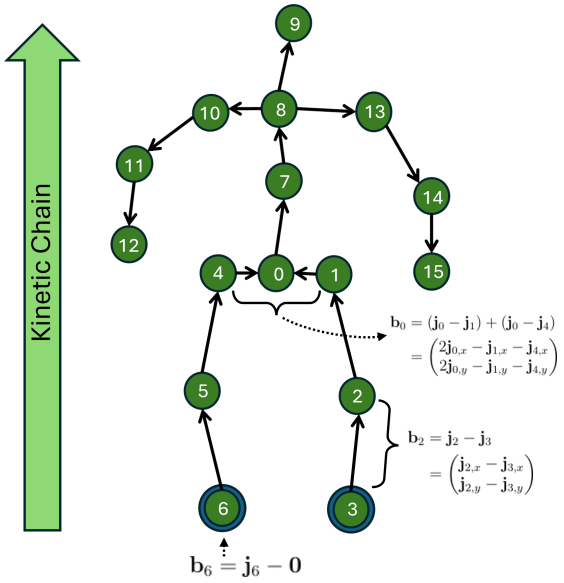


Figure 4.4: Depiction of bone conventions used in the model. Bones are directed vectors connecting specific joints, following the kinetic chain of badminton strokes. The numbering corresponds to joint indices, and arrows indicate bone directions.

4.3.1 Dataset on Badminton Stroke Placement (BadPL)

A confidential dataset provided by Badminton Danmark and TeamDanmark. The dataset contains 5566 samples of backcourt badminton strokes categorized into either attack or transport strokes. Additionally, the dataset includes information on the approximate location of the shuttlecock placement for each stroke grouped into different areas on the court: left backcourt, middle midcourt, left front court, etc. The placement annotations are quite unevenly distributed, but that primarily reflects the natural placement of backcourt shorts in professional badminton. Specific areas are rarely aimed at, e.g., the middle midcourt. As a result, the choice was made to group the labels into 12 classes for training and testing. The labels are grouped based on stroke type, horizontal placement of the shuttle (Left, Middle, Right), and courtside placement (near/far) with respect to the camera. The distribution of the stroke/placement classes is shown in Table 4.1. It can be observed that the amount of attacking/transport strokes is balanced. The amount of near/far strokes is split evenly. The horizontal placement categories are unbalanced, for example, a transport stroke is rarely placed in the middle of the court. This is disadvantageous from a

Algorithm 1: Pose Extraction and Preprocessing Algorithm

Require: Video frames $\{\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_T\}$, court dimensions
Ensure: Skeleton sequence $\mathcal{P}$
Compute homography transformation $\mathbf{H}$ using court dimensions
Initialize empty feature sequence $\mathcal{F}$
for $t = 1$ to T **do**
  Human Detection: Detect humans in frame $\mathbf{I}_t$ to obtain bounding boxes
  Pose Estimation: Estimate poses $\{\mathbf{P}_t^{(n)}\}$ for each detected individual
  for all detected individuals n **do**
    Foot Position Mapping: Compute $\mathbf{p}_{\text{foot}}^{(t,n)}$ using $\mathbf{H}$
    if $\mathbf{p}_{\text{foot}}^{(t,n)}$ within court boundaries **then**
      Assign skeleton $\mathbf{P}_t^{(n)}$ to player (top or bottom)
    end if
  end for
  if player's skeleton $\mathbf{P}_t$ is missing **then**
    Handling Missing Poses: Set $\mathbf{P}_t = \mathbf{P}_{t-1}$ (if $t > 1$)
  end if
  Normalization and Centering:
  Compute bounding box center $\mathbf{c}_t$ and diagonal length $d_{\max}$
  Center pose: $\mathbf{P}_t^{\text{centered}} = \mathbf{P}_t - \mathbf{c}_t$
  Normalize pose: $\mathbf{P}_t^{\text{normalized}} = \mathbf{P}_t^{\text{centered}} / d_{\max}$
  Append $\mathbf{P}_t$ to $\mathcal{P}$
end for
return $\mathcal{P}$

model training perspective, but it reflects interesting properties of badminton.
The train/test splitting is done cross-matches, meaning the test and training dataset contains strokes from randomly mixed matches[1]. The matches are not densely annotated, i.e., the number of strokes annotated for each match varies quite a bit. Thus, random-seed splitting for the train-test sets seemed the best option.

Advantages: BadPL consists of manual annotations made by analysts for match reports tailored for coaches and players. Hence, the annotations contain helpful information for high-level discussions and insights. A model trained on such data could provide consistent, customized information and/or analysis tailored to the needs of

[1]Seed 12 in scipy.train_test_split

Table 4.1: Bad PL Stroke Placement Distribution

Stroke Type	Placement	Count
Attack	Near/Left	718
	Near/Middle	213
	Near/Right	715
	Far/Left	703
	Far/Middle	288
	Far/Right	658
Transport	Near/Left	607
	Near/Middle	71
	Near/Right	481
	Far/Left	658
	Far/Middle	99
	Far/Right	467

the coaches and players. Additionally, annotating the stroke placement can be used to train a model to forecast future stroke placement.

Limitations: Unfortunately, BadPL has some pretty severe limitations. Only a single timestamp is annotated for each stroke, and since the dataset is not densely annotated, the duration of a stroke motion cannot be determined. Instead, a fixed sequence length for each stroke is decided (± 25 frames) around the annotated timestamp. Moreover, the timestamps for the strokes are inconsistent. Sometimes, the time stamps signify the beginning of motion for the stroke, and sometimes, the timestamp is when the player hits the stroke. The dataset can also not be applied to forecasting tasks. It does not contain continuous stroke annotation of the entire rally and, by extension, matches. Finally, only 5500 samples are provided, which is quite limited for training even moderately complex recognition models. For this reason, BadPL was only utilized for experiments in the initial part of the project at which time no alternative datasets were available.

4.3.2 Badminton Olympics (BadOL)

A badminton dataset annotated by [46]. The dataset contains badminton-related actions from broadcasted matches at the 2012 Olympics. The dataset consists of 10 videos with 15300 samples belonging to 13 different classes of badminton strokes with

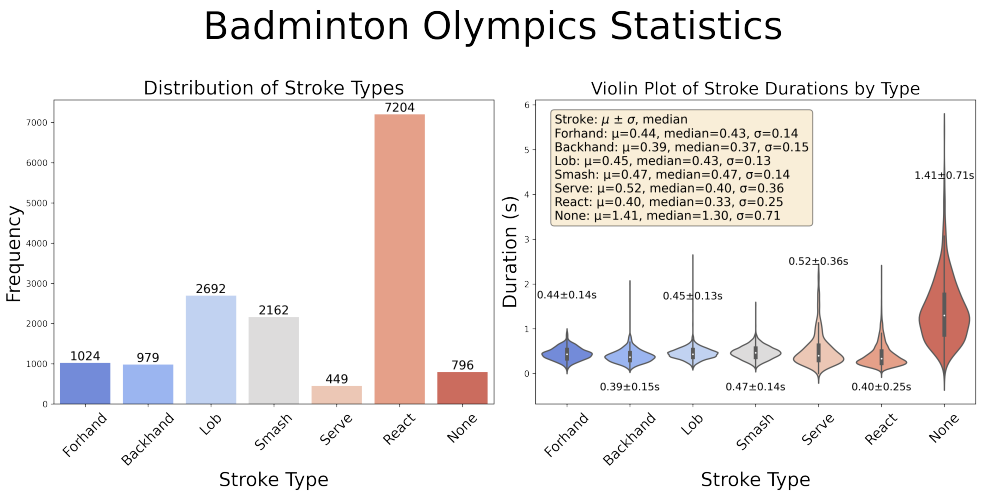


Figure 4.5: Badminton Olympics (BadOL) dataset characteristics.

the following classes: forehand, backhand, smash, lob, react motion, and a none class. This only constitutes seven different classes, but the strokes except for the None class are additionally divided into top/bottom strokes. All top strokes are performed at the side the furthest from the camera, i.e., the top of the image, and strokes closest to the camera are the bottom strokes. The react class is a unique part of the BadOL dataset. The react class corresponds to a player's motion when they react to the stroke performed by the opponent.

In the experiments for Paper 1, match splitting is used for the train/test splitting of the data. Thus, all clips from one video (match) are kept as the test set. The Test match chosen was: *Firdasari-Zaitsava-GrpO-LondonOlympics-2012.txt*. Compared to BadPL, all strokes (visible) in the 10 games are annotated, but the dataset was only used for recognition tasks.

BadOL is, in many ways, an improvement from BadPL. It contains three times more data, and the beginning and end of each class action are annotated. The dataset also has some prominent drawbacks inhibiting its usefulness for badminton stroke recognition. The None class is helpful if one wants to develop a model that can divide the segments of active rallies and inactive segments between points. Similarly, the duration of the react class can be used to model and gauge the pace of a match, i.e., long react motions mean a slow pace and short reach motions are the opposite. However, both None and React classes are not tied to a stroke motion. Thus, BadOL is more

accurately used for badminton action recognition rather than stroke recognition. The stroke types are also less detailed than other datasets in the thesis. Another effect of the less fine-grained data is that discrimination between the classes is more straightforward compared to more finely defined stroke types. Finally, the distribution of the classes and the class durations is shown with a bar and violin plot in Figure 4.5. The react class makes up half of the data samples since it is a response to a stoke class. Thus, the dataset is not particularly well balanced. The None class is also far longer than the other actions[2]. When counting just stroke classes, BadOL only has 7000 samples. For these reasons, BadOL was no longer used when improved datasets became available.

ShuttleSet + ShuttleSet22 The ShuttleSet [115] dataset contains 42 professional matches from 2018 to 2021, featuring 27 players across men's and women's singles categories. It is composed of about 3400 rallies and 34500 strokes, with an average rally length of 10 strokes. Domain experts annotated the strokes in the dataset into 10 distinct shot types: {net shot, clear, push/rush, smash, defensive shot, drive, lob, dropshot, serve, unknown/error}. The number of strokes for each type can be seen in Figure 4.6. The following year, an extension to the dataset was released called shuttleset22 [114]. Shuttleset22 has with 47 different matches – 41000 strokes and 35 different athletes – from 2021 to 2022 with an identical annotation style to shuttleset. The shuttleset datasets are better balanced than previous datasets, with reasonably detailed annotations and a significant increase in training samples.
All strokes in the given matches are annotated, making the datasets suitable for stroke recognition and forecasting. Although not perfect, the stroke timestamps are consistently annotated in the videos when the player performing the shot hits the stroke. This allows for segmenting the strokes into the split-T strokes (recall Figure 1.4 directly from the annotations. The strokes are manually converted from split-T to split-M by assuming a stroke is 70% preparation (before hitting) and 30% follow through and recovery. While this assumption cannot be perfect, it seems to work well by visual inspection.

Train-Test splitting Shuttleset [115] and the extension Shuttleset22 [114] are tested differently for recognition and forecasting tasks. For **Stroke forecating** training and

[2]However, a solution to this is to downsample the None actions

testing, the datasets are (random, seed 12) divided such that 80% of the rallies from each match are used for training, ensuring comprehensive player history, and the remaining 20% for testing.
For **Stroke recognition** a number of full matches are chosen for testing only (match-splitting)[3][4]. Seven and six test matches are chosen from shuttleset and shuttleset22, respectively. This was done in an attempt to select a nuanced collection of matches containing both women and men. The shuttleset datasets provide the best conditions for implementing badminton-oriented deep learning models.
The two shuttleset datasets are merged, particularly for pretraining in section 5.5.

BadmintonDB The BadmintonDB [8] dataset consists of 9 annotated video data professional men's singles matches. The dataset includes 811 rallies and 9,671 strokes, all featuring the players Kento Momota and Anthony Sinisuka Ginting. The dataset provides annotation dividing the strokes into 10 distinct types that follow the recommended coaches' guide of the Badminton World Federation (BWF), also shown in Table 1.1. The BadmintonDB paper suggests a specific badminton annotations procedure and tool. In the current dataset is provided as a proof-of-concept. The same two players play in all matches. Hence, the recognition and forecasting train-test splitting can be the same, since the two players are always equally represented in the train and test data. Two complete matches are reserved for testing[5], and the remaining seven are used for training. Furthermore, the shot types are almost identical to the shuttleset data, see Figure 4.7 for the stroke distribution. It shows that the distribution of shot types is strongly imbalanced. This complicates training a model on badmintonDB without overfitting on overrepresented classes. However, it also indi-

[3]Shuttleset the test matches are: ['Kento_MOMOTA_Viktor_AXELSEN_Malaysia_Masters_2020_Finals', 'Viktor_AXELSEN_CHEN_Long_Malaysia_Masters_2020_QuarterFinals', 'Anders_ANTONSEN_Jonatan_CHRISTIE Indonesia_Masters_2020_QuarterFinals', 'Carolina_Marin_An_Se_Young_TOYOTA_THAILAND_OPEN_2021_SemiFinals', 'Carolina_Marin_Supanida_Katethong_YONEX_Thailand_Open_2021_QuarterFinals', 'Viktor_Axelsen_Jonatan_Christie_YONEX_Thailand_Open_2021_QuarterFinals']

[4]Shuttleset22 test matches are:['Viktor_AXELSEN_LEE_Zii_Jia_EAST_VENTURES_Indonesia_Open_2022_Semifinal 'Viktor_AXELSEN_Rasmus_GEMKE_French_Open_2022 _Finals', 'Kento_MOMOTA_Kunlavut_VITIDSARN_Malaysia 'Carolina_MARIN_Akane_YAMAGUCHI_French_Open_2022_SemiFinals', 'An_Se_Young_Chen_Yu_Fei_PERODUA_Malaysia_Masters_2022_Finals', 'CHEN_Yu_Fei_Ratchanok_INTANON_Denmark_Open_2022_SemiFinals']

[5]*2019-43-3QF-MS-Anthony Sinisuka GINTING (INA)-Kento Momota (JPN)-YONEX French Open* and *2019-50-5F-MS-Anthony Sinisuka GINTING (INA)-Kento MOMOTA (JPN)-HSBC BWF World Tour Finals*

cates that players chose different strokes in certain matchups. On the other hand, the stroke distribution was more balanced in the shuttleset dataset, which contained more than 35 players.
The limited amount of data samples and uneven stroke type distribution makes badmintonDB well-suited for fine-tuning and testing pretrained models.

4.3.3 Evolution of Annotated Data

As seen in this section, the publicly available annotated badminton videos have changed considerably in terms of data amount, consistency, and detail of the annotations during the course of this research project. The annotated data samples for the different datasets are shown in Figure 4.8. If this evolution continues, the quality and robustness of deep-learning models for badminton analysis will certainly continue to increase.

4.3.4 Evaluation Metrics for Recognition and Forecasting

This section introduces the performance metrics used to evaluate the models for stroke recognition and stroke forecasting tasks in badminton. Specifically, multi-class *accuracy* (Acc), *top- k accuracy* (Acc-k), and the *F1 macro score* (F1-M) are employed. The metrics capture different aspects of predictive performance, to provide a nuanced assessment of model performance.
Acc provides the single most likely prediction, (Acc-k) evaluates the quality of the model alternative predictions, and F1-M provides a metric robust to potential class imbalance across multiple stroke types.

Accuracy and Top-k Accuracy Accuracy (Acc) computes the fraction of correct predictions relative to the total number of samples. Given the total number of samples N, y_i the true label for the i-th instance, and $\hat{y}_i$ the predicted label for the i-th instance. Then the accuracy is specified by:

$$\text{Acc} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(y_i = \hat{y}_i), \tag{4.17}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function, returning 1 if the condition is true and 0 otherwise.

In multi-class settings, more than one stroke choice can be plausible. Therefore, *top-k accuracy* (Acc-k) is also measured to evaluate whether the true label y_i appears among the model's top- k predictions $\hat{Y}_i^{(k)}$:

$$\text{Acc-}k = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\big(y_i \in \hat{Y}_i^{(k)}\big), \tag{4.18}$$

where k represents how many top model predictions are considered. When $k = 1$, Acc-1 reduces to standard accuracy (Acc). For larger values of k, such as Acc-2 or Acc-3, the metric reflects the frequency with which the correct stroke type is included among the top- k predictions. This is particularly useful in badminton stroke forecasting, where multiple options may be valid.

F1 Macro Score The *F1 macro score* (F1-M) addresses potential class imbalance among different stroke types by treating each stroke class equally regardless of how frequently it appears in the dataset. Given the total number of stroke types (classes) N. For each stroke type i, an F1 score, denoted $F1_i$, is computed using the precision and recall for that class (Precision_i and Recall_i) found similarly to Equation 3.12:

$$F1_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}. \tag{4.19}$$

The F1 macro score then averages the class-specific F1 values:

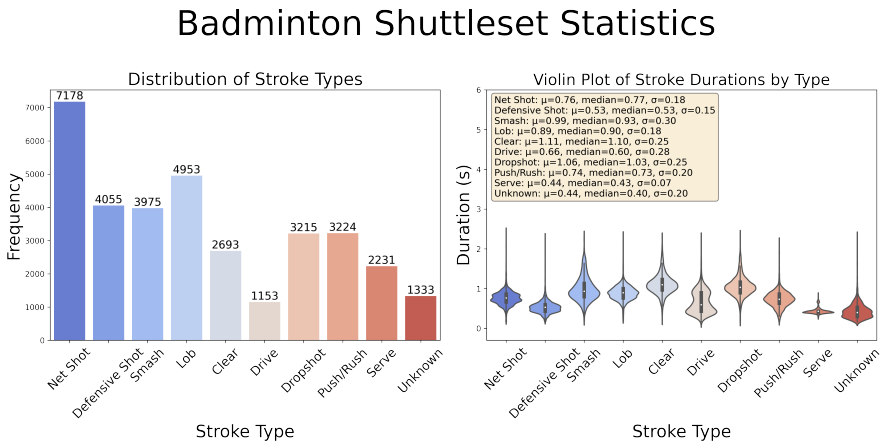
$$\text{F1-macro} = \frac{1}{N}\sum_{i=1}^{N} F1_i. \tag{4.20}$$

By averaging over all classes without weighting by class frequency, F1-macro provides a balanced view of model performance, even for an imbalanced dataset.

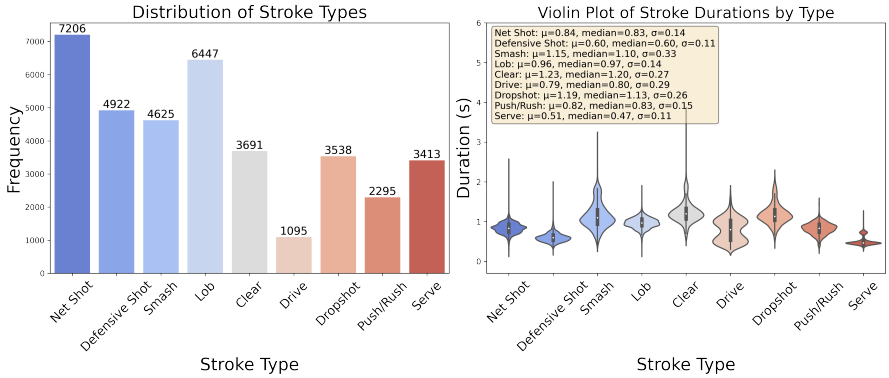
4.4 Chapter Conclusion

In the previous chapter, the extraction and processing pipeline for the features used as model input in the upcoming predictive tasks – stroke recognition (chapter 5) and stroke forecasting (chapter 6) – was introduced. The specific datasets used for recognition and forecasting were also described, and their respective train and test splits

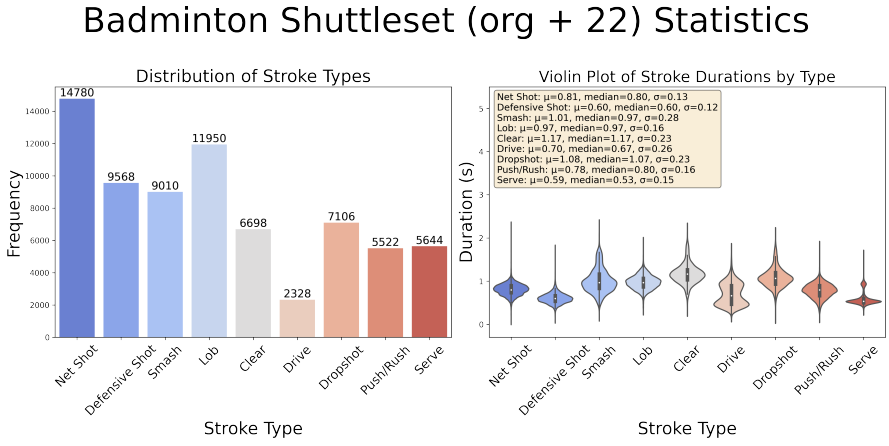
were outlined. Finally, the relevant metrics for evaluating task performance were presented.



(a) Statistics for the strokes in the shuttleset data.



(b) Statistics for the strokes in the shuttleset data.



(c) Statistics for the strokes in the shuttleset data.

Figure 4.6: Statistics for the strokes in different shuttleset dataset versions.

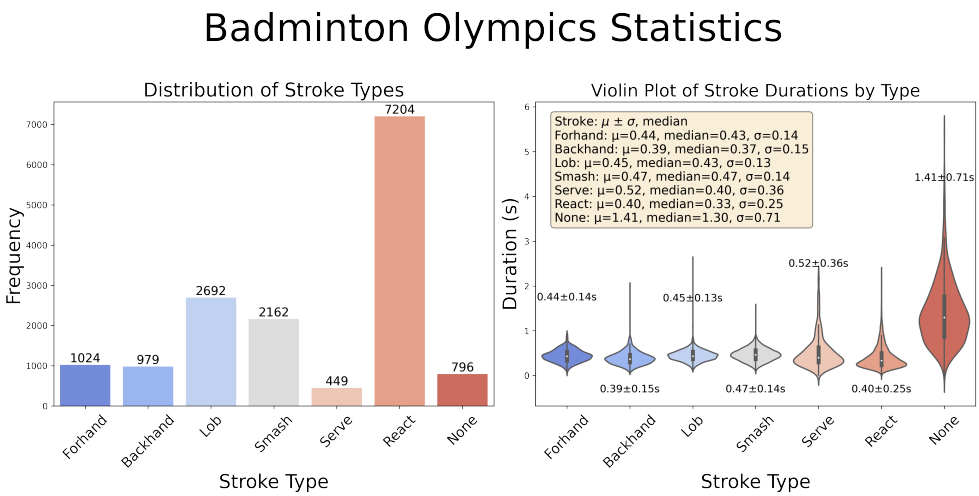


Figure 4.7: BadmintonDB (BadminDB) dataset characteristics. The left plot shows the stroke type distribution. The right figure shows violin plots that clarify how the duration of the strokes are distributed.

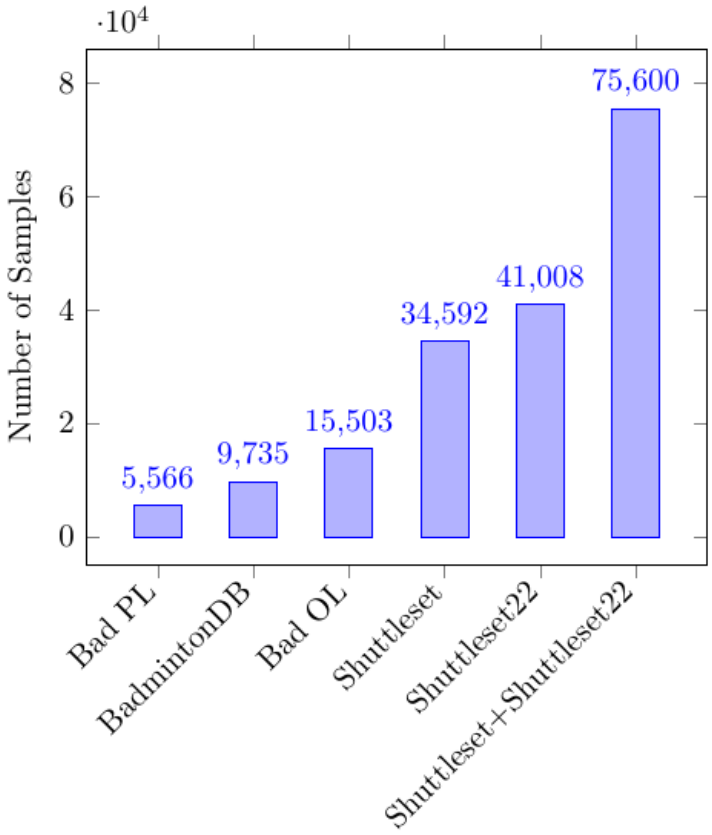


Figure 4.8: Training and Testing Sample Distribution Across Datasets

Chapter 5

Skeleton-based Stroke Recognition

This thesis aims to explore how data-driven deep-learning approaches can benefit badminton analysis. For that purpose, this chapter focuses on skeleton-based action recognition in badminton, investigating how factors such as skeleton data quality, model design, and input modality can influence the performance and generalization of stroke recognition. However, the goal is not only to design the best possible model architecture for identifying actions in badminton. The goals are:

1. To examine which factors are essential for improving stroke recognition in badminton.
2. To identify the main challenges and aspects limiting action recognition.
3. To explore solutions that address the factors limiting autonomous recognition in the field.

The content of Paper I provides the basis for this chapter, integrating its primary concepts along with supplemental experiments performed post-publication. The added content includes experiments on the Shuttleset data, a more substantial modality-ablation study, and tests of different encoder structures. Furthermore, the importance of the different phases of the stroke motion defined in chapter 1 is estimated by leaving out sections of the stroke sequences. Lastly, self-supervised pretraining methods on skeleton sequences are tested to address annotation limitations in the field.

The chapter contains the following sections:

- An experiment assessing the performance of RGB-based models against skeleton-based action recognition models using the Badminton Olympics dataset.
- A review of related works, introducing relevant skeleton-based recognition models and sports-centered research.
- An introduction to the proposed model from Paper I, TemPose, along with experiments demonstrating the performance on badminton data.
- Experiments assessing action recognition capabilities specific to badminton using skeleton and shuttlecock information.
- An exploration of a self-supervised approach to skeleton-based model learning.

5.1 Raw Video Frames vs. Skeleton Features for Stroke Recognition in Badminton

One of the main goals of the thesis is to explore capabilities and contribute towards the automatic detection of actions in badminton using deep-learning approaches. This can be considered part of the field of human action recognition (HAR).
Racket sports comprising badminton are highly technical disciplines that require high precision and accuracy in movement execution. Badminton stroke recognition is, thus, tied to the domain of fine-grained action recognition. Fine-grained action recognition deals with closely related action classes, e.g., Different types of badminton strokes. This typically coincide with the actions being tied to similar motion. Different stroke type are often differentiated by minute and subtle differences in how players execute specific movements, which is difficult to capture from raw video (RGB) data [130]. Instead, skeleton-data sequences provide a compact and background-invariant representation of human motion.
Skeleton motion as a primary feature in fine-grained action recognition has been effective in various sports disciplines [32, 65], including badminton [71, 72]. Skeleton data provide a focused representation of the body movement through spatiotemporal sequences of joint and bone positions. By capturing the detailed representation, dis-

criminating between subtle movements becomes possible, even those evading imaging techniques.
Thus, an initial experiment is performed to motivate and validate using skeleton features, and other modalities[1], which has compact discriminative information related to human motion, compared to raw image/video features. The experiment compares two baseline skeleton-based action recognition models to state-of-the-art RGB recognition models.

5.1.1 Experiment setup

The experiment was performed on the badminton Olympics (BadOL) [43] dataset, see Figure 4.5, with two models using skeleton-data and one using RGB video frame features. Thus, the initial experiment requires the preparation of RGB video snippets for each action. Recall BadOL is a dataset comprising 10 full badminton videos from the 2012 Olympics, with annotations of 7 distinct stroke types: serve, forehand, backhand, smash, net shot, react, and None. Additionally, the classes are split between the top and bottom players of the image. The 10 matches amount to approximately 15300 samples in the entire dataset. For this experiment, data train-test splitting is done across matches. The first 8 are used for training, and the remaining 2[2] are used for evaluation/testing.

Raw Video Frames

Video frames were extracted at a temporal resolution of 30fps and resized to 224x224 pixels, and pixel values were standardized using the training datasets mean and standard deviation.

Skeleton Features

The experiment is performed on two sets of skeleton data: One extracted using OpenPose (25 joints) and one set from HRnet (17 joints). All remaining preprocessing follows the preprocessing pipeline from chapter 4

[1] Specifically player Court Position (U) and Shuttle Position (G) also introduced in chapter 4.
[2] Test matches: WeiLee v Long Mens Semifinal & Yihan v Nehwal Women's semifinals.

5.1.2 Baseline Models

ViViT [4] is a factorized spatiotemporal encoder that utilizes both spatial (frame-wise) and temporal transformer blocks to recognize actions in videos. On the other hand, for skeleton-based models, two simple baselines are used to showcase the relevancy of the modality.

1. A Pose Transformer modified from ViT [34] with skeleton poses as tokens as opposed images patches. The PoT transformer uses a $D_{embed} = 128$, $N_{head} = 6$, and $L = 4$.

2. A TCN model with 2 hidden layers (refer to earlier chapter). A kernel size of $K = 5$ and a dilation of $d = [1, 2, 4]$ at layer 1 (input),2, and 3, respectively. Similarly, the channel size from each layer is set to $C_{in} = [100, 125, 150]$. The output of the final layers is flattened and passed through a two-layer MLP with GELU activation to get the stroke prediction.

5.1.3 RGN vs. Pose Results

Table 5.1 shows the performance comparison between skeleton-data and RGB data. A minimal performance boost of $\sim 2 - 3\%$ is observed across both skeleton models using HRNet skeleton data compared to OpenPose. The TCN using HRNet skeleton features reaches the highest accuracy, achieving an accuracy of around 81%, compared to ViViT, achieving a significantly lower score of 55% accuracy on video RGB input. Even when using the ViViT model pre-trained on kinetics-400 dataset [61], the RGB model fails to reach performance levels close to the skeleton-based models. Taking skeleton data as input, the TCN model outperforms the ViViT model by 26%. The experiments signify that skeleton-data models are more robust in cross-match generalization (since the train/test data is split across different matches) and perform better on the relatively limited annotated data available for badminton stroke recognition. The findings indicate that using skeleton features enhances model accuracy and lowers computational demands by reducing the dimensionality of the input space and removing redundant background information. An interpretation of the result could be that skeleton features are particularly beneficial in detailed motion-focused scenes, like sports, where condensed features provide higher informational efficiency, which is beneficial with limited training data. This is considered sufficient motivation and vali-

Table 5.1: Performance Comparison of Different Models and Feature Types

Feature Type	Model	Pose Estimator	ACC (%)	F1 (%)
Skeleton Features	PoT	OpenPose	77	70
Skeleton Features	TCN	OpenPose	80	67
Skeleton Features	PoT	HRNet	78	72
Skeleton Features	TCN	HRNet	81	73
RGB Video Frames	ViViT	N/A	45	34
RGB Video Frames	ViViT (Pre K-400)	N/A	55	50

dation for exploring the stroke recognition capabilities of deep-learning models trained on skeleton data.

5.2 Related Work

Action recognition is a central field in computer vision, and many prominent works have influenced this thesis.

Action Recognition in Sports Most work on action recognition in badminton uses convolutional neural network (CNN) architectures for feature extraction on RGB images [88, 90, 89]. Decision-making algorithms like Support Vector Machines then use the extracted features to make predictions. Other approaches involve using hand-crafted features such as Histogram of Oriented Gradients and temporal convolutional networks (TCN) to process the action's spatial and temporal aspects [46, 26]. Instead of using image data, skeleton data have been successfully used for analysis and recognition tasks in other sports, such as Tai Chi [35, 32, 113] and fencing [130, 78]. Despite its potential, skeleton poses have yet to be thoroughly tested for badminton tasks. In one recent example [72], skeleton data are used in a Gated Recurrent Unit (GRU) model to perform binary hit detection. However, like other recurrent models, GRUs can struggle with exploding gradient training issues, especially for longer sequences. While badminton is a fast-paced sport, stroke duration varies, and specific strokes like clears can take more than 2 seconds (60+ frames. TemPose from Paper 1 is a transformer model that does not suffer from the same gradient issues and there-

fore can handle sequences of flexible length.
Another interesting approach [97] uses 3D motion-captured skeleton data for binary classification of tennis strokes using a backbone spatiotemporal GCN (ST-GCN)[127] model to predict forehand or backhand strokes in badminton. While the work holds potential, the binary classification task and controlled environment for data collection limit the relevancy of their work.

Human Action Recognition Using Skeleton Data Graph Convolutional Networks (GCNs) are a popular method for skeleton-based action recognition [112, 127]. GCNs use nodes to represent every human joint in every time step. Connecting spatially and temporally to other nodes via edges allows GCNs to capture both spatial and temporal aspects of human motion.
To address the limitations of traditional GCNs, such as their bias towards very local connections, MSG3D, [74] introduces multi-scale graph convolutional networks by creating different scale adjacency matrices. This allows the network to capture multi-scale spatial relationships between joints. In this work and several others, GCN methods are often paired with Temporal Convolutional Networks (TCNs) to handle the temporal aspects of the skeleton signal. While the GCN components model the connections between joints within specific frames, the TCN looks across different frames of each joint to model motion.
CNNs are also commonly used to analyze skeleton data. One approach is to stack heatmaps along the temporal dimension into a 3D input and use 3D-CNNs to extract information [36, 14]. Other studies, such as [130, 65], generate a temporal sequence of joint coordinates and use TCNs to encode the information.

Transformers for Human Action Recognition The emergence of Vision Transformers (ViT) [34] has led to many applications of transformer backbones in computer vision [73, 9, 124, 117], including Human Action Recognition (HAR) [68, 39, 5, 73]. These works are typically trained on large datasets like Kinetics-400 [61] to mitigate overfitting issues.
Existing methods for skeleton-based action recognition achieve good results on controlled action benchmark datasets [74, 128], but many tend to lack robustness and scalability for new tasks like stroke recognition in badminton [36]. An approach to address this issue is to use more general transformer models, which have shown excel-

lent capabilities in natural language processing (NLP) [31] and image segmentation [34, 51], to model sequential data for video action recognition [73, 5, 68].
When Paper 1 was published, only a few works considered transformers on modalities other than RGB data, such as skeleton data, and to our knowledge, it had not been applied to action recognition in sports at the time. Utilizing transformer-based models on skeleton data has been attempted in other recent work. [86] combine self-attention with a GCN and TCN to model spatial and temporal attention. Similarly, [79] performs temporal encoding of the skeleton poses with a sequence of temporal transformer layers.
The combination of graph and transformer methods is becoming increasingly popular because it allows the strong benefits of graph methods to be complemented by the more general architecture of transformers. An example of this is InfoGCN [24], which utilizes intrinsic topology, i.e. the graph connections of the skeleton, in an entirely learned manner through the self-attention mechanism of the tokenized joints, multiplied by a learned adjacency matrix. InfoGCN also incorporates an information bottleneck, aiming to minimize the mutual information between the input signal and the learned representation while maximizing discriminative information in the latent representation. Like other methods, they also use TCN blocks to capture temporal aspects of the skeleton sequences.

5.3 TemPose: A Multimodal Factorized Transformer for Fine-Grained Stroke Recognition in Badminton

TemPose, a skeleton-based transformer model designed to improve fine-grained stroke recognition in badminton, was proposed in Paper 1. The model consists of a factorized transformer model that combines temporal and interaction layers, allowing for the modeling of multi-person interactions and the fusion of multi-model badminton-relevant features.
A graphical outline of TemPose, is shown in Figure 5.1. The model takes processed skeleton data as input and passes it through a sequence of transformer layers in the TemPose encoder. This process creates tokens of the temporal data and captures the temporal body dynamics and sequential interactions between an arbitrary number of people involved in the action. The MLP head at the top predicts the action based on

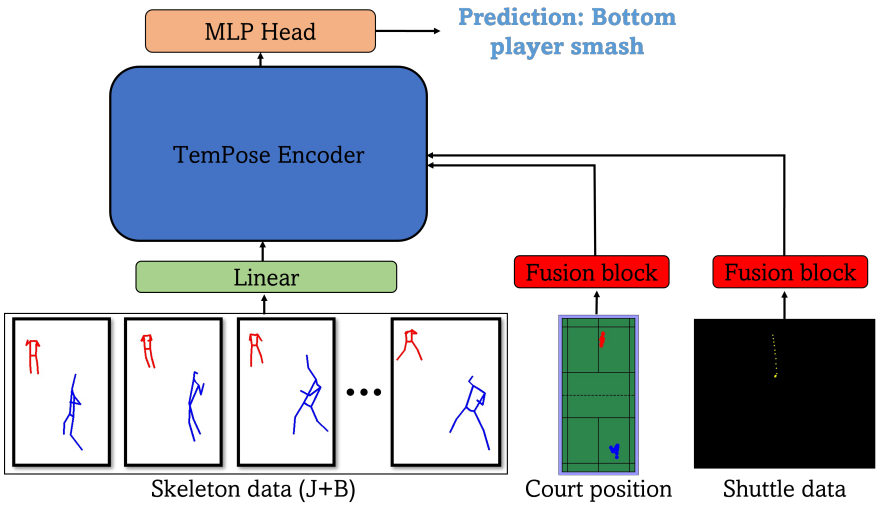
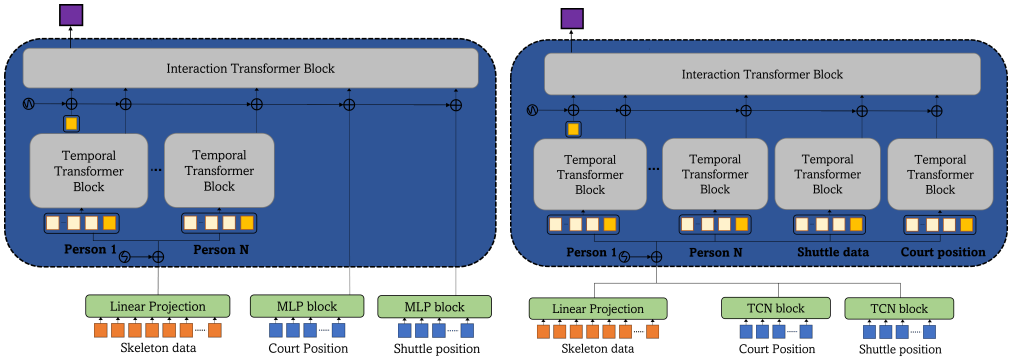


Figure 5.1: Illustration of the proposed action recognition framework, TemPose. The framework uses human skeleton data, consisting of joint and bone information, and incorporates additional features such as player court position and shuttlecock position from a badminton action (e.g., smash). The TemPose encoder, composed of multiple transformer layers, processes the input to embed relevant features into a class token. Finally, an MLP head utilizes these features to predict the action class. The composition of the MLP block is shown in the upper right corner.

the information embedded in a class token.

Figure 5.1 Shows the primary and badminton-specific version of the model, including the fusion of skeleton data with player court positions (G) and shuttlecock positions (U). Two different versions of TemPose are investigated, where the additional modalities are integrated at different stages of the TemPose encoder. The two encoder versions are shown in Figure 5.2. In one version (TemPose-NF), the G and U sequences are tokenized and appended to the embedded skeleton data before the interaction transformer layers. Figure 5.2a depicts TemPose-NF. The other version (TemPose-TF) prioritizes an early fusion of the skeleton, U, and G modalities, see Figure 5.2b for a visualization. Since the publication, many interesting encoder structures with transformers have been explored, and in the experiments section, modifications to the temporal encoder structure that combines GCN, TCN, and transformer blocks are tested.

The overview of TemPose builds upon the transformer fundamentals discussed in Chapter 2 and the data preprocessing methods detailed in Chapter 4. Initially pro-



(a) TemPose-NF: Interaction fusion encoder **(b)** TemPose-TF: Temporal fusion encoder.

Figure 5.2: The two – badminton specific – encoder version explored during the chapter.

posed in Paper 1, TemPose comprises a factorized transformer architecture that models multi-person interactions and fuses multimodal badminton-relevant features. The factorized transformer encoder combines temporal and interaction layers to capture individual and interactive dynamics in badminton strokes. An illustration of the TemPose encoder is shown in Figure 5.3. The model processes preprocessed skeleton data and, optionally, integrates player court positions (G) and shuttlecock positions (U) to enhance recognition performance.

5.3.1 Input Modalities

Skeleton Data

As detailed in Chapter 4, the skeleton data for each player in a stroke sequence includes joint coordinates and bone vectors. For a sequence with T frames, the skeleton features of a person are represented as:

$$S = [[P_1, B_1],\ [P_2, B_2],\ \ldots,\ [P_T, B_T]]^{\mathrm{T}} \in \mathbb{R}^{T\times 2(J+B)}, \tag{5.1}$$

where:

- $P_t \in \mathbb{R}^{J\times 2}$ contains the 2D coordinates $(x_i^{(t)}, y_i^{(t)})$ of the J keypoints at frame t.
- $B_t \in \mathbb{R}^{B\times 2}$ contains the bone vectors computed as $(x_i^{(t)} - x_j^{(t)},\ y_i^{(t)} - y_j^{(t)})$ for each bone connecting joints i and j.

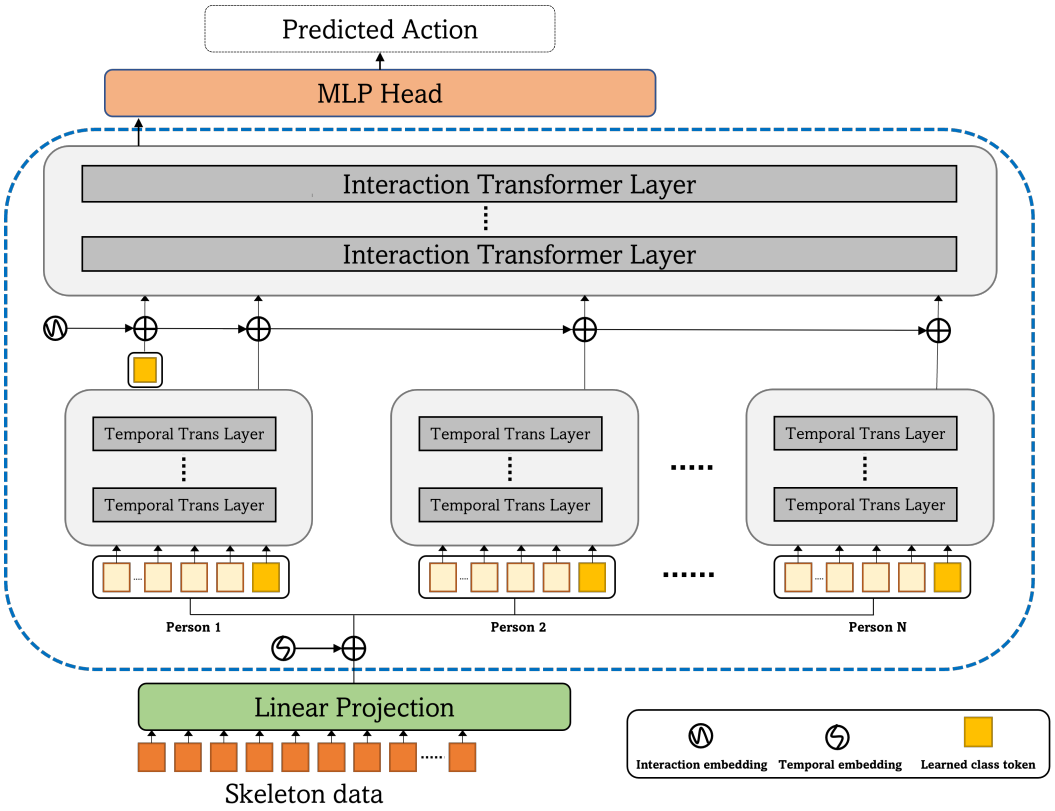


Figure 5.3: Illustration of the TemPose encoder showing the factorized transformer structure. First, temporal tokens for each person are encoded by temporal transformer layers. Second, interactions between players are modeled based on the temporal context of each person.

Including bone vectors provides additional information about the motion dynamics and kinetic chains of the athletes' motion, attempting to enhance the model's ability to recognize subtle differences in strokes.

Player Court Positions and Shuttlecock Positions

Player court positions (G) and shuttlecock positions (U) offer valuable context for distinguishing between strokes. The extraction and preprocessing of G and U data have been covered in Chapter 4. Briefly, G data captures players' 2D ground plane positions, while U data tracks the shuttlecock's image coordinates over time. See Figure 4.3 for a depiction of the features.

5.3.2 Model Architecture

Skeleton-Based Temporal Self-Attention

First, the skeleton sequence of each player is processed individually using temporal transformer layers to capture temporal body dynamics. For each player, the skeleton sequence S is mapped through a linear projection to a sequence of temporal tokens:

$$x = [x_{\text{cls}},\ \mathbf{W_e}(S)]^{\text{T}} + E_T \tag{5.2}$$

$$= [x_{\text{cls}},\ x_1,\ x_2,\ \ldots,\ x_T]^{\text{T}} + E_T \in \mathbb{R}^{(T+1)\times D_L}, \tag{5.3}$$

where $x_{\text{cls}} \in \mathbb{R}^{D_L}$ is a learned class token. D_L is the embedding dimension. $E_T \in \mathbb{R}^{(T+1)\times D_L}$ is a learnable temporal positional embedding. $\mathbf{W_e} \in \mathbb{R}^{D_L \times 2(J+B)}$ is a learned linear projection matrix mapping the input features to the embedding space. The tokens are passed through L_T temporal transformer layers, as described in Chapter 2, to model the temporal relationships within each player's movement. After processing through the temporal layers, the final representation of the class token for player n is denoted as $\tau_{\text{cls},n}^{(L_T)}$.

Factorized Temporal and Interaction Structure

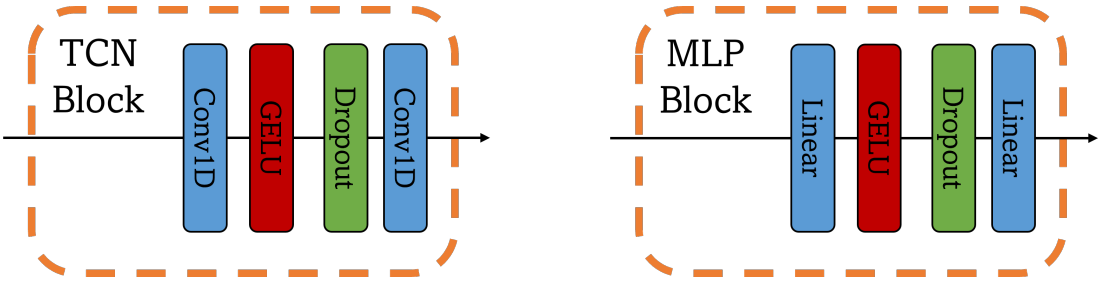
A factorized encoder structure inspired by ViViT [5] is employed to model interactions between multiple players. After obtaining the temporal class tokens for all N players, the input for the interaction encoder is constructed:

$$z = \left[\eta_{\text{cls}},\ \tau_{\text{cls},1}^{(L_T)},\ \tau_{\text{cls},2}^{(L_T)},\ \ldots,\ \tau_{\text{cls},N}^{(L_T)}\right]^{\text{T}} + E_I \in \mathbb{R}^{(N+1)\times D_L}, \tag{5.4}$$

where $\eta_{\text{cls}} \in \mathbb{R}^{D_L}$ is a learned interaction class token. $E_I \in \mathbb{R}^{(N+1)\times D_L}$ is a learnable interaction positional embedding. The tokens z are passed through L_I interaction transformer layers to capture player interactions. The final representation of the interaction class token $\eta_{\text{cls}}^{(L_I)}$ is used by an MLP head to predict the action class:

$$x_{\text{act}} = \text{MLP}\left(\eta_{\text{cls}}^{(L_I)}\right), \tag{5.5}$$

where $x_{\text{act}} \in \mathbb{R}^{D_{\text{cls}}}$ is the model's prediction, and D_{cls} is the number of action categories.



(a) TCN Block for Temporal Fusion **(b)** MLP Block for Interaction Fusion

Figure 5.4: Fusion blocks used in TemPose-NF (Interaction Fusion) and TemPose-TF (Temporal Fusion), respectively.

Integration of Position and Shuttle Data

As previously introduced, two configurations are explored for integrating G and U data into the TemPose encoder, referred to as TemPose-NF (Interaction Fusion) and TemPose-TF (Temporal Fusion). TemPose-V referes to the vanilla version of temPose with no modality fusion. In TemPose-V only the representations of the players' motions are considered in the interaction layer.

Temporal Fusion (TemPose-TF) In the TemPose-TF configuration, G and U sequences are processed through separate Temporal Convolutional Network (TCN) blocks, each consisting of two 1D convolutional layers with GELU activation and dropout, as illustrated in Figure 5.4a. The TCN blocks transform the G and U data to match the embedding dimension D_L. The outputs of the TCN blocks are treated as additional temporal tokens and appended to the transformer input along with the skeleton tokens for each player. This approach allows the temporal transformer layers to jointly model the temporal dynamics of skeleton data G and U.

Interaction Fusion (TemPose-NF) In the TemPose-NF configuration, G and U data are integrated after the temporal transformer layers. To obtain fixed-size embeddings, the G and U sequences are flattened and passed through separate MLP blocks, as shown in Figure 5.4b. These embeddings are appended to the interaction tokens z in (5.4), adding two additional tokens representing G and U in the interaction transformer layers. This approach allows the model to capture interactions between players and the additional modalities at a higher level.

Table 5.2: Hyperparameters for the TemPose training procedure. The right part of the table includes regularization and data augmentation choices.

Training	
Batch size	64
Optimizer	AdamW
Warm-up	25%
Learning rate	1e-04
LR scheduler	cosine decay

Regularization	
Label smoothing	0.1
Flipping	30%
Random shifting	30%
Dropout	0.3
Weight decay	0.01

5.3.3 Handling Variable Sequence Lengths and Number of Players

Transformers inherently handle sequences of varying lengths. In TemPose, the maximum number of temporal tokens T is fixed to correspond to the longest clip length in the dataset. For shorter sequences, the input is padded with zeros, and padding masks are used in the self-attention mechanism to ignore these positions, as discussed in Chapter 2. Similarly, an upper limit of N is set for the number of players to model interactions. Sequences with fewer players are padded with zeros, and padding masks ensure that these positions do not contribute to the attention computations in the interaction layers.

5.4 Experiments

This section presents the results of our experiments to assess the performance of the factorized transformer layers compared to other state-of-the-art skeleton-based recognition models. Specifically,TemPose is evaluated on two stroke-annotated badminton datasets [46], but contains results for two additional datasets, shuttleset and shuttleset22 [115]. Recall that all datasets were introduced and examined in section 4.3.

5.4.1 Implementation details

Table 5.2 list all settings and hyperparameters used for the training procedure of the experiments performed in Paper 1. The hyperparameters were chosen based on a heuristic hyperparameter testing. Later in the project, the hyperparameter optimization tool Optuna [3] was used to select the best hyperparameters. The AdamW optimization algorithm [76] is used for all training runs along with cosine-annealing [77]. Each training run is initialized with a sequence of warm-up epochs, slowly increas-

ing the learning rate linearly from 0 to prevent overfitting. Unless specified otherwise, joint and bone data, J and B, respectively, are used together as skeleton data input. The follow-up experiments on the shuttleset dataset utilize almost identical hyperparameters for the training procedure. However, they were subjected to slight changes in learning rate, batch size, and number of training iterations (epochs).

5.4.2 Stroke Recognition Metrics

As introduced in subsection 4.3.4, the metrics used to evaluate stroke recognition in this chapter are accuracy (Acc), top-2 accuracy (Acc-2), and the F1 macro score (F1-M).

5.4.3 Component studies

The individual components and different model configurations of TemPose were chosen based on transformer depth and configuration studies. The default model configuration uses the depth $L_T = L_N = 2$, $N_{heads} = 6$, embedded dimensions of $D_L = 100$ and $D_k = 128$, and lastly, an MLP scale factor of 4 between the input and hidden layers in MLP blocks.

Model configurations To validate the multi-modal fusion approaches of the G and U data, the performance of TemPose-V, TemPose-TF, and TemPose-NF are examined for different model settings shown in Table 5.5. AcT [79], a purely temporal skeleton-based model, is shown as the baseline model. Among the TemPose versions, TemPose-TF with $D_L = 100$ and $D_k = 128$ has the highest accuracy of 90.7% while only having 1.7 million parameters. The results suggest that temporal fusion of U and G is the best approach, achieving the highest accuracy using the fewest parameters.

5.4.4 Modality Ablation study

One small experiment shows the benefit of combining bone information with the joint data on the BadOL dataset. The results presented in Table 5.3 show the TemPose performance with only joint data input versus the combined joint and bone data. Our findings are consistent with previous studies [74, 36, 24], where bone data are shown to improve performance. The performance of TemPose significantly improves by utilizing both bone and joint data as input in this experiment.

Table 5.3: Joint + Bone architecture study. TemPose-V (Vanilla) refers to TemPose without any modality fusion. Thus, only player motion representations in the interaction encoder.

Models	Acc
Baseline (AcT[79])	
with (J)	81.8%
with (J+B)	83.7%
TemPose-V	
with (J)	81.4%
with (J+B)	85.6%

The impact of the different input modalities is examined in Table 5.4 for an ablation experiment with the TemPose-TF architecture. The experiment shows the model's performance on the shuttleset (Sset) data for the four input features. The foundational input of skeleton joint data is always included, but it shows the difference in performance when the bone, shuttle, and court position data are absent.
Not including both U and G data corresponds to TemPose without fusion as in Table 5.3. The results show that the model, used on the Sset data, does not change significantly depending on the input features. All data features offer incremental improvements to the performance. The best accuracy of 86.4% and macro F1-score 79.5% is observed when all modalities are included in the model. Apart from skeleton features, the ground position data provide the largest performance boost to the model, as the $P + G$ model performs the second best. However, the relative increase in performance is minor compared to the Joint+Bone experiment compared to the model in Table 5.3. It can be concluded that the more generally defined classes from the BadOL appear to be more dependent on the Bone features compared to stroke classes in Shuttleset.

Importance of transformer depth. A transformer depth study was performed to exhaustively test the TemPose model. Figure 5.5 shows the accuracy performance on the TemPose-TF model tested on the BadOL dataset. All other model settings, except for the number of training epochs, are kept constant throughout the study. Initial results presented in Paper 1 indicated that increasing the transformer depth beyond four layers would begin to impact the performance negatively. However, Figure 5.5 shows a more extensive depth study, where all different layer combinations are tested until $L_N = L_T = 8$. The results differ somewhat from the limited study in the

Table 5.4: Performance evaluation on the Sset dataset For TemPose-TF using different modality combinations. P: Skeleton joints , B: Bones, G: Court Position, U: Shuttle Position.

Modality				Performance	
P	B	G	U	F1-MA	Acc
✓				77.9%	85.1%
✓	✓			77.8%	85.4%
✓		✓		78.2%	85.9%
✓			✓	78.4%	85.5%
✓	✓	✓	✓	**79.5%**	**86.4%**

paper as no clear performance trend is observed.
While the best-performing combinations all have $L_N < 5$ and $L_T < 5$, the drop in performance for larger depth models is insignificant. One pattern does emerge; the increasing depth of the temporal transformer block does not affect the model significantly, but having too many layers in the interaction transformer block overall leads to worse performance. The best accuracy is achieved for the combination of $L_T = 2$ and $L_N = 2$, with a top-1 accuracy of 90.7%.
One crucial factor is that the number of epochs is adjusted for each model. Otherwise, the models with more layers in the transformer blocks would quickly overfit, leading to a drastically worse performance. However, they still perform similarly when the number of training iterations is adapted accordingly. Another observation is that variance in the performance sometimes changes 3% between similar depth combinations. The training instability can be attributed to limited training samples in the BadOL dataset. The results align well with the limitations of the BadOL dataset, which have already been discussed in section 4.3.
The results motivated the exploration of additional annotated data resources and other training approaches, such as self-supervised learning, which is included at the end of this chapter.

5.4.5 Evaluation

Stoke recognition The stoke-recognition results of TemPose compared to state-of-the-art skeleton-based recognition models are shown in Table 5.6 and Table 5.7. Table 5.6 shows the Top-1 accuracy results for TemPose-TF and TemPose-NF on the BadPL and BadOL datasets following the training/test split outlined in section 4.3,

Table 5.5: Accuracy and model size for different settings of the 3 TemPose versions. The number of attention heads $N_{heads} = 6$ and depth $L_T = L_N = 2$ are shared for all model configurations.

Model configuration	Params	Acc
Baseline (AcT[79])	2.1M	83.7%
TemPose-V		
with ($D_L = 75$, $D_k = 100$)	0.9M	85.6%
with ($D_L = 200$, $D_k = 200$)	5.2M	83.6%
TemPose-TF		
with ($D_L = 50$, $D_k = 75$)	0.5M	88.6%
with ($D_L = 100$, $D_k = 128$)	1.7M	90.7%
with ($D_L = 200$, $D_k = 256$)	6.7M	88.0%
TemPose-NF		
with ($D_L = 50$, $D_k = 75$)	2.5M	88.1%
with ($D_L = 100$, $D_k = 128$)	3.8M	89.3%
with ($D_L = 200$, $D_k = 256$)	9.0M	86.2%

Table 5.6: Top-1 accuracy results for TemPose with temporal (TF) and interaction (NF) fusion, to state-of-the-art (HAR) models on Badminton placement (BadPL) and Badminton Olympics (BadOL).

Model	BadPL	BadOL	Params
Bi-TCN [130]	80.4%	86.1%	4.1M
TCN Hog [46][3]	66.6%	77.0%	1.1M
ST-GCN [127]	72.3%	82.0%	3.4M
AcT-M [79]	77.9%	83.7%	2.1M
MS-G3D [74]	78.0%	83.2%	3.2M
TemPose-TF	83.9%	90.7%	1.7M
TemPose-NF	84.3%	89.3%	3.8M

which is the exact datasets and by extension results that were reported in the Paper 1. Overall, the results show that TemPose outperforms all other models on both datasets, with TemPose-TF achieving the highest accuracy on BadOL and TemPose-NF achieving the highest accuracy on BadPL. As observed in the configuration study, both fusion approaches achieve strong results, and no method is superior by a significant margin.

Since Paper 1, the two shuttleset datasets have been released, which contained significantly more data samples, with more detailed and more consistent stroke annotation shown in Table 5.7. Again, the TemPose model performs the best across both met-

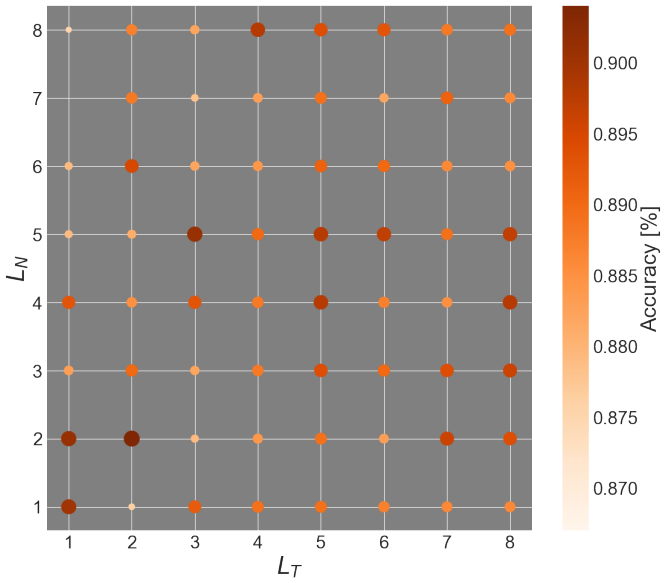


Figure 5.5: 8×8 grid showing the accuracy TemPose-TF for different depth values L_T and L_N of the interaction and temporal transformer layers, respectively.

rics, Acc and F1-M. Acc was 86.4% and F1-M 79.5%, on shuttleset and Acc 91.4% and F1-M 85.0%, on shuttleset22. All tested models are observed to achieve significantly higher evaluation scores on shuttleset22 than shuttleset. Both datasets train-test splits are across matches, defined in section 4.3. This showcases that the type and predictability of strokes vary from match to match, which resulted in the matches chosen for shuttleset22 being more straightforward to predict.
The results on the shuttleset and shuttleset22 are consistent with results from Paper 1. However, compared to the experiments on BadOL and BadPL. the gap between TemPose-TF and other state-of-the-art, models [98, 74, 22] has decreased slightly. The reason is likely that the increased dataset size has greatly benefited those models. Another possibility could be that the misclassified samples are very ambiguous or flawed, leaving the specific model choice less relevant. Still, the results support the claim that TemPose can accurately classify the different types of strokes in badminton.

Table 5.7: Evaluation metrics for various models on the sset and sset22 datasets, with match and ratio splits. The best score for each metric is highlighted in bold.

	sset		sset22	
Model	F1-M	Acc	F1-M	Acc
Bi-TCN [130]	77.0%	84.5%	83.6%	90.4%
ST-GCN [128]	77.8%	85.2%	84.2%	90.7%
AcT-M [79]	77.8%	84.2%	83.6%	90.6%
GRU [72]	76.7%	83.9%	84.0%	90.5%
CTR-GCN [22]	79.2%	86.0%	84.6%	91.1%
MotionAGFormer[128]	76.9%	84.2%	83.4%	90.5%
MS-G3D [74]	79.1%	85.7%	84.8%	91.0%
TemPose-TF	**79.5%**	**86.4%**	**85.0%**	**91.4%**

5.4.6 Importance of the phases in Strokes

One factor regarding stroke recognition in badminton that should not be overlooked is the impact the different phases of a stroke motion have on recognition performance. As stated at the beginning of the section, all recognition experiments are based on the Split-M definition of the stroke, i.e. the duration of a stroke sequence is set as the time of executing the stroke-motion. From the annotations, the motion can be divided into motion before shuttle contact (preparation + forward acceleration phases) and motion post shuttle contact (follow through + recovery phases). It is apparent from qualitative inspection that it is much harder to anticipate which stroke the player will execute, looking only at motion before they hit the shuttle. The athletes are proficient in masking/faking their intended stroke before hitting the shuttle, making it more difficult for the opponent to prepare for the incoming shot.
The TemPose-TF model has been exposed to a similar exercise. Here, the multi-modal input sequence has been divided between before the player hits the shuttle (before hit) and after the player hits the shuttle (after hit). For reference, the results from the full sequence are also provided. The results are shown in Table 5.8. TemPose-TF has a performance difference of 19% for accuracy, F1-macro, and balanced accuracy between the after-hit motion and the before-hit motion, which supports that the model experiences the same difficulties using only the motion before the hit.
Furthermore, using the full sequence input only results in slight ($\sim$ 6%) improvement to performance metrics compared to the after-hit motion sequence. This suggests the

Table 5.8: Performance of TemPose-TF on the sset22 dataset with different stroke sequence configurations: sequences cut before and after hitting the shuttle, and full sequences.

Stroke Sequence	F1-MA	Acc	Acc-2
After Hit	71.9%	80.3%	94.2%
Before Hit	55.2%	61.0%	92.0%
Full Sequence	**79.5%**	**86.4%**	**96.6%**

model can extract enough information to differentiate between the strokes from the post-hit motion, shuttle, and court position. The last observation is that the top-2 accuracy is similar for all three input sequences. An explanation for this is that in badminton, only a few shots are viable in a given situation. Thus, players will still decide between 1-3 options even when they mask their strokes, i.e., when a shuttlecock is lifted with a high trajectory toward the backcourt, only a smash, clear, or drop are viable options for the next stroke. Thus, even when a player can hide whether they intend to perform a smash or a clear, it is still apparent that it will be one of the two. The top-2 accuracy confirms that the model can infer this from the before-hit motion.

Encoder blocks testing

The TemPose-TF encoder is comprised of two main parts. First, a feature representation learning component. Second, a feature representation fusion component. The feature fusion, or mixing, happens in the interaction transformer block, where individuals' different features and motion representations are passed as tokens through a transformer block.

In TemPose, the motion representation is captured through a temporal transformer block. While the representations captured by the temporal transformer block are sufficient to outperform concurrent action recognition methods, it is conceptually a simple transformer block considering embeddings of entire skeleton poses as tokens, with no explicit spatial, i.e., skeleton structure components incorporated.

In related works, other architectures for capturing relevant motion representations were outlined, some relying on both spatial (body structure) and temporal (keypoints between time steps) using architectures integrating GCN, TCN, and transformer blocks. None of the suggested methods contest the TemPose architecture for stroke recogni-

tion. However, one could attribute this to integrating additional modalities through the interaction block, partially supported by the modality ablation study in Table 5.4. Thus, testing extensions to the temporal transformer block could yield improved recognition performance. This is accomplished by replacing the temporal transformer block with more intricate encoding modules and observing the effect. Recall the 2D skeleton-joint sequences for a single individual is given by

$$\mathbf{S} \in \mathbb{R}^{T \times J \times C_{\text{in}}},$$

where T is the Number of time frames (temporal dimension). J is the Number of joints (spatial dimension). C_{in} is the number of input channels (e.g., coordinates per joint or Bone vector etc).
Thus far, the transformer block has been operating on the sequence by projecting the combined $J \cdot C_in$ channel to D_L embedding dimension and attending to the time step tokens with $\mathbf{x}^{(0)} \in \mathbb{R}^{T \times D_L}$. In the following two alternative forms of Multi-Head Self-Attention (MHSA) are introduced:

1. **Spatial Multi-Head Self-Attention (S-MHSA)**: Models relationships among joints within the same time step.

2. **Temporal Multi-Head Self-Attention (T-MHSA)**: Models relationships of each joint across different time steps.

The key difference is that now the input features projected (or tokenized) to D_L is C_{in}, i.e, single keypoint channels and not the complete skeleton, which results in joint ST-tokens $\mathbf{x}^{(0)} \in \mathbb{R}^{T \times J \times D_L}$, i.e. rank-3 tensors now. The Positional encoding of the tokens also changes slightly, now

$$\mathbf{x}^{(0)} = \text{Linear}(\mathbf{S}) + \mathbf{P}_{\text{pos}}, \tag{5.6}$$

where $\mathbf{x}^{(0)} \in \mathbb{R}^{T \times J \times D_L}$, and $\mathbf{P}_{\text{pos}}$ represents a learned positional encoding to incorporate spatial and temporal positional information, where $\mathbf{P}_{\text{pos}} = \text{PE}_S + \text{PE}_T$. PE_T is learnable temporal parameters repeated over the joint dimension. PE_S is learnable spatial parameters repeated over the temporal dimension.
Apart from operating on a single dimension of the ST-tokens, S-MSHA, and T-MSHA are equivalent to the previously defined self-attention and multi-head attention mech-

anisms in chapter 2. The classification token becomes $x_{cls} \in \mathbb{R}^{1\times1\times D_L}$ which after the spatiotemporal transformer is passed on to the interaction transformer precisely as in the base TemPose-TF. [4]

5.4.7 Encoders

The following extended motion-encoder blocks are considered:

1. Joint Spatiotemporal (ST) Transformer, where the S-MSHA and T-MSHA are performed sequentially inside a single transformer layer.

2. Parallel Joint Spatiotemporal Transformer (ST+TS), in a single transformer block ST and TS attention attention is performed in parallel and implemented from [131].

3. Joint Spatiotemporal transformer block + with TCN-GCN block in parallel, inspired by [98].

4. Graph Transformer; In parallel, performs "Global" S-MSHA, and "Local" S-MSHA inspired by [110].

The extended motion encoders are shown in Figure 5.6.

Joint ST-Transformerblock The Joint ST-Transformer block integrates S-MHSA and T-MHSA in the transformer layer to model spatiotemporal dependencies. At each layer l, the processing flow in the encoder block is as follows:

[4]Note that tests with the average token embedding representations were performed instead of classification tokens. Like many other works, [108, 34, 4], no performance difference was observed, and as such, the x_{cls} implementation is kept throughout the experiments for simplicity.

Joint Spatiotemporal Transformer

Input: $\mathbf{x}^{(l)} \in \mathbb{R}^{T \times J \times D_L}$

Spatial Attention:

$$\mathbf{x}'^{(l+1)} = \mathbf{x}^{(l)} + \text{S-MHSA}(\text{LN}(\mathbf{x}^{(l)})), \tag{5.7}$$

$$\mathbf{x}''^{(l+1)} = \mathbf{x}'^{(l+1)} + \text{MLP}(\text{LN}(\mathbf{x}'^{(l+1)})). \tag{5.8}$$

Temporal Attention:

$$\mathbf{x}'''^{(l+1)} = \mathbf{x}''^{(l+1)} + \text{T-MHSA}(\text{LN}(\mathbf{x}''^{(l+1)})), \tag{5.9}$$

$$\mathbf{x}^{(l+1)} = \mathbf{x}'''^{(l+1)} + \text{MLP}(\text{LN}(\mathbf{x}'''^{(l+1)})). \tag{5.10}$$

Output: $\mathbf{x}^{(l+1)} \in \mathbb{R}^{T \times J \times D_L}$

The composition of the joint ST-transformer block is shown in Figure 5.6 to the far left. All attention modules contain renormalization, a residual connection and a residual MLP operation. In the following descriptions of the different extended encoder blocks, the normalization, residual connections and MLP operations are left out for simplicity since they are present and kept the same for all attention operations in the extended encoder.

Joint ST+TS transformer Joint ST refers to spatiotemporal, which means the embeddings are first subjected to spatial attention (S-MHSA) and then a temporal attention operation (T-MSHA). Similarly, Joint TS refers to temporospatial, where the embedded representation is first subjected to a temporal attention operation followed by a spatial attention operation. The Joint ST+TS transformer block follows an implementation from [131], where ST and TS are performed in parallel. The ST and TS representations are aggregated adaptively at the end of the block. This means each

representation is weighted based on the value of each representation as follows

$$\mathbf{x}^{(l+1)} = \alpha_0 \mathbf{x}_{ST}^{(l+1)} + \alpha_1 \mathbf{x}_{TS}^{(l+1)}, \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \mathbf{W}_\alpha [\mathbf{x}_{TS}^{(l+1)}, \mathbf{x}_{TS}^{(l+1)}], \tag{5.11}$$

where $\mathbf{W}_\alpha \in \mathbb{R}^{2D_L \times 2}$ is a learnable projection matrix.

Joint ST Transformer + GCN-TCN The implementation of this encoder block is heavily influenced by [98]. Similar to the Joint ST+TS encoder, this encoder has two parallel streams of embedded representations. The first stream mirrors the Joint ST encoder block, i.e., S-MSHA attention, followed by T-MSHA. In the second stream, a single layer GCN, see subsection 2.1.4, is applied to the spatial dimension of the representation (+ Batch normalization [58]), followed by a single layer TCN, see Equation 2.29, and a batch norm. The key idea is that both the single-layer GCN and TCN are locally processing the representation spatially and temporally, respectively. Thus, the GCN-TCN stream focuses on the local structure of the data, whereas S-MHSA and T-MHSA process the global structure in the first streams.

Spatial Graphformer + Temporal Transformer block This transformer block utilizes graph attention described in subsection 2.3.4 in parallel with spatial attention. In the graph attention, the adjacency matrix representing the skeleton topology is multiplied with the attention matrix to keep only local attention scores. Thus, the S-MHSA can access the global spatial structure, while the graph attention focuses on the local intrinsic topology. As depicted in Figure 5.6 (far right), the two representations are subsequently concatenated and passed to the T-MHSA section. The principle is similar to the *Joint ST Transformer + GCN-TCN* encoder. The main difference is that for Graphformer, the local and global spatial representations are concatenated before the T-MHSA. In contrast, the GCN-TCN keeps the local and global spatial representations separated until after the temporal modules (Global:T-MHSA,local:TCN), where they are subjected to adaptive aggregation. Thus, the Graph attention temporal module can access local and global considerations of the current layer.

Encoder Extension Performance The result of the experiment is shown in Table 5.9. The different choices of motion encoder do not heavily affect the model's

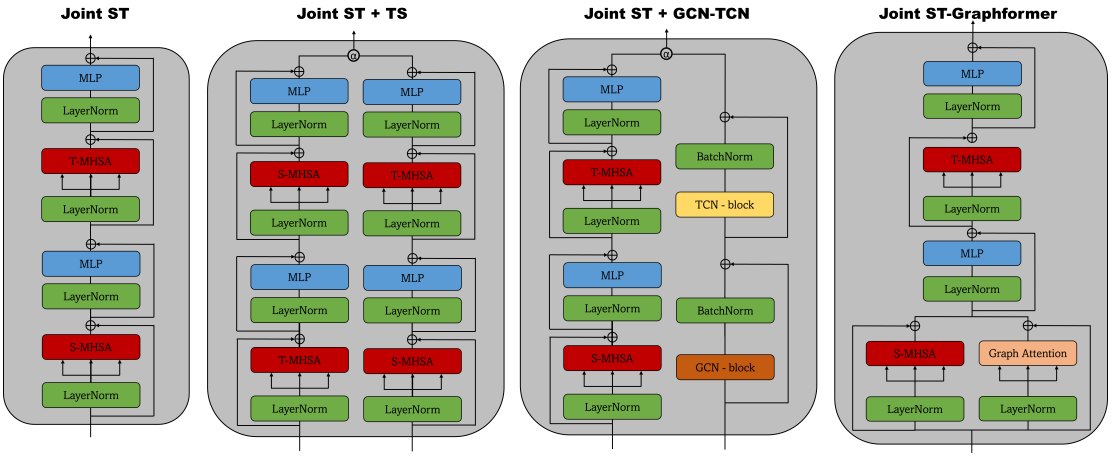


Figure 5.6: Shows the four different Joint ST transformer block testes as potential extensions to the Temporal transformer block.

Table 5.9: Performance comparison of different TemPose-TF encoder modules on the shuttleset dataset.

TemPose-TF Encoder Modules	F1-MA	Acc	Acc-2
Temporal (Base)	**78.8%**	**86.4%**	**96.6%**
Graph Transformer	76.4%	84.8%	96.0%
Joint ST Transformer	77.6%	84.7%	95.6%
Joint ST-TS Transformer	76.4%	85.0%	95.9%
Joint ST-Transformer + TCN-GCN [98]	77.5%	85.0%	90.9%

performance. The Base temporal encoder still shows the best performance, but all achieve comparable results, within $1-2\%$ accuracy of each other. However, extending the temporal encoder module with ideas from other state-of-the-art skeleton-based action recognition models does not improve performance.

The result here supports the conjecture from [129], where they argue that the overall structure of the transformer block is the most critical aspect of a transformer or "MetaFormer". The "mixing" methods of the tokens, i.e., utilizing MHSA or MLPs, are less important, and even basic mixing techniques can come close to state-of-the-art performance. The experiment suggests extending the complexity of the motion transformer block [5] does not result in measurable performance increase and thus, might be adding unnecessary complexity to the architecture.

[5] Originally, the temporal transformer block

5.4.8 Qualitative analysis of temporal and interaction attention

The attention maps of the transformer layers can be examined to inspect what information the encoder captures. The temporal attention maps of two forehand strokes shown in Figure 5.7 reveal that similar patterns emerge between actions of the same class. The similar attention maps suggest that the model has learned to focus on specific temporal aspects of the actions to predict the entire sequence.
The attention maps determine a temporal and interaction attention score for all actions. We define the attention score as the self-attention of the x_{cls}-token in the last transformer layer, aggregated and normalized across all attention heads. The temporal attention score is averaged over all individuals but weighted according to their interaction attention score.
For a badminton smash, the attention score is depicted in Figure 5.8. TemPose identifies the frames around contact with the shuttlecock as the most significant section. The red and purple text represent the target and prediction class of the action, respectively. The model accurately predicts the action as a smash from the bottom player.
Additionally, more attention is given to the smashing individual. The logical distribution of attention suggests that the model has developed the ability to gauge the relevance of each individual for the action based on their skeleton movements.
TemPose demonstrates top results on badminton action recognition tasks. However, in the experiments, the larger configurations of TemPose show signs of overfitting/unstable convergence. The result indicates that the performance of TemPose may be further improved if additional steps are taken to prevent overfitting. In general, increasing the amount of training data would be worthwhile. One prospect involves generating synthetic data to increase the amount of training data. Another possibility would be to leverage unlabelled information in the skeleton data as a pretraining task to obtain higher-quality motion representations before fine-tuning for the specific recognition task.

5.5 Skeleton-based Pretraining Methods

In this section, we explore pretraining approaches for representation learning. Self-supervised learning utilizes inherent ”unlabeled” information from the video/skeleton

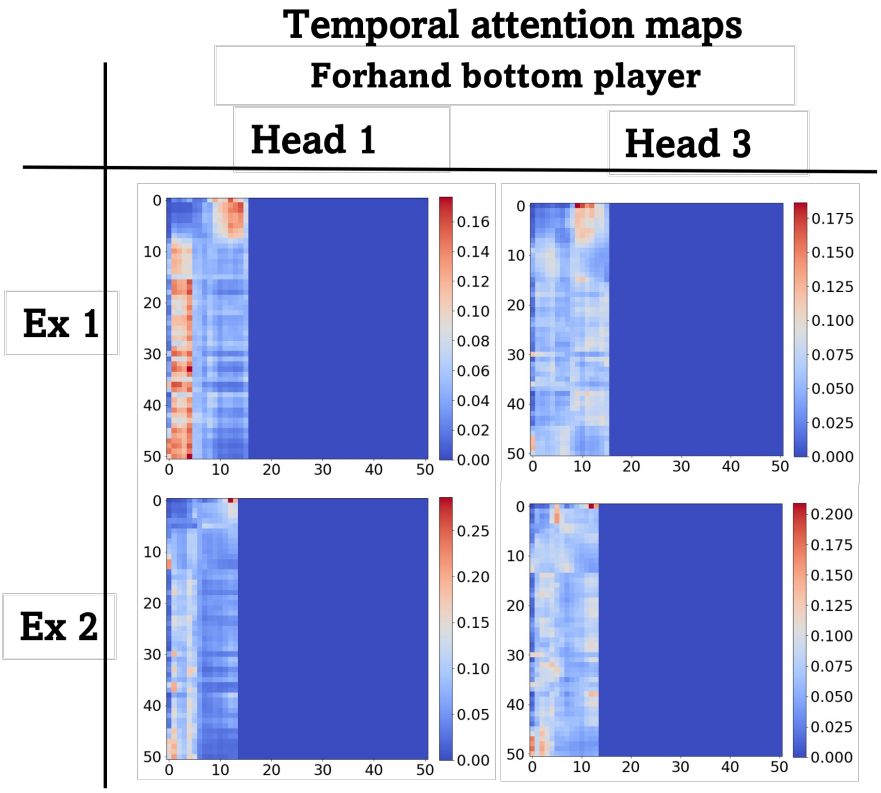


Figure 5.7: Temporal attention maps for a forehand by the bottom player (from BadOL). The distribution of attention shows that TemPose prioritizes similar information when the actions are of the same class. Additionally, the attention maps also show the effect of the padding mask. The padding tokens are given no attention.

sequences to pretrain the models, in this case TemPose-TF, before fine-tuning it on a downstream task such as stroke recognition. The limited availability[6] of high-quality annotated video material in sports presents an explicit limitation for reaching human-level recognition capabilities. However, unlabeled video material exists in abundance. The possibility of leveraging unlabeled data to achieve more robust latent representations manifesting in an incremental performance gain holds great potential due to the scale of data available in this category.

In this section, pretraining methods for the skeleton-based models are explored. First,

[6]Due to their immense value, existing high-quality datasets are proprietary in popular sports like football, badminton, and tennis with large monetary incentives.

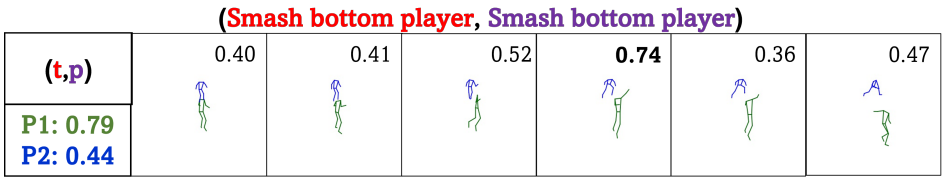


Figure 5.8: Prediction and attention score produced by TemPose-V for a skeleton sequence from Badminton Olympics. (t,p) refers to t as the target and p as the prediction. The interaction attention score is shown at the left, with the color corresponding to the person in the action sequence. The weighted temporal attention score is shown atop each frame in the sequence. For visual clarity, the frames are grouped by three, showing only the middle one, and the listed attention score is the average between them.

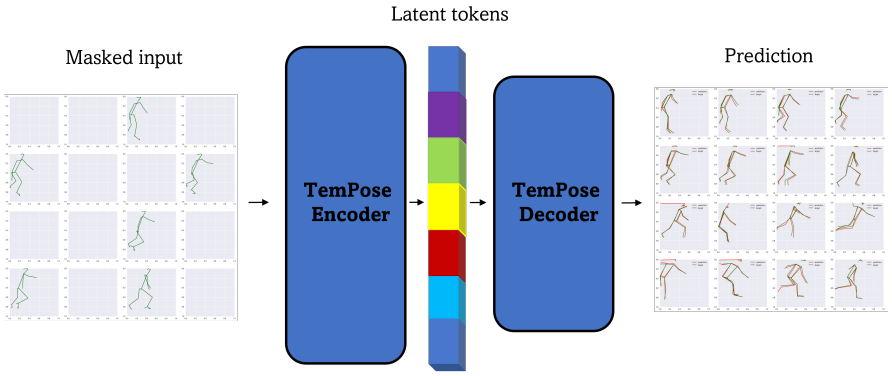


Figure 5.9: Proposed Masked Auto Encoder structure, using transformer-based encoder/decoder modules to reconstruct masked skeleton poses.

we examine self-supervised pretraining tasks to enhance performance in downstream action recognition tasks [7]. Second, the performance of the two methods is assessed by comparing the quality of trained and fine-tuned latent representations. The following pre-training approaches are considered:

1. Pretraining on different larger datasets.
2. Masked autoencoder pose reconstruction. [51]

5.5.1 Masked Autoencoder for Pretraining

The Masked Autoencoder (MAE) framework in [51] works by reconstructing an original image from a partially observed input. MAE contains the following stages: input

representation, patch masking, encoding of visible patches, decoding with mask tokens, and calculating the reconstruction loss.
The input image is divided into N non-overlapping patches. Each patch is represented as a vector in $\mathbb{R}^{C_p}$. C_p being the flattened resolution of an image patch. The collection of all patch embeddings is denoted by:

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{N \times C_p}, \tag{5.12}$$

where $\mathbf{x}_i$ represents the embedding of the i-th patch.
A subset of patches is randomly selected to be visible while the remaining patches are masked. The indices of the masked patches are denoted by $\mathcal{M} \subset \{1, 2, \ldots, N\}$, and the indices of the visible patches are $\mathcal{V} = \{1, 2, \ldots, N\} \setminus \mathcal{M}$.
The encoder processes only the visible patches. Each visible patch $\mathbf{x}_i$ is first linearly projected and combined with a positional embedding $\mathbf{p}_i$:

$$\mathbf{z}_i = \mathbf{W}_e \mathbf{x}_i + \mathbf{p}_i \quad \forall i \in \mathcal{V} \tag{5.13}$$

Here, $\mathbf{W}_e \in \mathbb{R}^{D_L \times C_p}$ is the embedding matrix that projects the patch embeddings to a latent dimension D_L. The set of embedded, visible patches is:

$$\mathbf{Z} = \{\mathbf{z}_i\}_{i \in \mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times D} \tag{5.14}$$

These embeddings are then processed through the encoder, composed of multiple Transformer blocks, to obtain the latent representation $\mathbf{H}$:

$$\mathbf{H} = \text{Encoder}(\mathbf{Z}), \tag{5.15}$$

where $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times D_L}$ captures the encoded information from the visible patches.
To reconstruct the full image, the decoder receives both the encoded visible patches and mask tokens corresponding to the masked patches. Each masked patch is represented by a shared learnable mask token $\mathbf{m} \in \mathbb{R}^D_L$. The complete set of tokens input to the decoder is:

$$\mathbf{T} = \{\mathbf{h}_i\}_{i \in \mathcal{V}} \cup \{\mathbf{m}\}_{j \in \mathcal{M}} \tag{5.16}$$

Positional embeddings $\mathbf{E}_P \in \mathbb{R}^{N \times D_L}$ are added to each token to retain spatial information:

$$\mathbf{T}_{\text{pos}} = \mathbf{T} + \mathbf{E}_{\mathbf{Pos}} \tag{5.17}$$

The combined tokens are then processed by the decoder, which is a lightweight transformer architecture:

$$\hat{\mathbf{X}} = \text{Decoder}(\mathbf{T}_{\text{pos}}) \tag{5.18}$$

The decoder outputs reconstructed patch representations $\hat{\mathbf{x}}'_i \in \mathbb{R}^{C_p}$ for each masked patch.
The final step involves mapping the decoder's output back to pixel space using a linear projection matrix $\mathbf{W}_d \in \mathbb{R}^{C_p \times D_L}$:

$$\hat{\mathbf{x}}_i = \mathbf{W}_d \hat{\mathbf{x}}'_i \quad \forall i \in \mathcal{M} \tag{5.19}$$

The reconstructed image $\hat{\mathbf{X}}$ is formed by assembling the predicted patches $\hat{\mathbf{x}}_i$ for all masked indices $i \in \mathcal{M}$.
The model is trained by minimizing the Mean Squared Error (MSE) between the reconstructed patches and the original patches, focusing solely on the masked regions:

$$\mathcal{L} = \frac{1}{|\mathcal{M}|} \sum_{j \in \mathcal{M}} \|\hat{\mathbf{x}}_j - \mathbf{x}_j\|_2^2 \tag{5.20}$$

The encoder is implemented with more layers and model parameters to process only the visible patches, while the decoder is more lightweight, handling the full image reconstruction. The design facilitates that the computational resources are predominantly allocated to the encoder, allowing for scalability and efficiency in training.
For the stroke recognition pretraining, the masked autoencoder design is repurposed for skeleton sequences, employing a similar asymmetric encoder-decoder architecture based on the temporal transformer block from the TemPose model for the skeleton-pretraining architecture.
The encoder compresses the input into a latent representation while the decoder reconstructs masked portions of the sequence. During training, masking is applied to random time steps and joints, forcing the model to learn temporal dynamics and the

intrinsic skeleton topology necessary to reconstruct the removed elements.
This has been researched in another work that have implemented similar ideas, namely, SkeletonMAE [126] in which the researchers adopt the masking and encoder module of the MAE to handle skeleton data. Instead of a transformer, they implement an asymmetric encoder-decoder GCN architecture that works on frame-wise skeleton poses in a sequence, i.e., no temporal component. Skeletons with fractions of their joints hidden are provided as input to the encoder, creating a skeleton representation. Then the decoder uses the representation to reconstruct the missing joints based on the human topology and information from the visible joints.
All work presented here is not based on this publication (with no released source code) but purely takes inspiration from MAE. Currently, the pretraining is limited to components until and including the temporal transformer block. Future work could benefit from suggesting pretraining methods for the interaction part of the encoder. Additionally, only skeleton data is considered for the pretraining experiments. Including similar approaches for shuttle trajectories and court position could also benefit the model.

5.5.2 Pretraining task

The TemPose MAE is initially trained to minimize the reconstruction loss of the masked sequence elements, given by

$$\mathcal{L} = ||x - \text{Dec}(\text{Enc}(x))||^2, \tag{5.21}$$

where x is the input sequences, and Enc and Dec are the temporal part of TemPose encoder and a Transformer-decoder, respectively (see fig.Figure 5.9).
After pretraining, the encoder is fine-tuned for action recognition on subsets of the training dataset to explore the efficacy of the pre-training method.

5.5.3 Pre-training Results

Experiments use a TemPose model architecture subjected to different training routines (with or without pretraining and fine-tuned on subsets of the BadminDB dataset). The classification accuracy and F1-macro score on the BadmintonDB dataset (small dataset) are reported for the two pretraining methods. Results are shown in Table 5.10.

Table 5.10: Accuracy (%) Across Different Pre-training Methods and Models on the BadminDB dataset after being trained on the merged SSet data.

Fine-tuning Data Usage	**25%**	**50%**	**75%**	**100%**
Fine-tuning only (on BadminDB)				
TemPose-V	57.0	61.3	62.8	63.6
Action Recognition Pre-training (AR)				
TemPose-V	61.0	63.9	63.7	65.1
Skeleton Reconstruction Pre-training (SR)				
TemPose-V	58.0	63.0	62.7	64.0

The results show that both pretraining methods improve the recognition performance on BadmintonDB after subsequent pretraining. The results further show that the supervised pretraining method, i.e., training on the combined shuttleset + shuttleset22 dataset before fine-tuning, actually leads to better performance than doing masked skeleton-reconstruction (SR) pretraining, also on shuttleset + shuttleset22, before fine-tuning. This makes good sense since the supervised pretraining dataset contains closely related action classes (stroke types), with only minor differences in stroke type definitions. However, the skeleton reconstruction methods hold much greater promise in constructing foundation models for badminton recognition. This experiment used a similar amount of data to pretrain the model, but raw broadcasted matches are abundant compared to annotated data. Thus, observing a boost in performance from (SR) pretraining on this small (compared to what is accessible) amount of skeleton sequences shows promise for future development. Another observation from this experiment is that the TemPose recognition performance on the badmintonDB dataset is lower than on the two shuttleset datasets. This could be attributed to the more detailed stroke types and lower data amount. It could also be because of the heavy imbalance of stroke classes in BadminDB.

5.6 Chapter Conclusion

This Chapter explored different areas of action recognition with a focus on the sport of badminton.
First, it was verified that for badminton recognition tasks, skeleton data outperforms RGB video as an input feature modality.

Next, from Paper 1, TemPose was introduced, a skeleton-based action recognition model that uses temporal transformer layers to capture human motion dynamics and factorized interaction transformer layers to model human interaction and multi-modality integration.

Experiments on multiple datasets shows that TemPose outperforms existing methods in recognizing fine-grained badminton strokes by fusing shuttlecock data, player court positions, and skeleton movements. The model is still slightly below human-level recognition, but there is room for improvement in multiple areas.

One possibility would be increasing the quality and amount of training data. The most easily accessible data are online broadcasted badminton videos. An experiment into different pretraining methods for stroke recognition showed that pretraining on unlabeled badminton sequences improved the performance of TemPose on BadminDB. Thus, utilizing self-supervised learning shows great promise for improvement and should be explored further.

Additionally, TemPose was used to show that using different phases of the stroke motion as input strongly affects the prediction accuracy of strokes. The follow-through motion (and shuttle) has a $\sim 20\%$ accuracy compared to the preparation and acceleration phase. This will serve as the starting point of the next chapter, where forecasting of badminton strokes will be explored.

Chapter 6

Stroke Forecasting

In this chapter, the focus shifts from action recognition to stroke forecasting in badminton, examining how future strokes can be anticipated based on the sequence of previous strokes during a rally. Unlike the previous chapter, which emphasized recognizing actions from complete sequences, this chapter explores the predictive aspects of gameplay by considering only the data available up to a certain time to predict future actions.

Stroke forecasting in badminton is challenging due to the fast-paced and detailed nature of the rallies and the probabilistic elements inherent in player decision-making. Players often have multiple viable actions at any given moment, and an interplay of physical condition, psychological state, and tactical strategies influences their choices. This unpredictability complicates predictive analytics, making it difficult for models to anticipate future strokes based solely on historical data accurately.

This chapter's main hypothesis is that stroke forecasting can be improved by designing a model that fuses visual and player-specific information into a sequential prediction framework. The model aims to capture the underlying processes behind stroke selection in racket sports by incorporating additional contextual information such as player skeleton data sequences, player identity (ID), and turn-based rally awareness.

This chapter is predominantly based on Paper II, which explores how stroke forecasting can be enhanced by conditioning the current exchange of strokes with motion and player characteristics to make more informed predictions of the next stroke in a rally. Additionally, later experiments not included in the paper are discussed, specifically addressing the two stroke definitions outlined in Chapter 1. Paper 2 primarily focused

on strokes split based on the player's motion (split-M), where stroke anticipations were provided contextual information purely from the player's motions of previous strokes. While this yields small prediction improvements, relying only on prior motion is exceedingly difficult, and results do not show significant improvements compared to sequential stroke prediction with no added context. In contrast, a method using shuttle trajectory-based stroke definitions (split-T) includes the entire preparation and acceleration phase of the next stroke in the sequence. This could be argued to resemble an action recognition task rather than forecasting, but this stroke definition actually quite well reflects real-match conditions for players. Players must anticipate their opponent's stroke selection based on information available until shuttlecock contact. Thus, this approach investigates how well a player's strokes can be predicted in a realistic match setting.
One strategy to reduce the uncertainty associated with subsequent stroke predictions involves incorporating various contextual factors into the model:

Player skeleton data sequences: These sequences contain the movements and positions of a player's joints over time and have shown promising results in general action recognition [127, 79, 86, 49] and sports applications [32, 130, 72, 65]. By revealing patterns in a player's technique and movement, this data can provide information about future strokes, allowing models to account for individual players' physical capabilities and limitations.

Player identification (ID): Incorporating player IDs enables predictive models to consider historical performance data and personal playing styles. Recognizing that each player has unique characteristics, e.g., strengths, weaknesses, and strategic preferences, can help the model better predict a player's likely actions in various game situations.

Turn-based rally awareness: This introduces an additional contextual layer by specifying the actor and reactor behind each stroke, partially inspired by [116]. Including turn-based nuances allows the model to isolate individual player motions and obtain a clearer representation of each stroke.
The architecture proposed in this chapter builds upon concepts from the previous chapter to present a transformer-based model for action forecasting in badminton.

The primary contributions of this research include:

1. Developing a skeleton-based spatiotemporal encoder that uses transformer and pooling blocks to learn representations for enhancing next-stroke predictions in badminton.

2. Introducing an adaptive cross-attention decoder that incorporates contextual stroke descriptors from high-dimensional embeddings of a pre-trained language model.

3. Demonstrating how the latent variables can be used for match and playstyle analysis, providing insights into athletes' shot selection and aiding strategic adjustments in competitive settings.

4. Implementing stroke embeddings from text embeddings of detailed stroke description using a pretrained language model.

The chapter is organized as follows: Section 6.1 outlines related research on forecasting models and probabilistic modeling that has both inspired and contributed to the development of the work in this chapter. Section 6.3 details the methodology behind the proposed forecasting approach, including the architecture of the transformer-based model. Section 6.4 presents the findings from various experimental evaluations, including experiments not included in Paper II that address the different stroke definitions. Finally, Section 6.5 discusses prospects for how the forecasting model could be used for match preparation and directions for future research.

6.1 Related Work

6.1.1 Action forecasting

Prior works have attempted to develop a wide range of neural network models to forecast future action sequences from observed action labels or extracted video features. The paper [40] introduced a method using a recurrent neural network (RNN) - hidden Markov model to classify actions from video frames, followed by a convolutional neural network (CNN) or RNN that predicted the following actions in the sequence.

In [1], one-hot encodings of the action labels were processed by a Gated Recurrent Unit (GRU) to estimate the probability of the next action in the sequence and subsequently predict the duration of said action. In [75], they expanded on this and employed a variational multi-headed GRU to predict future actions and their duration. They showed that their approach worked for both one-hot action labels and extracted video features.
In [44], they suggested jointly using both frame and annotation features to improve the prediction capacity of their model. [83] employed sequence-to-sequence models using a gated recurrent unit (GRU) encoder-decoder architecture to predict future actions from RGB frames alone.
Recently, transformer architecture has been used for this task. In *Uncertainty-aware Action Decoupling Transformer for Action Anticipation* [48], the authors used decoupled transformers to separately predict nouns and verbs of future human actions in a video. Moreover uncertainty-based fusing of the verb and noun predictions is used to predict the complete action.
Skeleton data are not commonly used as a modality for action sequence forecasting. However, a recent publication, called *InfoGCN++* [25] extended their action recognition architecture to first process a given skeleton sequence. The representation of the skeleton sequence is then evolved in time using a neuralODE[1] [19] to predict the future skeleton motion. Subsequently, the current and future skeleton motion is used to predict the action of the sequence.
This area of research, called early action prediction, is closely related to action anticipation. Unlike in action forecasting, where the anticipated action is entirely unobserved, early action prediction has some incomplete information about the actions. This distinction is similar to the separation between split-M and split-T forecasting outlined for stroke forecasting in this thesis. Early action recognition has been attempted for both video [99, 118] and skeleton-based models [67, 62]. With the different splitting of the input data, this forecasting chapter explores the boundary between the two domains of action anticipation and early action prediction.
Lastly, a skeleton data-oriented area that has seen attention recently is motion prediction, where action labels are used to condition observed skeleton sequences to gen-

[1]NeuralODEs perform standard iterative numerical integration for each timestep to solve a differential equation, but initial values (state), computed by a neural architecture usually, the network can be optimized at arbitrary timesteps.

erate subsequent motion of the sequences. [85, 47] employs variational autoencoders for this task. Both transformer and TCN-GCN architectures are actively being used to model the prediction of unobserved skeleton sequences. [66] introduces SPGSN, a cascaded multi-part graph scattering block to adaptively model diverse body parts. Whereas [54] utilizes an encoder-decoder architecture with spatial and temporal TCN-GCN layers to capture motion representation to predict future motion.
Similarly, [104] introduces a model that combines Recurrent Neural Networks (RNNs) with attention mechanisms to capture spatial and temporal dynamics in human motion, which are then used to predict future motion. [60] performs multi-agent forecasting using an interaction-aware transformer-based architecture Trajectory2Pose. The architecture first predicts the global trajectories of all agents and then predicts individual poses conditioned on the trajectories. [66] introduces SPGSN, a cascaded multi-part graph scattering block to adaptively model diverse body parts.

6.1.2 Data analytical sports applications

Action recognition tasks fill up the majority of sports focused research in the field of computer vision. Here, convolutional neural networks (CNN) have been used for feature extraction on RGB images [88]. Classification algorithms such as Support Vector Machines then use the extracted features to make predictions. Transformer models have also gained traction for sports application tasks. In [16], a Vision Transformer (ViT) [34] is used as the backbone to do group activity recognition (GAR) in Volleyball and basketball.
Skeleton data, as opposed to image data, has proven effective for the analysis and recognition of activities in various sports, including Tai Chi [35, 32, 113] and fencing [130, 78]. Skeleton-based Temporal convolutional networks (TCN) have seen use for action recognition in table tennis [65], where TCNs performed better than LSTM models.
In badminton, [72] used skeleton data and shuttle trajectory data in a GRU model to perform binary hit detection. They further improved the detection rate by using badminton-specific rules. Specifically for stroke prediction [116] employed a transformer-based player and position-aware model that used prior stroke types and shuttle placement to predict future position and type of strokes. Instead of the shuttle placement, this work uses the players' skeleton and ground motion to provide a dynamic under-

standing of each stroke as the basis for predicting the subsequent strokes in the sequence.

6.2 Forecating task formulation

In action forecasting for racket sports such as badminton, the strokes are the central actions. A stroke is the motion of a player preparing to hit the shuttle until shortly after contact between the racket and the shuttle. The exchange of strokes between players is called a rally, and it continues until one player fails to return the opposing player's stroke. The objective is to predict the next stroke within a rally based on previously executed strokes while also considering the actual motion of players by incorporating 2D skeleton pose data.
A rally R is denoted $R = [s_1, \ldots, s_N]$, where s_i is the ith stroke within the rally. Each stroke is described by a sequence of skeleton data $S_1^{(i)}, \ldots, S_T^{(i)}$, with T representing the duration of a stroke sequence and N the number of strokes in a rally.
A player skeleton $S_j^{(i)}$ within a stroke s_i (ith stroke in the sequence) captures the spatial configuration of the player's body at a given time frame j, represented by a set of key points that denote the 2D image positions of the body joints. Additionally, the sequence $G = G_1^{(i)}, .., G_j^{(i)}, ..G_T^{(i)}$, representing the 2D positions of the players' feet[2] on the ground plane for each frame, is sampled and structured as $G_j^{(i)} \in \mathbb{R}^{T\times 2}$, as an additional data source.
The goal is to predict the subsequent stroke s_{i+1} in the rally sequence R and show that leveraging both the historical sequence of strokes and motion provided by the 2D skeleton poses improves the prediction rate. The input data structure with the segmented skeleton sequences for each stroke in the rally are shown in Figure 6.1.
The forecasting task is additionally split into Split-M and Split-T forecasting. In split-M, the feature sequence is cut off after the previous player's motion of the current stroke is done. On the contrary, for split-T strokes, the motion (and shuttle trajectory) is cut off right before a player hits the shuttle for the next stroke s_{t+1}. Recall, Figure 1.4 depicts the stroke definitions.

[2] Average of left and right foot position

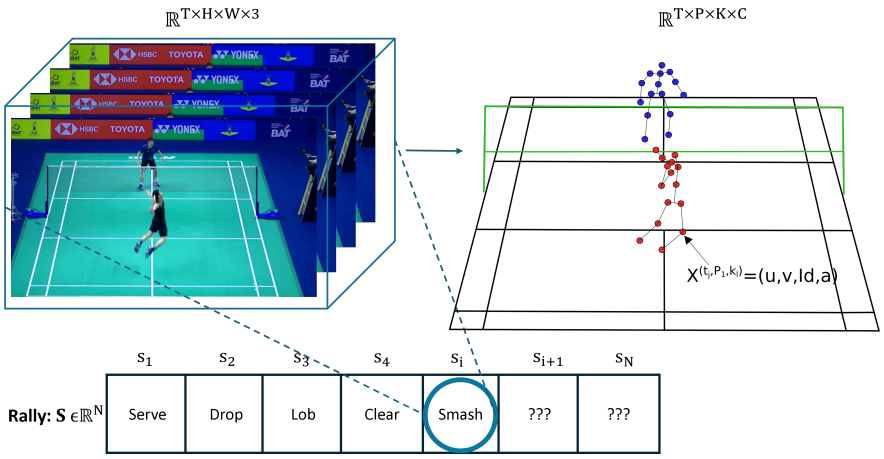


Figure 6.1: Data structure overview: Each stroke/action in a rally, i.e., stroke sequence, is provided the skeleton motion sequence of the stroke for additional context.

6.3 RallyTemPose

This section presents the causal stroke prediction model, *RallyTemPose*. The primary contribution of the model is an encoder-decoder architecture that leverages skeleton data and player court position to predict the next stroke in a badminton rally. The encoder computes an embedded representation that conditions the rally sequence, enabling the prediction of the subsequent stroke:

$$p(s_{i+1} \mid s_i, K_{1:i}, G_{1:i}, Id) = \mathrm{Dec}(s_{1:i}, \mathrm{Enc}(S_{1:i}, G_{1:i}, Id)), \tag{6.1}$$

where $s_{1:i}$ denotes the sequence of strokes up to time i, $S_{1:i}$ represents the skeleton seqeunce from the initial timestep to timestep i. Likewise, $G_{1:i}$ is the player ground position data, and Id is a player identification index. Thus, the encoder module Enc captures the representation of the stroke motion and player identifiers, which the decoder is tasked with fusion with the embedded stroke sequence $s_{1:i}$ to predict the next stroke s_{i+1}. An overview of *RallyTemPose* architecture is shown in Figure 6.2.

6.3.1 Encoder

The encoder processes raw data frames of player positions and skeleton poses to produce embedded representations for stroke prediction. It consists of several key components. First, a linear projection layer embeds the raw skeleton poses and player posi-

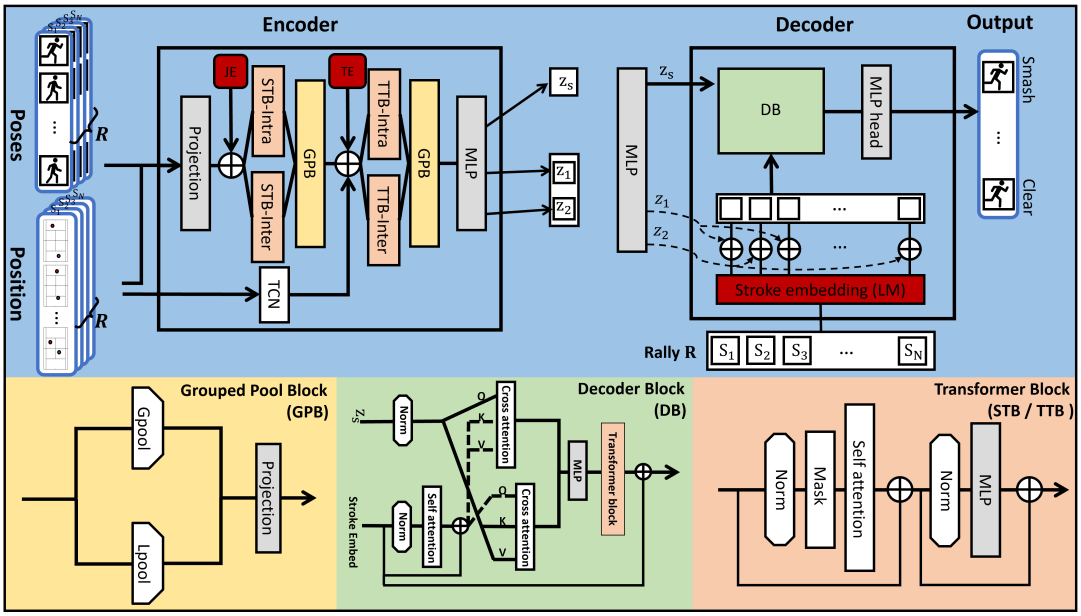


Figure 6.2: Overview of RallyTemPose architecture with individual model components.

The abbreviations refer to the following: JE: learned joint Encoding added to each pose keypoint, TE: learned Temporal Encoding added to the frame level tokens in a stroke, STB: Spatial Transformer Block, TTB: Temporal Transformer Block, GPB: Group Pooling Block, MLP: Multi-Layer Perceptron, TCN: Temporal convolutional Network smoothing over the player ground positions, DB: decoder block, LM: Language Model.

tions into tokens suitable for transformer processing. A learnable joint encoding (JE) is added to the tokenized data to provide information about the joint arrangement in the skeleton data.

Next, the **Spatial Transformer Block** (STB) applies a pose-wise transformer mechanism focusing on spatial relationships between keypoints in the players' movements. The STB captures spatial dependencies within each player's pose at a single time frame. This block follows the standard transformer architecture introduced in section 2.3, consisting of multi-head self-attention (MHSA), layer normalization, and feed-forward networks. However, it is adapted to include self-attention (intra-player attention) and cross-attention (inter-player attention) to capture interactions between players and individual movements.

Following the STB, a **Grouped Pooling Block** (GPB) aggregates information for a tensor $X \in \mathbb{R}^{G \times N \times D}$ and emphasizes useful global features of X while maintaining relevant local features of the local groups in the N-dimension. The GPB operates by

performing global and local max pooling over the embedded features, allowing the model to pick out the most important features globally and locally within each group. Specifically, operating on an embedded tensor $X \in \mathbb{R}^{G\times N\times D}$, where G is the group size, N is the number of groups (e.g., keypoints or time frames), and D is the feature dimension, the GPB performs global max pooling:

$$M_d = \text{Gpool}(X)_d = \max_{n,g} X_{n,g,d}, \tag{6.2}$$

and local max pooling within each group:

$$Q_{n,d} = \text{Lpool}(X)_{n,d} = \max_{g} X_{n,g,d}. \tag{6.3}$$

The locally pooled features Q are concatenated with the globally pooled features M (expanded to match dimensions), resulting in a tensor $Y \in \mathbb{R}^{N\times 2D}$:

$$Y_{n,d} = \text{Concat}[Q_{n,d}, M_d]. \tag{6.4}$$

In the end, a linear weight matrix layer maps $Y \in \mathbb{R}^{N\times 2D}$ to the feature dimension D.
The encoder continues with the **Temporal Transformer Block** (TTB), which focuses on temporal dynamics by processing the sequence of poses over time. Like the STB, the TTB captures temporal dependencies across multiple time frames for each player and incorporates self-attention and cross-attention mechanisms to model interactions over time.
After the TTB, another GPB pools over the embedded temporal representation. The encoder outputs three latent variables (see Figure 6.2): a stroke representation z_s and player-specific representations z_1 and z_2. The stroke representation z_s merges information from both players for each stroke, providing complete context, while z_1 and z_2 focus on individual players.
In both the STB and TTB, the block compute both inter-player attention (cross-attention) and intra-player attention (self-attention), enabling the model to capture interactions between players as well as individual movements. Figure 6.3 illustrates the different types of attention in the encoder module. The stroke representation z_s receives a concatenated array of both player representations, then focused through GBP.

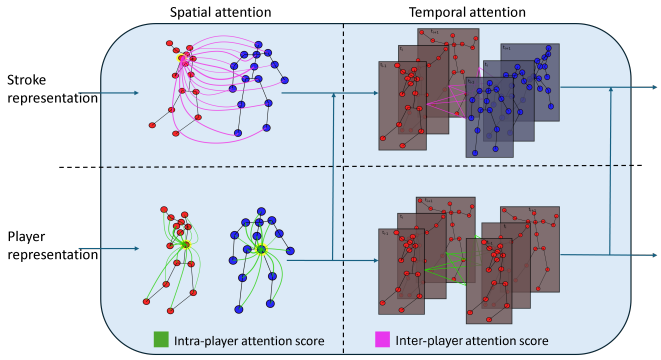


Figure 6.3: Illustration of the different types of attention present in the encoder module.

6.3.2 Decoder

The decoder predicts the next stroke in the sequence using the encoded representations. It incorporates modifications beyond the standard transformer decoder architecture to suit the specific requirements of stroke prediction in badminton. An embedding layer maps the one-hot encoded stroke sequences into stroke tokens. Exploiting the turn-based nature of badminton, the player-specific representation (z_1 or z_2) of the player performing the current stroke is added to the corresponding stroke token. The decoder block then processes this enriched sequence.

The **Decoder Block** (DB) combines self-attention, dual cross-attention, and an adaptive fusion mechanism, extending on the transformer decoder structure from [108] outlined in Chapter 2.3. The block begins by applying masked multi-head self-attention to the embedded stroke sequence, ensuring that the model attends only to previous strokes and maintains causality.

Following self-attention, the DB employs two forms of cross-attention in parallel:

Encoder-to-Decoder Cross-Attention: This module performs cross-attention using the stroke representation z_s as Key (K) and Value (V) while using the a linear transformation of the Stroke type embeddings as Query (Q).

Decoder-to-Encoder Reverse Cross-Attention: This mechanism utilizes the reverse procedure as the Encoder-Decoder cross attention. Here the representation of the stroke embeddings are used for computing the K and V, while Q is determined based on z_s.

Subsequently, an adaptive fusion layer linearly combines the outputs of these dual

cross-attention mechanisms, effectively merging the different sources of information. This fusion allows for capturing the dependencies between past strokes, player movements, and the context provided by the encoder. An additional transformer layer processes the output of the fusion layer to refine the combined representation further. The transformer block includes the standard feed-forward network (MLP) with GELU activations [53], layer normalization, and residual connections, following Figure 2.4. The decoder's architecture is tailored to enhance the integration of spatial and temporal context from the encoder with the sequence of past strokes, improving the prediction of the next stroke. By incorporating player-specific representations and modified attention mechanisms, the decoder effectively models the dynamics of badminton rallies.

6.3.3 Enhanced Stroke Embeddings

Another aspect of the model is using a pretrained language model to provide richer representations of stroke types. Specifically, BERT [31] is used to generate stroke embeddings. Each stroke type is annotated with a textual description of its characteristics and typical use cases. These descriptions are processed through BERT to extract high-dimensional embeddings from its latent layers.
Figure 6.4 shows the three types of stroke-embeddings explored in this work. The language model-derived embeddings offer more detailed representations than those learned from the comparatively smaller badminton datasets for training the prediction model. Incorporating these enhanced stroke embeddings allows the model to leverage semantic information about the strokes, potentially improving its ability to generalize and adapt to strokes with similar characteristics.

6.3.4 Training

The design of a transformer allows for $N - 1$ training samples to be created from a N stroke rally S. In each training sample, the last stroke functions as the prediction target of the model, while all prior strokes in the rally serve as the observed sequence. This strategy allows for variable-length training sequences, allowing the model to observe the connection between all possible strokes in a rally during training. Sequence diversity helps the model avoid overfitting. The network is trained using two loss functions.

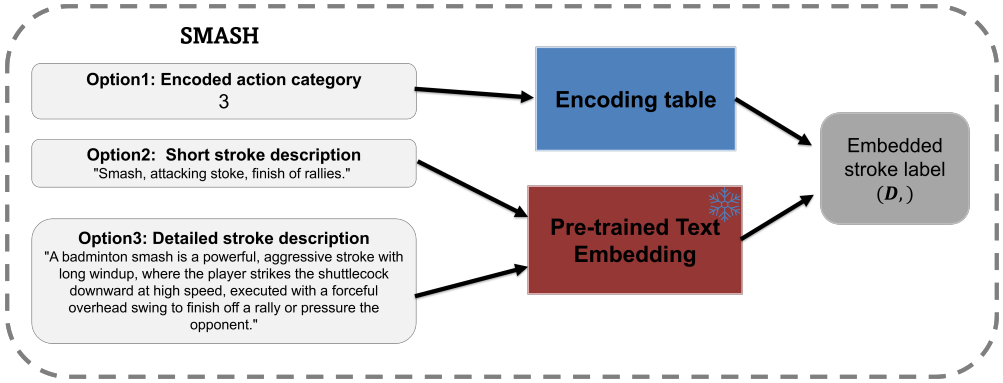


Figure 6.4: The 3 different stroke embeddings explored for the forecasting model. 1) A learned embedding table for the different stroke classes, 2) Pre-trained text-embeddings of very brief stroke descriptions. 3) Pre-trained text embeddings of detailed stroke descriptions.

First, the cross-entropy loss is minimized between the target and predicted strokes:

$$\mathcal{L}_{main}(s_{i+1}, \hat{p}_{i+1}) = -\sum_{j=1}^{C} s_{i+1}^{(j)} \log(\hat{p}_{i+1}^{(j)}), \tag{6.5}$$

where C is the number of stroke classes, s_{i+1} is the one-hot encoded target stroke, and $\hat{p}_{i+1}$ is the predicted stroke type probability vector.
Second, an auxiliary objective is defined on the output of the encoder's latent stroke variable, z_s. The cross-entropy of the linear projection, $\hat{a}_i$, of the latent stroke variable $z_{s_{(i)}}$ and the corresponding stroke type s_i is minimized as

$$\hat{a}_i = W_{aux} z_{s_{(i)}} + b_{aux}, \tag{6.6}$$

$$\mathcal{L}_{aux}(s_i, \hat{a}_i) = -\sum_{j=1}^{C} s_i^{(j)} \log(\hat{a}_i^{(j)}). \tag{6.7}$$

Here, both objectives are described for a single stroke denoted by the subscript i, but in practice, the loss is the average of all strokes in a sequence. The total loss is the weighted sum of the two losses

$$\mathcal{L} = \gamma \mathcal{L}_{aux} + \mathcal{L}_{main}, \tag{6.8}$$

where γ is a hyperparameter, $\gamma = 0.3$ during experiments.

6.4 Experiments

6.4.1 Evaluation metrics

In badminton, more than one stroke is often a viable choice, which should be reflected in the evaluation metrics. The performance of the models is judged based on the accuracy (Acc) of their prediction, the top-2 accuracy Acc-2), and the top-3 accuracy (Acc-3). Additionally, the F1-score is included to inspect the model's ability to handle stroke-type imbalance in the sequences. The metrics were all introduced in subsection 4.3.4.

6.4.2 Baselines

To my knowledge, no other existing work uses stroke skeleton data to enhance future stroke prediction capabilities. Therefore, the model performance is compared to other sequence and action prediction baselines not explicitly designed for badminton. All model baselines consist of current state-of-the-art concepts for sequence prediction, and thus, while not intentionally designed for badminton stroke prediction, comparing to the baselines allows for a reasonable validation of the prediction capabilities of the RallyTemPose model. The following baselines are used for comparison:

- Seq2Seq [103]
- Transformer [109]
- Actionformer [79] + Transformer decoder

6.4.3 Implementation details

The dimension of embedded representation (d) per head is set to 16, the number of heads (h) in the MHSA is set to 4, and the forward expansion in the inner dimension of feed-forward layers is set to 4 following [51]. A rally's max sequence length (s) is set to 35, and T varies for different rallies. Similarly, the max temporal length of each stroke motion (T) is set to 30. Dropout and attention dropout are utilized in each MHSA block with a drop rate of 0.3. The models are trained with a batch size of 1 using AdamW with a learning rate set to 10^{-4}. Zero padding is performed for individual stroke motion sequences. Padding the rallies was also tested but did not improve performance.

Table 6.1: Accuracy (Acc), Top-2 Accuracy (Acc-2), and Top-3 Accuracy (Acc-3) of the models of this work and other baselines on the ShuttleSet and BadminDB datasets.

Split-M	ShuttleSet			BadminDB		
Model	Acc (%)	Acc-2 (%)	Acc-3 (%)	Acc (%)	Acc-2 (%)	Acc-3 (%)
Seq2Seq (LSTM)	47.9	72.4	83.5	57.3	82.3	86.0
Transformer	49.8	73.9	87.2	61.5	85.4	92.5
POT + Trans Dec	52.1	74.1	91.2	58.4	82.0	91.7
RallyTemPose	54.3	77.3	92.5	62.8	83.5	93.1

Table 6.2: Accuracy (Acc), Top-2 Accuracy (Acc-2), and F1-Macro (F1-M) are reported on the shuttleset + shuttleset22 rallies [115] with split-T data for the baseline models and RallyTemPose (RTP).

split-T	ShuttleSet			ShuttleSet22		
Model	Acc (%)	Acc-2 (%)	F1-Ma	Acc (%)	Acc-2 (%)	F1-Ma
Seq2Seq (LSTM)	48.4	73.0	39.7	49.0	76.3	40.4
Transformer	49.1	73.9	40.0	51.3	76.1	41.6
PoT + Trans Dec	52.9	78.5	43.9	56.3	81.0	47.1
TemPose-TF	51.6	72.7	42.0	54.7	73.3	46.4
RallyTemPose	**61.2**	**85.1**	**52.4**	**61.0**	**85.5**	**51.8**

6.4.4 Main Experiments

Split-M Forecasting: Experiments from Paper 2 compare forecasting models on the ShuttleSet and BadminDB datasets with Split-M input data. RallyTemPose outperforms the other baseline models in standard and top-3 accuracy. On the ShuttleSet dataset, it achieves an accuracy of 54.3%, a top-2 accuracy of 77.3%, and a top-3 accuracy of 92.5%, indicating its ability to rank the correct outcome within the top three predictions in over 90% of the cases. In the BadminDB dataset, the model achieves an accuracy of 62.8% and a top-3 accuracy of 93.1%. The BadminDB is much smaller than ShuttleSet, which resulted in the model often overfitting. As a result, the much simpler sequential models perform better on BadminDB comparatively. Here, RallyTempose shows only marginally better prediction accuracy than the baselines.

The results show the model's prediction prowess and reflect its ability to select the most logical outcomes. For a given situation, multiple stroke candidates can be per-

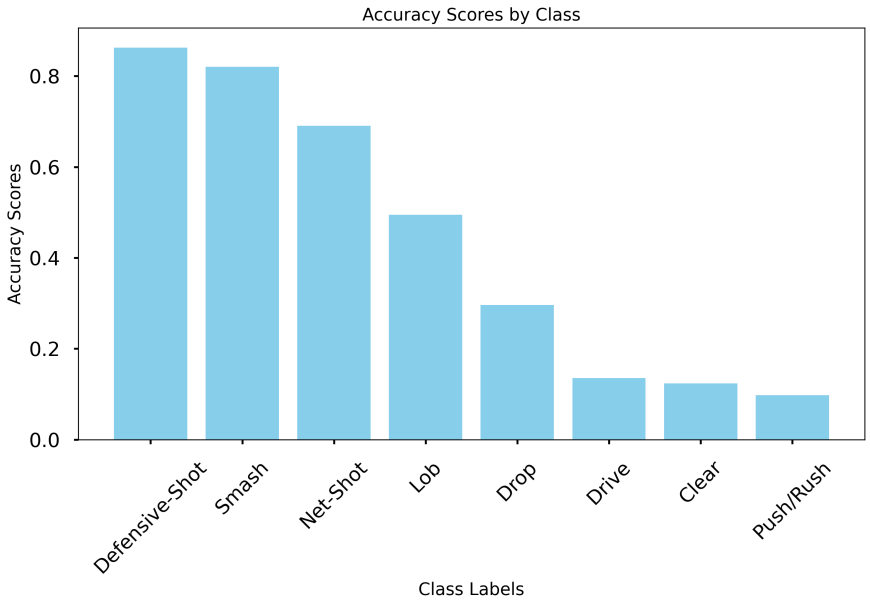


Figure 6.5: Comparisons of class accuracy for the different stroke types in the ShuttleSet dataset.

fectly viable simultaneously. The results in both accuracy metrics, especially in the top-3 accuracy, suggest that the RallyTemPose model's way of incorporating skeleton motion and player-specific information improves the prediction logic compared to the baselines in the context of badminton datasets.

Split-T Forecasting: Post-publication of Paper 2, experiments have been pursued for the Split-T input data, where the trajectory of the shuttlecock comprises the stroke. As previously stated, this is on the border between action anticipation and early action prediction (of the next stroke) since the preparation phase for the next stroke motion includes some of the preparation movement but none of the shuttle trajectory information.
From a practical perspective, it presents a more player-realistic situation for predicting the stroke choice of the opponent. The results are shown for the baseline models and RallyTemPose in Table 6.2 with the top results for each metric and dataset shown in bold. The experiments show that RallyTemPose (in the table RTP) outperforms all other baselines by a $> 3\%$ margin.
Another apparent observation is that the PoT + Transformer decoder also has a sig-

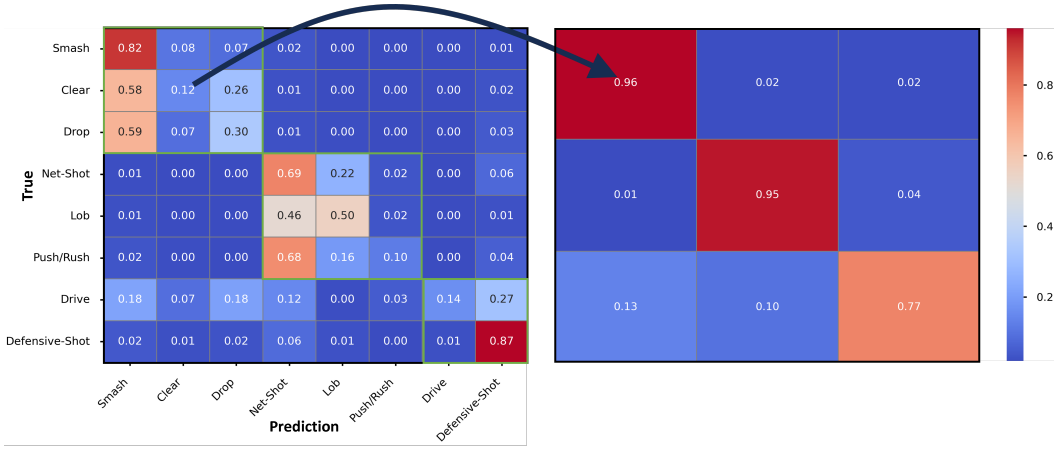


Figure 6.6: To the left is the confusion matrix for the shuttles data, and to the right is the confusion matrix grouped according to logical classes.

nificant performance improvement compared to the other baselines, which makes sense since the PoT + Dec skeleton-based transformer model also benefits from the added information of the preparation motion of the player performing the shot. Compared to the Split-M forecasting, the performance is improved by $5-7\%$, which seems reasonable given the extra information presented to the models. Including the shuttle trajectory $\mathbf{U}$ and Bone information $\mathbf{B}$ results in a minor performance increase across the metrics. However, including the shuttle position has a much smaller effect than anticipated. One possibility is that the current model architecture is not well-suited for handling the shuttle data, which is handled identically to the player court position in RallyTemPose.

Lastly, compared to the results from the cut-off experiments in chapter 5 Table 5.8 it might appear strange that TemPoseTF based only on the preparation phase skeleton motion (+ $\mathbf{U}$ and $\mathbf{G}$), i.e not information from the other strokes in the rally, can appear to perform better than RallyTemPose. However, that is actually not the case. The data is evaluated differently for the stroke recognition and forecasting experiments, as described in chapter 4. For example, the model will never encounter serves as the output target in forecasting experiments since the first stoke is always given as an observable input. To make a fair assessment of the two, the performance of the TemPoseTF is included in the comparison. This model sees no historical rally information, only prior stroke trajectory and current stroke preparation according to the

forecasting experiments procedure. Here, it can be observed that TemPoseTF performs slightly worse but is comparable to the Seq2Seq and Transformer baselines. Thus, having information about the rally does boost performance significantly.

Logical misclassifications: In Figure 6.5, the specific accuracy for all stroke types is plotted. Strokes like smash are accurately predicted, while strokes like clear and drives are only correctly predicted 12% and 14% of the time, respectively. However, by examination of the confusion matrix in Figure 6.6, most classifications can be attributed to logical reasoning, and all misclassifications belong to sensible groups (Netshots, push-rush, and lobs), (drives and defensive shots) and (smash, clears and drops). For example, a clear is predominantly hit from the backcourt on shuttle trajectories and racket swings similar to a smash and, to a lesser degree, a drop. This is consistent with the faulty prediction of clears being smashes or drops. Thus, the mispredictions follow the underlying logic of the game. Similarly, a drive can easily be confused with a defensive reaction shot.
The prediction model can still be improved further. One hypothesis is that a deeper strategic understanding of each situation can increase accuracy even more. However, the results indicate that the model, through a purely next-stroke action prediction, has developed a rudimentary understanding of the game of badminton.

6.5 Discussion

6.5.1 Ablation Study

The impact of the skeleton-based stroke condition on the prediction capability is examined through an ablation study. Three relative contribution are examined:

1. Skeleton data **S**
2. Ground position of the players **G**
3. The specific player embedding Id

Based on model input, the RallyTemPose encoder is altered, e.g., when the court position is not included, logically, the TCN block is left out of the encoder. Thus, the ablation study was performed for six different model variants. After removing specific model inputs and their corresponding model components, the respective prediction accuracies are shown in Table 6.3. The results show that the most critical factor is

Table 6.3: Ablation Study of RallyTemPose model. The Keypoint corresponds to skeleton data S, Ground to the court position G, and Player rep to the player IDs I. The experiment was performed on the Shuttleset data using motion-based splitting (split-M).

Keypoint	Ground	Player Rep	Accuracy (%)
			48.3
✓			49.2
		✓	46.9
	✓		51.6
	✓	✓	50.1
✓	✓		52.4
✓		✓	51.7
✓	✓	✓	54.3

the inclusion of the player ground position, as leaving out this data along with the TCN block leads to a 2.6% drop in performance. The encoder version made solely of a TCN block achieves a 51.6% accuracy. Thus, ground position is by itself the clearly most important input for the encoder module. This is consistent with the [116] research, where they used only the player and shuttle hitting position for predicting the next stroke type. The court position strongly dictates which strokes a player can viably hit and, therefore, is also consistent with basic understanding of the sport.
The skeleton keypoint data only slightly outperforms the decoder-only benchmark, which takes the sequences of embedded stroke labels as input. By itself, the skeleton data provides insufficient information to affect the forecasting performance positively. However, when supported by other information modalities, the skeleton data do lead to a positive performance improvement, achieving accuracies of 51.7% and 52.4% for player ID and court position, respectively.
Finally, the experiments indicate that the player-specific information does not significantly boost the prediction accuracy. Moreover, when player information is the only encoder input, the performance is even worse than that of the decoder-only baseline. Thus, player information does not bring significant value. However, as shown in the next section, learning player-specific representations allows for introspective player analysis that can be extrapolated from the model. Including the players' ground positions results in a significant performance boost. A potentially even greater performance increase could also be obtained by including precise 3D skeleton motion.

Table 6.4: Comparison of Text Embedding (TE) Methods, using either a learned embedding table for the stroke classes or a pre-trained bert for Text embedding of stroke descriptions.

Embedding Method	Acc (%)	Acc-2 (%)	F1-M (%)
Learned Embedding	58.5	83.7	49.6
BERT TE (Brief)	57.4	82.1	48.2
BERT TE (Detailed)	61.2	85.1	52.4

6.5.2 Text-embedding study

Table 6.4 shows the results of experimenting with different stroke sequence embedding methods. The performance for the top-1 and top-2 accuracy and the F1- macro scores are shown. The results show that the text embeddings of the detailed stroke descriptions lead to the best performance with about 1%. This gives a high probability that detailed-text embedding is the best-suited stroke embedding method for the forecasting model.

On the other hand, the brief descriptions actually yield a slightly worse performance than the learned stroke embeddings. This shows that the amount of detail in the descriptions actually has a significant impact on the performance. Doing more detailed experiments regarding the text descriptions in future experiments could be interesting, in two areas in particular. First, what information in the text is actually useful, and which details cause confusion? Second, can player characteristics be encoded in the stroke embeddings jointly with the stroke descriptions to provide more nuanced embeddings?

6.5.3 Match Analysis Prospects

The model's design allows for player comparison by analyzing the latent variables of the model, for both $\mathbf{z_s}$ – the encoder representation of a stroke – and $\mathbf{z_1}, \mathbf{z_2}$ the player representations for a stroke. Figure 6.7 and Figure 6.8 show t-SNE plots of the latent variables. t-SNE [12] is a dimensionality reduction method that reduces the dimensionality of the latent variables from $D_L \rightarrow 2$, allowing the representations to be visualized. In the visualization, $\mathbf{z_s}$ are colored based on the target stroke they represent, whereas $\mathbf{z_1}$ and $\mathbf{z_2}$ are colored according to the players they represent. Clear groupings are observed for the different $\mathbf{z_s}$ stroke variables and partial groupings of the player variables. This indicates that $\mathbf{z_s}$ and, to a lesser degree, $\mathbf{z_1}$ and $\mathbf{z_2}$ store

relevant information about strokes and playing styles, respectively. While the specific player embedding does not significantly improve the model's prediction accuracy, it allows for model intrinsic playstyle comparisons. Ensuring that the player information is stored properly in the $\mathbf{z_1}$ and $\mathbf{z_2}$ representations can surely be improved. One approach briefly explored was adding a regularizing loss-term based on predicting player IDs from $\mathbf{z_1}$ and $\mathbf{z_2}$. However, this yielded significantly worse prediction accuracy and has not been pursued further for the moment. Nevertheless, this direction will potentially be important for the development of a useful player comparison method.

Player Similarity Playstyle similarity of different players can be projected by looking at the cosine similarity of the player-specific latent variable for the other players in the dataset. The cosine similarity is calculated by random sampling of $N = 1000$, strokes, for each of the pair combinations of players and calculating the average cosine similarity between the latent player variables as

$$\text{Player Sim}_{i,j} = \sum_{n=1}^{N} \frac{\mathbf{z_i^n} \cdot \mathbf{z_j^n}}{\|\mathbf{z_i^n}\| \|\mathbf{z_j^n}\|}. \tag{6.9}$$

Table 6.5 shows the cosine similarity between the latent variables of players for five players. Observe that there is a notable difference in similarity between the players. On average, the male (first three players) and female (last two players) have a lower similarity, whereas the same gender similarity scores are higher. However, the player similarity score is also quite low between the three males. This is quite meaningful since Male 3, known for a unique, endurance-based, hard-to-read playstyle, Male 2, with a very fast-paced style, and Male 1, with a physical and powerful playstyle, are very different player types and the similarity score seems to reflect just that. Future work could include categorizing distinct playstyles and attempt to interpret them as defensive, offensive, power, placement, etc.

Playstyle analysis In Figure 6.9, a bar plot of the average accuracy for each player in the ShuttleSet dataset is shown. There is a notable gap of more than 20% average accuracy between the players for which strokes are predicted the best compared to the player predicted the worst. The prediction accuracy of specific players could potentially be used to indicate how well players can mask their strokes. However, that

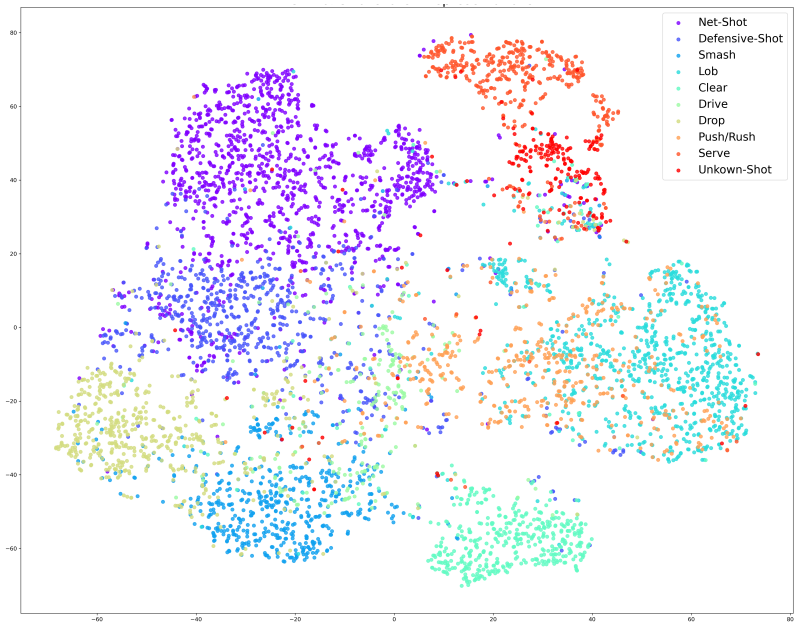


Figure 6.7: t-SNE plot over the latent stroke z_s representation, colored according to the observed stroke types.

approach would have to assume that the model can flawlessly predict straightforward unmasked strokes, which is not yet guaranteed. Still, through continuous improvement of the model, this could be a helpful asset for player analysis.

6.5.4 Future prospects

One aspect that could prove valuable for analysts doing match preparation would be the potential to generate realistic rallies for different match situations (game states) since that would allow in-depth sampling of the model's top suggestions for best con-

Table 6.5: Cosine similarity between latent player variables of different classes. (M: male, F: female)

Player sim	M1	M2	M3	F1	F2
M 1	0.61				
M 2	0.43	0.58			
M 3	0.37	0.41	0.67		
F 1	0.21	0.19	0.31	0.71	
F 2	0.23	0.51	0.49	0.57	0.65

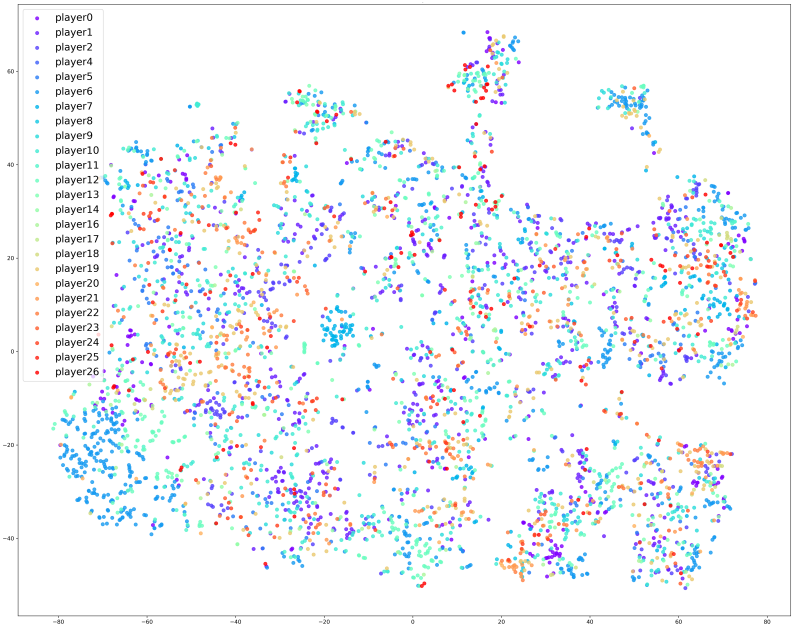


Figure 6.8: t-SNE plot over the latent player representation (z_1 and z_2), colored according to the target player Id. Note the lack of distinct groupings of the player variables, which could be explained by the difference/similarity in how players perform certain strokes.

tinuation (choice of stroke) in the given conditions. The forecasting architecture is the primary component in such a pipeline. While not yet implemented, a possible complete rally forecasting architecture would include the following 3 extensions to the current architecture:

1. The Stroke prediction decoder is tasked with predicting the duration of the next stroke in the rally, similarly to [75], where the stroke type prediction would be used to predict the stroke duration.

2. An additional Skeleton-motion decoder module would be added, fx based on [25] This would be based on the Stroke and player embedding + the stroke prediction/label. The skeleton motion would be predicted using an Inverse RallyTemPose encoder module with the duration dictated by the output of the Stroke prediction decoder.

3. The complete architecture would be trained in two phases. Firstly, skeleton-prediction decoders and the RallyTemPose decoders are trained separately on

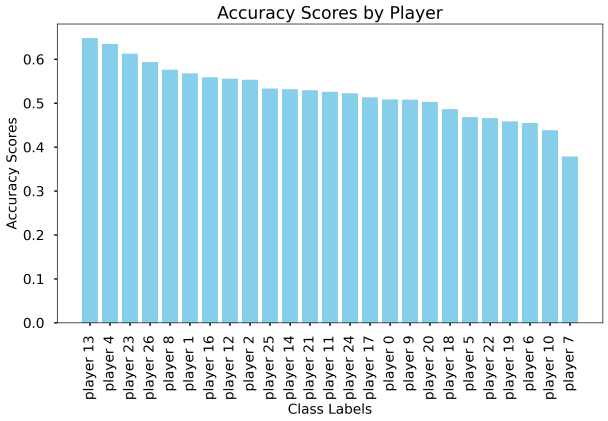


Figure 6.9: The average accuracy of next stroke predictions for all the players in the dataset.

their task. Secondly, joint training of the two model components, which can be trained separately in pertaining steps and then jointly.

Not only would a ”complete” rally generation pipeline be a valuable tool for analysts and coaches. A more coarse rally generation architecture could also significantly improve stroke recognition models. As discussed in section 5.5, even slightly inaccurate sequences could be used to pretrain recognition models before fine-tuning on curated real stroke sequences.
Another prospect involves enhancing the model's capabilities by incorporating additional variables, such as match outcomes (win/loss) to facilitate more sophisticated tactical analysis. Regularizing stroke-embedding and player-embedding loss terms might also improve the stability of the model training, while enhancing performance on the evaluation metrics.

6.6 Chapter Conclusion

This research introduced the RallyTemPose model designed specifically for stroke prediction in badminton, utilizing an encoder-decoder architecture. The model integrates skeleton data and player-specific information using a spatiotemporal transformer encoder.
Experiments conducted on two different real-world badminton datasets show an increase in performance for this approach compared to other forecasting baselines. Fur-

thermore, the extracted latent representations show potential use for player analysis and match preparation.

Chapter 7

3D Reconstruction of Shuttlecock Trajectories

In the previous chapters, badminton-specific downstream tasks were attempted using extracted features with the player skeleton, and shuttle position coordinates as input. However, all features (except player court position) were expressed in image coordinates. This final chapter looks into estimating real-world 3D coordinates of the shuttle trajectory for in-the-wild scenarios such as broadcasted badminton matches. While the skeleton modality was the primary modality focused on throughout the thesis, the previous chapters show that the shuttlecock provides valuable information to distinguish between and predict badminton strokes. Additionally, accurate estimation of the world space (i.e., real-world 3D) coordinates allows for direct extraction of specific player performance metrics, such as shuttle placement, shuttle speed of shots, hitting positions, shot height, flight time, etc. This information is essential to profiling players and helpful when broadcasting matches.

The central theme of the chapter is addressing the issue of the limited availability of 3D ground truth data. Publicly available data is either limited in its variety or the sheer number of training samples, which proves insufficient for teaching robust 2D to 3D lifting models that generalize to in-wild videos. Badminton videos present additional challenges because detailed, precise movements make up the stroke motion and shot trajectories. Badminton shots can reach velocities of up to $420 km/h$, and the trajectories can reach a height beyond 10 meters, often moving out of the camera frame. High-quality 3D data does exist, but it is mostly proprietary and owned

by specialized companies such as HawkEye, stemming from their multi-view camera recordings of badminton matches.
In the absence of real available 3D ground truth material, an alternative approach is necessary. This chapter will explore the potential of creating viable synthetic 3D data from 2D image projection combined with physical 3D shuttle flight modeling.
The chapter is divided into two sections.

1. The pin-hole camera model is introduced, which allows projecting 3D world coordinates to an image plane specified by the given camera parameters.

2. Estimating or "lifting" shuttle trajectories from 2D image coordinates to real-world 3D coordinates. The chapter extends on ideas from the Paper 3.

The key distinction between strokes and shots was defined at the beginning of the thesis in subsection 1.4.2. A stroke is attached to the motion of the player, which is referred to as a split-M motion sequence. Correspondingly, the trajectory of the shuttle produced by the stroke motion is defined as a shot. The motion (shuttle and skeleton) during the shot's trajectory is called a split-T motion sequence. Since this chapter looks deeper into the shuttle trajectory and reconstruction of the 3D trajectory, the data sequences will be referred to as shots.
The premise for the 3D shuttlecock reconstruction is centered around the physical modeling of the shuttle trajectories. The trajectories can be viewed as an initial value problem (IVP), with the equations of motion constituting the differential equations that describe the shuttle flight and the initial position and velocity providing the initial conditions of the system. This way the trajectory of different shots can be simulated numerically just by choosing realistic starting values. By sampling camera positions from known broadcasted matches, it is possible to create (2D,3D) pairs with no practical upper bound on the number of samples that can be simulated.
It is then demonstrated that the lifting model trained on the synthetic trajectories generalize well to the real-world broadcasting matches, using a combination of unique evaluation metrics to support the 2D projection error.
The shuttlecock 3D reconstruction section is an extension of Paper 3, a paper published in collaboration with two master's students based on their thesis work. The paper focuses on the potential of physical modeling to generate synthetic training data

for ball trajectory estimation in tennis [1]. Since the publication, the premise has been repurposed for badminton, with an entirely new code-base and improvements and extensions of the original Paper 3 suggested.
The contribution of the chapter and extensions to Paper 3 consists of the following:

1. Redefinition and code implementation of the synthetic reconstruction method for badminton.

2. Increased attention to detail for simulation of synthetic badminton strokes to allow for better generalization to real data.

3. Increasing robustness of the model by sampling camera positions from the distributions of camera positions of the broadcasted matches in [114]. All camera parameters were calibrated manually.

4. Testing more complex network architectures for the synthetic training pipeline, Transformer, TCN, LSTM, etc.

5. A unique cross-view consistency metric assesses the robustness of the predictions from several different angles.

6. Shot classification using shuttle trajectories, strictly comparing 2D vs 3D data.

Transitioning between real-world 3D coordinates (world space) and camera coordinates is important in training lifting models, especially in pseudo-self-supervised approaches with no ground-truth 3D data.

7.1 Camera models

Recall, the brief introduction to homogeneous coordinates section 4.1. A property of homogeneous coordinates is that all non-zero scalar multiples, i.e., $s\mathbf{P_h}$ (and $s \neq 0$), represent the same point. This is the defining property of perspectivity [50], which results in the points along a ray in the pinhole camera all projecting to the same image point.

[1] Credit goes to the students for writing most of the code, and the paper is based mainly on their thesis. As their supervisor, the contribution of the undersigned suggested the main idea behind the paper (and thesis) and played an important role in shaping the paper's development

7.1.1 Pinhole Camera Model

The pinhole camera model can project a 3D scene point in an arbitrary world frame system $\mathbf{P_w} = (X_w, Y_w, Z_w, 1)^T$ into a 2D image plane point $\mathbf{p} = (u, v, 1)^T$ through a perspective transformation. Both $\mathbf{P_w}$ and $\mathbf{p}$ are in homogeneous coordinates, making them 4D and 3D vectors, respectively. For simplicity, the 'homogeneous' part is often dropped and referred to as vectors.
The pinhole camera model's transformation [2] can be represented by the following matrix equation:

$$s\mathbf{p} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}]\,\mathbf{P_w} \tag{7.1}$$

where $\mathbf{P_w}$ is the 3D[3] point in the world frame, $\mathbf{p}$ is the 2D point in the image plane, $\mathbf{K}$ is the camera intrinsic matrix, $[\mathbf{R}|\mathbf{t}]$ is the joint rotation-translation matrix, where $\mathbf{R}$ and $\mathbf{t}$ are the rotation matrix and translation vector, and s is an arbitrary scale factor. Per Equation 7.1, the mapping from world to image coordinates is composed of two steps. First, the 3D point $\mathbf{P_w} = (X_w, Y_w, Z_w, 1)^T$ is mapped from the world frame to the camera frame $\mathbf{P_c} = (X_c, Y_w, Z_w, 1)^T$[4]. Second, the 3D position in the camera basis is projected onto the 3D homogeneous image coordinates using the intrinsic camera matrix.
The change of basis from world frame to camera frame coordinates is a product of 2 different matrix operations. The changing of the basis from the world frame to the camera frame can be represented by the following linear mapping/matrix operation:

$$\mathbf{P_c} = \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{P_w} \tag{7.2}$$

where the r_{ij} and t_i are the rotation parameters. Both the rotation and translation

[2] Assuming distortion-free projection
[3] 4D vector since it is represented in homogeneous coordinates.
[4] I.e. a change of basis from world frame to camera frame.

matrix have 3 degrees of freedom (DoF)[5]. Subsequently, the projective operation

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = [\mathbf{I}|\mathbf{0}]\mathbf{P_c} \tag{7.3}$$

extracts the 3D camera coordinates from the 4D homogeneous vector P_c. Thus, the extrinsic part of the camera projection can be governed by the joint rotation-translation matrix:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = [\mathbf{I}|\mathbf{0}] \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = [\mathbf{R}|\mathbf{t}]\mathbf{P_w} \tag{7.4}$$

The 3D point in camera coordinates are projected onto the 2D image plane using the intrinsic parameters. Intrinsic parameters describe the internal characteristics of the camera. They include the focal lengths f_x and f_y in the x and y directions, and the principal point coordinates (c_x, c_y). The focal length dictates the scale of the image the camera sees along the x, y axes in pixel units. The optical center is the point in the image coordinates corresponding to the projection of the camera center in the image plane. The parameters make up the intrinsic matrix $\mathbf{K}$

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{7.5}$$

[5]Even though the rotation matrix has 9 parameters, only 3 are independent. These 3 DoF correspond to the rotational orientation of the camera, i.e., rotation around the X-axis, Y-axis, and Z-axis.

from which the projection to image coordinates can be expressed as:

$$\mathbf{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} \tag{7.6}$$

$$= \begin{pmatrix} f_x X_c + c_x Z_c \\ f_y Y_c + c_y Z_c \\ Z_c \end{pmatrix} = Z_c \begin{pmatrix} \frac{f_x X_c + c_x Z_c}{Z_c} \\ \frac{f_y Y_c + c_y Z_c}{Z_c} \\ 1 \end{pmatrix} = Z_s \begin{pmatrix} f_x \frac{X_c}{Z_c} + c_x \\ f_y \frac{Y_c}{Z_c} + c_y \\ 1 \end{pmatrix} \tag{7.7}$$

Because the product of K and the 3D camera coordinates can be expressed as a 3D homogeneous vector (in the image coordinates), the scale ambiguity in the pinhole camera model emerges. The point's true scale (or depth) along the viewing ray is lost due to the fundamental property of perspective transformations.
By combining both steps, the transformation from a 3D world point to a 2D image point represented by the Equation 7.1:

$$s\mathbf{p} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{P_w} \tag{7.8}$$

$$Z_c \begin{pmatrix} f_x \frac{X_c}{Z_c} + c_x \\ f_y \frac{Y_c}{Z_c} + c_y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{7.9}$$

The projective transformation of the 3D points has 10 DoF, which means that to calibrate a camera to perform the projection, 5 corresponding $(\mathbf{p}, \mathbf{P_w})$ sets of points are needed. In practice, the image produced by the camera will be subject to some degree of distortion. The equations for the two most common distortion types, radial and tangential, are provided. Including the distortion parameters, at least 8 corresponding points are needed to estimate the camera parameters. Accounting for distortion is done with a few additional equations, but will not be elaborated further in this thesis.

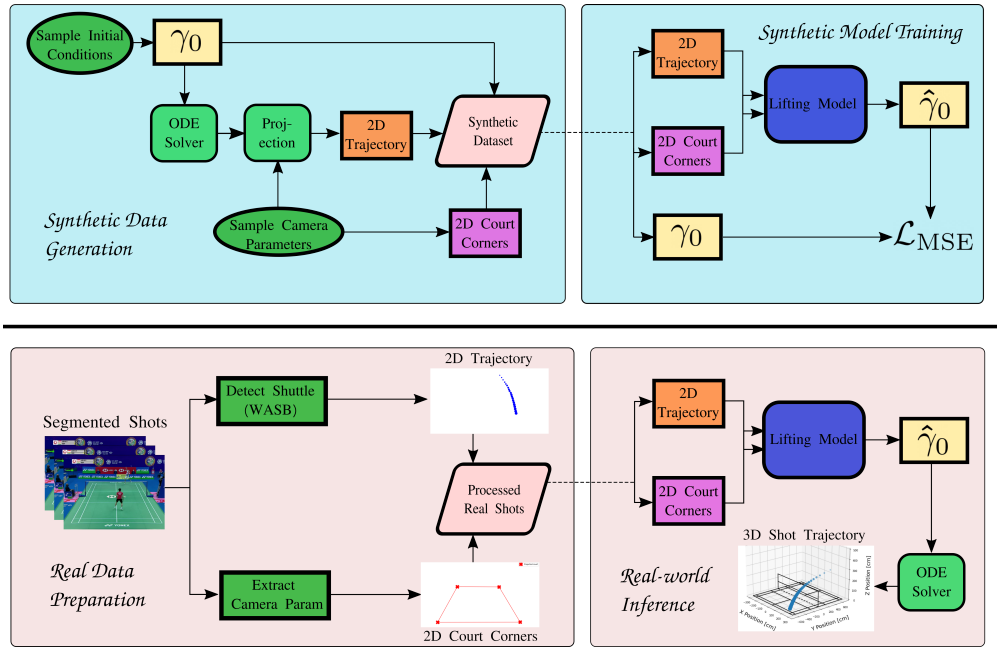


Figure 7.1: Complete End-to-End Pipeline. *For training*, synthetic data is generated by sampling initial conditions (γ_0). This data creates 3D and image trajectories along with sampled camera parameters. The model predicts initial conditions based on image trajectories and court corners. The lifting model is iteratively trained using Mean Squared Error (MSE) between true γ_0 and predicted $\hat{\gamma}_0$ as the loss function. *For inference*, features are extracted from segmented video data of competitive matches in shuttleset22. Individual shots are then used alongside court corners as input to the model. The Lifting model (TrajTrans) predicts a set of initial conditions, enabling the estimation of 3D shot trajectories.

7.2 3D shuttle-trajectory estimation with Physics-based modeling

In high-level racket sports, accurate 3D information on the ball/shuttle holds great value, assisting umpires in calling the game and gathering player and shot information for analytical purposes. Currently, the most precise commercially available 3D reconstruction technology is the Hawk-Eye system[6], which uses multiple cameras and triangulation to capture the ball's 3D position with an error margin of less than 3.6 millimeters. However, such a system is expensive and accordingly only available for the highest level of play. Most athletes will lack access to that quality of data. Enabling the tracking of the shuttle's 3D position using a single, everyday camera de-

[6] https://www.hawkeyeinnovations.com

vice, such as a mobile phone, would allow the average players to access game statistics previously reserved for professionals.

Estimating 3D ball positions from monocular video has been attempted in sports like volleyball [17], basketball [15], and badminton [72], but none has explored the synthetic training approach that was presented in Paper 3, which aims to enable efficient estimation of 3D trajectories from monocular video in racket sports. Emphasis in this chapter is for obvious reasons on Badminton.

Direct 2D-to-3D lifting of ball coordinates presents a significant challenge since $(2D, 3D)$ — pairs of ball coordinates are generally not publicly available. Instead, segmented individual shots can be modeled by kinematic motion equations including air resistance. The physical laws of motion set up differential equations for the trajectories, after which a numerical ordinary differential equation (ODE) solver is used to estimate the 3D positions of the shuttle iteratively for the full trajectory. The 3D coordinates can be mapped to 2D image coordinates using the camera parameters generating synthetic "pseudo" ground truth input-output pairs. Usually, a training loop involving iterative time integration and subsequent 3D-2D camera projection is too unstable to converge. Instead, a transformer architecture that predicts the **initial conditions** ($\boldsymbol{\gamma_0}$), given the image (2D) trajectories as input, provides a significantly smoother training routine.

Before training the model, the input/output pairs are obtained by selective sampling of initial conditions and subsequent trajectory simulation using a 4th-order runge-kutta time integration method. Then, only keeping trajectories, passing the net, and landing within the court allows for the acquisition of 3D shot trajectories. Finally, the image (input) coordinates are retrieved by projection to the image plane using sampled camera positions from the distribution of known camera settings (extrinsic and intrinsic parameters) from shuttleset22 matches.

A model could also be trained on the synthetic (2D,3D) pairs. However, compressing input image trajectories to initial condition predictions allows for better generalization of real-world trajectories.

The work of this thesis does not dive into segmentation/localization of stroke/shots from full badminton matches. Instead, the starting point of the thesis is an already segmented shot/stroke. Student projects, etc have looked into the segmentation/localization task. For example, a section of Paper 3 looks into hit segmentation to predict the duration of a shot (ball trajectory) where the start and end of each shot were identified

by detecting hits and bounces (in tennis). The *SynthNet* pipeline can be seen in Figure 7.1. This chapter focuses on badminton, which can be generalized to any racket sport with minor additional effort granted .

7.2.1 Related work for 3D trajectory estimation

Prior works in the field of computer vision in sports have focused on extracting individual components from video, such as court detection [41, 119, 111], ball/projectile tracking [64, 107, 102] or human pose estimation [11, 59] where the extracted features has mainly been utilized to perform downstream tasks, e.g., action recognition and forecasting, which have already been demonstrated in the earlier chapters of the thesis.

When doing 3D tracking of sports balls, the most used and most reliable way is to use a multi-camera setup [92, 122, 125, 42], where the scene is captured from multiple angles and triangulation is used to estimate the balls 3D location. While this can give accurate results, it is less cost-efficient than using a single camera as one needs access to multiple cameras and the possibility and permission to set them up.

Research in 3D ball tracking from monocular video has been done in several sports such as volleyball [17], table tennis [96], basketball [15] and badminton [72], all using a similar approach. Like in the case of this thesis, they view the 3D estimation task as an Initial Value Problem (IVP), where the projectile kinematic equations govern the motion of the balls. With accurate initial conditions the motions can be determined through numerical integration. Thus, they extract video features and use optimization techniques to find the best initial conditions for a reconstructed 3D trajectory. This method has evident flaws. First, using optimization techniques means optimizing each shot one at a time, resulting in long inference times. Secondly, the reprojection error between the reconstructed 3D trajectory and the image trajectory is used as part of the loss function, which is undesirable as this can lead to unrealistic trajectories. Instead, this thesis proposes using a neural network to predict the initial condition, train this network on synthetic data, and use the ground truth initial condition for loss during training.

7.2.2 3D Reconstruction Task Definition

The objective is to develop a model capable of reconstructing 3D shuttlecock shot trajectories from monocular video. Given segmented shot video segments, the process involves extracting the shuttle and court image coordinates, which a lifting model uses to predict the 3D initial conditions. The 3D trajectory can then be estimated through numerical integration with an ordinary differential equation (ODE) solver. The lifting model is trained exclusively on synthetic data to estimate initial conditions γ_0 for ballistic 3D trajectories using projected 2D image trajectories. Thus, the actual training dataset consists of image trajectories and γ_0 input-output pairs. The synthetic data generation process will be explained in detail in the following section. The model is evaluated on synthetic and real data using reprojection error metrics and a consistency loss, which tests whether perturbations to the camera position affect the predicted 3D trajectories. To reach a robust lifting model capable of generalizing to real *flawed* image input, much emphasis is put on the simulation and augmentation of the synthetic 3D trajectories, such that a diverse dataset reflecting accurate match shots is procured.

7.2.3 Synthetic Learning Procedure

Dividing each match (video) into a sequence of 3D shot trajectories (split-T) means each shot can be modeled as a projectile under drag:

$$\frac{d^2\mathbf{x}(t)}{dt^2} = \mathbf{g} - \frac{D}{m}|\mathbf{v}(t)|\mathbf{v}(t), \tag{7.10}$$

where m is the mass of the shuttlecock, $\mathbf{x}$ and $\mathbf{v}$ are the 3D position and velocity vectors, g is the gravitational constant, and D is the drag constant, which is a multiplication of the following physical constants:

$$D = \frac{1}{2}\rho A C_D = \frac{1}{2}\rho\left(\frac{R^2\pi}{4}\right)C_D, \tag{7.11}$$

with ρ being the air density at 20°, A is the cross-sectional area, R is the cross-sectional radius of the shuttlecock end, and C_D is the drag coefficient. The actual values of the constants used for simulating the trajectories can be found in Table 7.1, which uses the values from an empirical study [87], where the C_D estimate in particular is based

Table 7.1: Physcial constants for modeling shot trajectories, i.e., constructing proper equations of motion. The value of the constants are taken from [87].

Constants	g (cm/s^2)	m (kg)	A (cm^2)	R(cm)	ρ (kg/cm^3)	C_D
	$9.82 \cdot 10^2$	$5.85 \cdot 10^{-3}$	33.39	6.53	$1.2 \cdot 10^{-6}$	0.65

on free-fall testing of the shuttlecock.

The 3D equations of motion in Equations 7.10 constitute three coupled partial differential equations, which have no analytical solution but can be solved numerically by ODE solvers to retrieve a discretized version of the shuttle position with N time steps. Forward Euler's time integration provides a simple solution method. At each step, the acceleration is assumed constant for a sufficiently small time step $\Delta t = t_{n+1} - t_n$, where $n \in N$ is the current step. Thus, for each small Δt, we calculate acceleration $\mathbf{a_{n+1}}$, velocity $\mathbf{v_{n+1}}$, and position $\mathbf{x_{n+1}}$, based on the acceleration, velocity, and position of the current time n step as follows:

$$\mathbf{a}_{n+1} = \mathbf{g} - \frac{D}{m}|\mathbf{v}_n|\mathbf{v}_n \tag{7.12}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n \Delta t \tag{7.13}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n \Delta t. \tag{7.14}$$

To create a 3D trajectory initial conditions $\mathbf{v}_0 = (\mathbf{v}_{x0}, \mathbf{v}_{y0}, \mathbf{v}_{z0})^T$ and $\mathbf{x}_0 = (x_0, y_0, z_0)^T$ are required. A visualization of this process can be seen in Figure 7.2. Origin of the world space is placed in the middle of the badminton court. The x-axis is oriented parallel to the net (width) of the court, with the positive direction being from left to right. The y-axis direction is oriented perpendicular to the net, with positive pointing away from the camera. The z-axis is oriented vertically, perpendicular to the ground, pointing upwards.

Above, Eurler's method was used to demonstrate how differential equations can be solved numerically. The actual trajectory simulation procedure uses the 4th order Dormand-Prince pair Runge-Kutta formula [33], which can model the 3D trajectories more efficiently as a smaller number of timesteps are required to obtain accurate

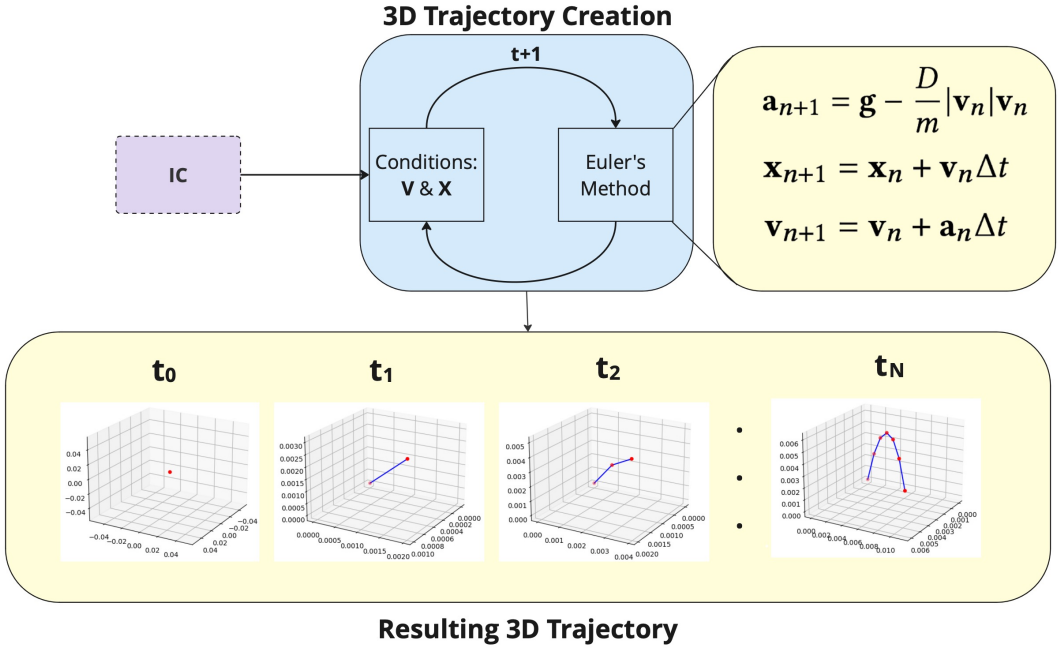


Figure 7.2: Visual presentation of creating 3D trajectory given a set of initial conditions (IC). The loop runs in N iterations, creating a position and velocity with each time step. Figure taken from Paper 3 [38]

3D solutions to the trajectories.

Sampling procedure The practical generation of synthetic dataset trajectories involves two overall generation steps:

1. Sampling γ_0 that, through simulation, leads to realistic and diverse shot trajectories.

2. Sampling of the camera parameters for projecting 3D trajectories to 2D "image" coordinates that can be used for input.

The simulated trajectories should span a broad range of trajectories, mimicking most strokes performed during a match. Therefore, the initial conditions + flight duration are defined for the different shot types: {Smash, Clear, Block, Lob, Drive, Net-Kill, Drop }. The different shot types are shown in Figure 7.3. The start conditions for simulating different shot types can be found in Table 7.2. They encompass the shuttlecock's starting position, speed, and spherical launch angles. The initial velocity is

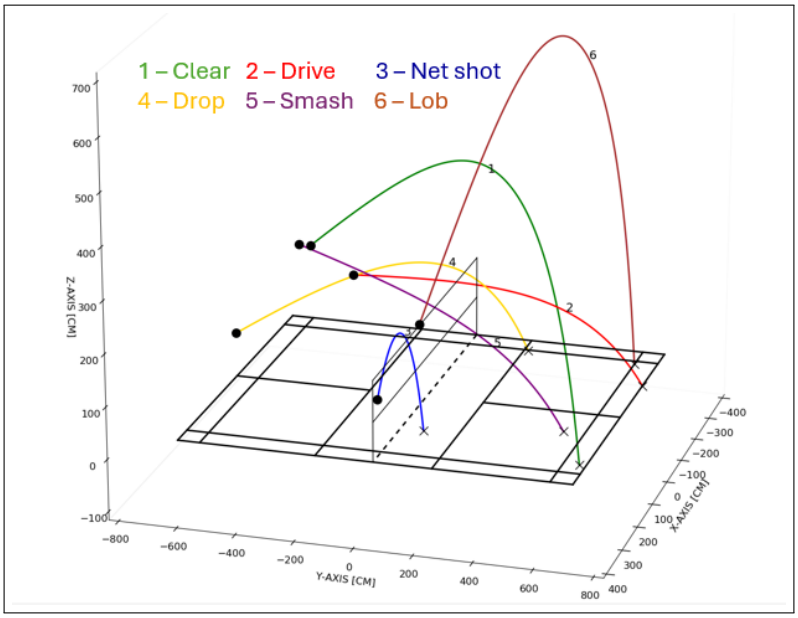


Figure 7.3: 3D Trajectories of the shuttle produced by Initial Conditions for different stroke types.

Table 7.2: Simulation Parameters for Badminton Strokes. v_0: Initial speed [cm/s], θ: Elevation angle [degrees], ϕ: Azimuth angle [degrees], x_0: Initial x-axis court position [cm], y_0: Initial y-axis court position [cm], z_0: Initial height [cm], t: Time [s].

Stroke	v_0	θ	ϕ	x_0	y_0	z_0	t
Clear	$[3500; 8000]$	$[30; 55]$	$[70; 110]$	$[-305; 305]$	$[450; 670]$	$[150; 275]$	$[0.82; 1.0]$
Drop	$[2000; 4500]$	$[5; 15]$	$[40; 140]$	$[-305; 305]$	$[300; 650]$	$[100; 225]$	$[0.80; 1.0]$
Smash	$[6000; 10000]$	$[-50; 0]$	$[60; 120]$	$[-305; 305]$	$[500; 600]$	$[200; 300]$	$[0.5; 1.0]$
Drive	$[3000; 4500]$	$[-5; 8]$	$[60; 120]$	$[-305; 305]$	$[150; 450]$	$[100; 180]$	$[0.4; 1.0]$
Net Shot	$[500; 2000]$	$[30; 80]$	$[30; 150]$	$[-305; 305]$	$[10; 180]$	$[30; 145]$	$[0.5; 1.0]$
Net Kill	$[2500; 5500]$	$[-80; -15]$	$[45; 135]$	$[-305; 305]$	$[20; 100]$	$[160; 220]$	$[0.90; 1.0]$
Lift	$[4000; 5000]$	$[40; 75]$	$[70; 110]$	$[-305; 305]$	$[30; 400]$	$[20; 60]$	$[0.80; 1.0]$
Block	$[2500; 4500]$	$[10; 50]$	$[45; 135]$	$[-305; 305]$	$[150; 450]$	$[10; 150]$	$[0.70; 1.0]$

sampled in spherical coordinates since it allows for a more homogenous sampling of shots with a specific speed and launch direction compared to cartesian coordinates. The sampled initial conditions are then input into a numerical ODE solver, in this case, based on the Runge-Kutta method, to compute the shuttlecock's trajectory. The resulting trajectory is evaluated to ensure it passes over the net and lands within the court boundaries, aligning with realistic play scenarios. If the trajectory does not meet these criteria, the initial conditions are resampled, and the simulation is rerun until a valid trajectory is achieved. Thus, the trajectories are generated using a Monte Carlo accept-reject methodology. The iterative process ensures that the simulated trajectories closely mimic real-world shuttlecock behavior. The generation procedure is shown in algorithm 2.
The camera setting parameters of the synthetic are sampled from the position distribution of the 47 different camera positions of the shuttleset22 matches. As seen in Figure 7.4a, broadcasting cameras are permanently placed directly behind the court for broadcasting matches. This is very natural since it produces the best viewing experience. However, this does provide a narrow diversity in viewing perspectives of projected 2D trajectories. This lack of variety in 2D input could inhibit learning for the lifting model and limit generalization to real-world inference. This is addressed by:

1. converting a sampled position from cartesian to spherical coordinates.
2. Rotating the position along the azimuthal and polar angle while keeping the same radius.
3. Orienting the camera direction toward the middle of the net and converting back to cartesian coordinates.

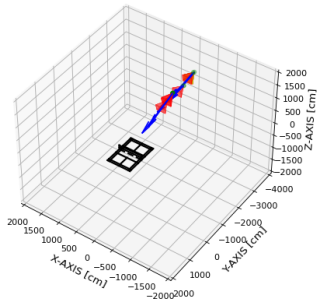
This produces much more diverse 2D input trajectories from which the model can learn. The effect is shown in Figure 7.4, where six sampled camera settings are shown along with the original (yellow) camera placement from shuttleset22.

Model

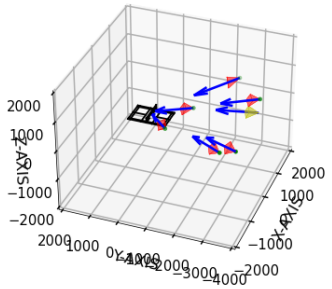
The prediction capabilities of several time-series models (TCN, GRU, LSTM) along with the FNN (MLP) baseline from the synthnet paper (Paper 3) are tested. However, the primary model proposed is a trajectory transformer (TrajTrans), taking

Algorithm 2: Shuttlecock Trajectory Simulation

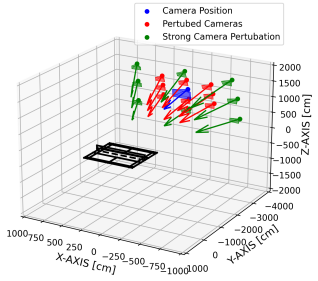
Data: Shot parameters for each type and side
Result: Valid trajectories for each shot type and side
Function `ProjectileMotion3D(`t, **y**`):`
    Extract position (x, y, z) and velocity (v_x, v_y, v_z) from **y**;
    Compute speed $v = \sqrt{v_x^2 + v_y^2 + v_z^2}$;
    Compute drag forces $F_{\text{drag},i} = -\frac{1}{2} C_d \rho A v v_i$ for $i \in \{x, y, z\}$;
    Compute accelerations $a_i = \frac{F_{\text{drag},i}}{m}$ for $i \in \{x, y\}$;
    Compute acceleration $a_z = -g + \frac{F_{\text{drag},z}}{m}$;
    return $[v_x, v_y, v_z, a_x, a_y, a_z]$;
foreach *shot type* **do**
    foreach *court side* **do**
        repeat
            Sample initial conditions $(x_0, y_0, z_0, v_0, \theta, \phi)$;
            Convert θ and ϕ to radians;
            Compute initial velocities $v_{x0} = v_0 \cos\theta \cos\phi$, $v_{y0} = v_0 \cos\theta \sin\phi$, $v_{z0} = v_0 \sin\theta$;
            Initialize ODE solver with state $[x_0, y_0, z_0, v_{x0}, v_{y0}, v_{z0}]$ and `ProjectileMotion3D()`;
            while *solver successful and* $z \geq 0$ **do**
                Integrate to next time step;
                Append (x, y, z) to trajectory;
            Verify trajectory passes over net and lands within court;
        until *valid trajectory obtained*;
        Store shot data: type, side, initial conditions, trajectory;



(a) Fixed camera settings from shuttleset22



(b) Camera setting sampled from shuttleset22 distribution with rotated augmentations in spherical coordinates.



(c) Positional perturbations + orientation on original match camera placement.

Figure 7.4: Different camera view "sampling methods"

the image trajectories and court corners as input to predict initial conditions. Transformers scale well with a large amount of training data. They are, therefore, a logical model choice given the *virtually unlimited* amount of synthetic data that can be generated for training. The composition of a standard transformer layer can be seen in Figure 2.4. The specific TrajTrans architecture is depicted in Figure 7.6.

Multi-view Supervised Learning with Consistency Regularization

The training objective of the model is to predict a set of the six initial conditions $\hat{\gamma}_0$, compared to the ground truth $\gamma_0 = (x_0, y_0, z_0, v_{x0}, v_{y0}, v_{z0})$. Since γ_0 are used to simulate the 3D trajectories with the numerical ODE, the objective is implicitly to predict

the best-fitting 3D trajectory. Since the synthetic data provides ground-truth initial conditions (a target vector $\gamma_0 \in \mathbb{R}^m$) viewed from different camera projections, the variety in 2D viewpoints can be used to improve robustness and consistency for the model. This is realized through implementing a regulatory cross-view consistency loss and the mean-squared error (MSE) loss between prediction $\hat{\gamma_0}$ and ground truth target for γ_0 for each view.
Concretely, MSE is computed with respect to the ground truth γ_0 and the model prediction for each view v_i, the model prediction $\hat{\gamma_0}^{(v_i)} = f(x^{(v_i)})$.

$$\mathcal{L}_{\text{sup}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{V}\sum_{j=1}^{V}\|\hat{\gamma_0}_i^{(v_j)} - \gamma_0\|_2^2, \tag{7.15}$$

where γ_0 is the ground truth initial conditions and $\hat{\gamma_0}$ the predicted initial conditions. Implicitly, averaging over all views ensures the model learns to produce consistent and accurate predictions regardless of which view it sees.
Additionally, the model can be further encouraged for predictions across different views of the same shot to be similar by imposing a consistency constraint:

$$\mathcal{L}_{\text{consistency}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{V_{j\neq l}}\sum_{j\neq l}\|\hat{\gamma_0}_i^{(v_j)} - \hat{\gamma_0}_i^{(v_l)}\|_2^2. \tag{7.16}$$

Thus, the overall loss becomes:

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda\mathcal{L}_{\text{consistency}}, \tag{7.17}$$

where λ is a hyperparameter controlling the strength of the consistency regularization. For experiments performed in this chapter $\lambda = 0.5$.
This term ensures that the model tries to align predictions across views, even if one view is noisy or incomplete. Combined with the supervised loss, this encourages the model to produce stable, view-invariant predictions that match the target. 3 different camera view input – and predictions – are shown with the simulated 3D trajectory in Figure 7.5.

Synthetic Data Augmentation The lifting model is trained purely on simulated 3D shot trajectories and has not seen any real trajectories during training. As such,

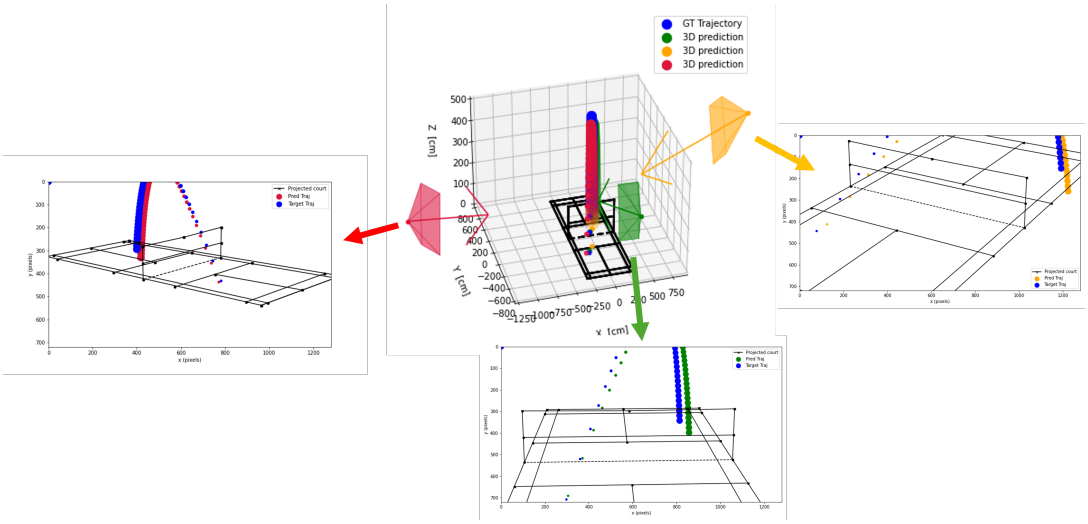


Figure 7.5: A simulated 3D trajectory (blue) is shown together with predictions from 3 different camera-view 2D inputs. Each view has a different color: yellow, red, and green.

the synthetic data can be flawless, which is not true for the image trajectories estimated by the pretrained ball (shuttle) detection model in the case [105]. Thus, another crucial step taken to improve generalization towards the noisy real image trajectories is that the synthetic input 2D trajectories are subjected to three types of augmentation during training.

1. All trajectories are subjected to a small amount of Gaussian noise.

$$u_i = u_i + \mathcal{N}\epsilon_s \forall u_i \in U \in \mathbb{R}^{T\times 2}$$

2. A small fraction of the image coordinates in particular trajectories are assigned a much larger random noise, which simulates significant input coordinate outliers, sometimes present in the extracted image trajectories.

3. Removing a small fraction of shuttle coordinates in a number of training trajectories. This stimulates failed detections by the shuttle extraction model.

This is in addition to the noise added through distorting the image by including distortion parameters in the sampling procedure of the camera parameters, which adds "realism" to the synthetic trajectories. The cross-view consistency loss complements

the data augmentation excellently as strongly augmented, and thus, flawed input trajectories are compared and aligned with clean trajectories from different camera angles. Thus, the model should implicitly be able to infer information on the effect of missing/distorted input coordinates.

7.2.4 Model Evaluation

To evaluate the model, two different sets of trajectories are defined or selected:

1. The 40000 synthetically created trajectories for all primary stroke types present in [8, 115, 114]. All shots are projected to image coordinates for six unique sampled camera settings. 32000 Shots are used for training, and the remaining 8000 for testing, corresponding to a (80/20) train/test split. The splitting is random with seed 12.

2. The ground truth annotations and image shuttle trajectories from the [114] dataset. The synthetic data allows for calculating both reconstruction (3D) and reprojection (2D) distance errors since both ground truth 2D and 3D data are available.

Shuttle Trajectories Evaluation Metrics

The lifting model uses the 2D coordinates and court corners (extracted from the videos) to predict initial conditions for inference. Subsequently, the 3D trajectories are found using the numericalODE solver to find the discretized solution to the projectile equations of motion. The reprojected image coordinates can be found using the perspective transformation of the pinhole camera model. The camera parameters for each match (i.e., camera setting) are estimated using the badminton court's known world coordinates and corresponding image coordinates following a similar procedure for calculating the homography map in chapter 4. The corresponding coordinates comprise the court corners from Figure 4.2 and manually annotated net-pole coordinates. Given the *real* image trajectory (U) and predicted image trajectory ($\hat{U}$), the trajectory-average reprojection error $E_{RE}^{(n)}$ for the n^{th} trajectory in a dataset, can be calculated as the Euclidian distance between the coordinates over span of the trajectory:

$$E_{RE}^{(n)} = \frac{1}{T}\sum_{t=1} ||u_t - \hat{u}_t||_2 = \frac{1}{T}\sum_{t=1}^{T} \sqrt{(u_t - \hat{u}_t)^2}, \tag{7.18}$$

where $u_t \in U \in \mathbb{R}^{T\times 2}$ is the image coordinate at time step t belonging to the input trajectory U, and $\hat{u}_t$ is the predicted image coordinate at time step t.
Hence, the mean reprojection error $\bar{E_{RE}}$ can be found by averaging over all N trajectories in the tested dataset.

$$\bar{E}_{RE} = \frac{1}{N}\sum_{n=1}^{N} E_{RE}^{(n)} \tag{7.19}$$

Similarly, the median reprojection error $\tilde{E}_{RE}$ over a dataset can be found.
The E_{RE} does provide insight into the accuracy of the predictions and, thus, the capabilities of the lifting model. However, as will be shown later, in many instances, the reprojection error can be a misleading metric for the actual quality of a prediction. Therefore, additional metrics will complement the E_{RE}. Specifically for the synthetic dataset, the (trajectory-averaged) 3D reconstruction error E_{rec} is given by:

$$E_{rec} = \frac{1}{T}\sum_{t=1} ||X_t - \hat{X}_t||_2, \tag{7.20}$$

where $||\cdot||_2$ is the Euclidean distance, and the X and $\hat{X}$ are ground truth and predicted 3D world space trajectories, respectively. Like for Eq 7.19 the mean reconstruction error $\bar{E}_{rec}$ can be found. Furthermore, for the synthetic trajectories, the Mean Absolute Error (MeanAE) between the ground truth and predicted initial condition γ_0 and $\hat{\gamma_0}$ determine the mean absolute difference between each component of the initial condition vector $\gamma_0 = (x_0, y_0, z_0, v_{x0}, v_{y0}, v_{z0})$.
Finally, for real datasets a consistency error E_C is utilized. E_C follow the same procedure as the reprojection error for generating the 3D prediction $\hat{X}$ but now, instead of projecting the prediction to image coordinates with the match-specific camera parameters, the camera position is subjected to 8 small perturbations, which rotates or elevates the camera position by a few degrees.
The principal idea of the perturbations is shown in Figure 7.4c. The predicted 3D trajectories are projected to image coordinates using the new perturbed camera positions (the intrinsic camera parameters are kept the same). The new image trajectories are then fed back into the TrajTrans model to predict new initial conditions and, after subsequent simulation based on each, new perturbed 3D predictions $\hat{X}_p$. $\hat{X}$ now function as pseudo ground truth 3D data, while $\hat{X}_p$ functions as the 3D trajectory predictions. Thus, the mean consistency error $\bar{E}_C$ and median consistency error $\tilde{E}_C$ for the

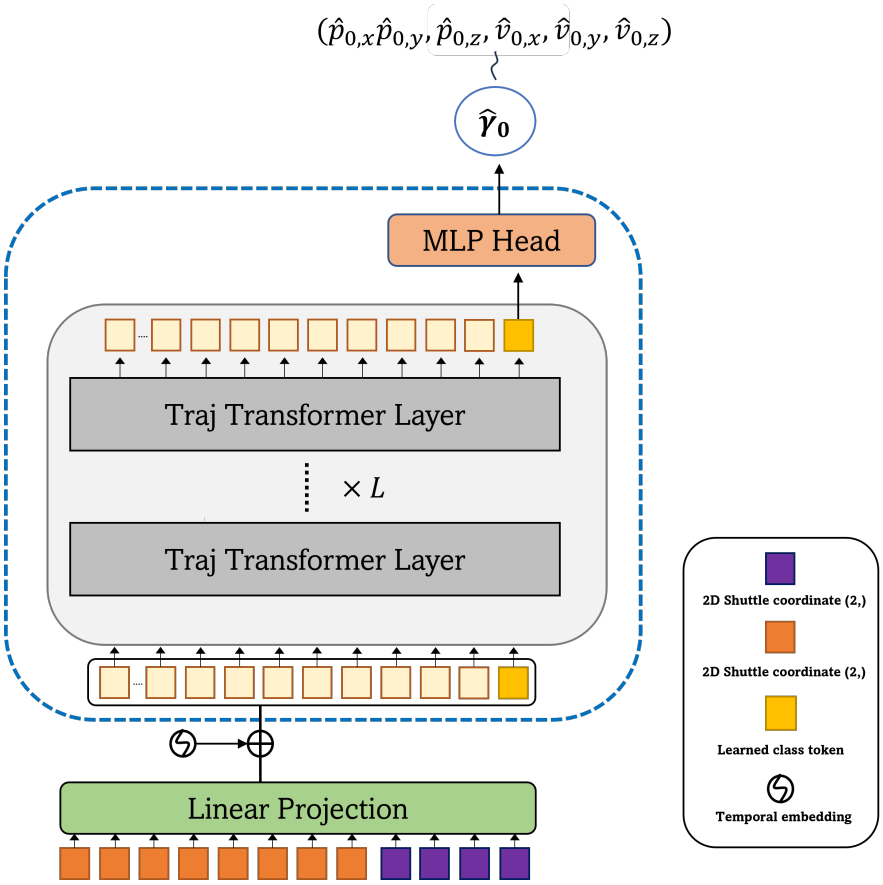


Figure 7.6: Transformer Model Architecture.

real dataset can be calculated between $\hat{X}$ and $\hat{X}_p$ exactly like X and $\hat{X}$ for E_{rec} for the synthetic dataset. This metric, however, has several limitations compared to E_{RE}. Specifically, the error is only concerned with the consistency between the initial and perturbed predictions. Thus, the quality of the initial prediction does not influence the E_C as long as the perturbed prediction is consistent with the original prediction.

7.3 Shuttle Experiments

7.3.1 Dataset and Implementation Details

The lifting model is evaluated on synthetic data and actual broadcasted shot sequences from shuttleset22. The synthetic test set comprises 20% of the generated data, corre-

Table 7.3: Traning augmentation parameters.

Augmentation fraction	Small noise	Outlier noise	Outlier rate	Removal rate
0.3	0.002	0.5	0.05	0.2

sponding to 8000 3D initial conditions γ_0 and 48000 input "image" trajectories.
The augmentations to the synthetic data during the training are specified by the parameters shown in Table 7.3. The augmentation fraction is the number of trajectories where any augmentation is applied to a trajectory. The Small and outlier noise parameters are the standard deviation of the 0-mean Gaussian noise types added to the augmented data; 0.002 corresponds to about 2 pixels for normalized data, whereas 0.5 corresponds to an augmentation by ~ 450 pixels, i.e., 68% of trajectories with outliers applied are augmented by 0 to 450 pixels. The outlier rate is the number of shuttle coordinates where the outlier augmentation is used. Hence, the number of coordinates added outlier noise is $0.3 \cdot 0.05 = 1.5\%$.
The Shuttleset22 [114] matches and annotated shot segmentations are used for real-world testing of the 2D to γ_0 (3D) lifting model. WASB estimates the image trajectories [105], a model using HRnet as a backbone. All available 47 matches are used for testing, constituting 41000 test samples. The matches are recorded with a static camera from an overhead *broadcast* view behind one of the backlines, with a resolution of 1280x720 pixels and a frame rate of 30 fps.
The image trajectories are zero-padded to have a length of $T_{max} = 100$ frames to create inputs of equal length. The image resolution 1280×720 scales down trajectory and court coordinates. As shown in the model visualization in Figure 7.6, the court corners are concatenated to the models as additional points in the temporal dimension. This approach is perhaps not intuitive, and court dimensions could be incorporated in other ways, such as an additional input channel for each trajectory. However, testing different methods showed no significant difference in performance, so this method was selected for its simplicity. The TrajTrans for is implemented with $L = 5$, $heads = 8$, $dim_{head} = 8$, $D_L = 256$, Dropout $= 0.3$.

Synthetic Evaluation Table 7.4 shows the performance metrics of different models on both the real-world and synthetic test data. The real-world data are trajectories from all non-failed annotations (unknown shots and shots with a duration un-

der five frames removed). Unsurprisingly, the results show that the models consistently perform the best on the synthetic trajectories. This is expected since a $\hat{\gamma_0}$ exists, which can perfectly follow the simulated synthetic 3D trajectories. In practice, this is not easy as the model has to capture a representation that can do this based only on the image (2D) trajectory for $\sim$ 200000 different and augmented trajectories. If the dataset was narrowed down to $\sim$ 50 samples (with no augmentations), then the model could estimate the trajectories perfectly. This approach would mirror the optimization-based approaches like [72], just on the initial conditions and not the 2D reprojection loss.
On the synthetic data, TrajTrans performs the best with a reprojection error of 20.4 pixels. Compared to Paper 3, this extended training approach appears to perform worse. However, that is not actually the case. It is a consequence of the synthetic dataset with the different viewing angles and greater shot diversity being a much more challenging dataset. The results for the exact FFN model from the paper are also shown, with an average Euclidean pixel distance of 65.0px. This increased complexity of the artificial data results in better relative generalization of the models on the real-world dataset, and the capabilities of TrajTrans model be shown by comparing to the previous baseline.

Real-world Evaluation The TrajTrans model can limit the mean reprojection error to 52.0px. Again, the results do not appear overly remarkable, but when compared to the different lifting models, TrajTrans performs significantly better. Moreover, poor data input and poorly segmented shot sequences can most likely explain the non-optimal reconstruction. This will be elaborated on in the discussion section. The median reprojection $\tilde{E}_{RE}$ of TrajTrans is 23.6px. Thus, a significant difference between the mean and median of reprojection error is observed for the real-world data. This indicates that a minority of outliers mainly contribute to the larger error. While the majority of the predicted trajectories have a much lower reprojection error E_{RE}. The differences between mean and median values on the synthetic data are smaller, showing fewer outliers. The model can be evaluated using reconstruction E_{rec} error on the synthetic data, where an average distance (error) of 48 cm is observed. Finally, the consistency loss E_C is pretty consistent. The baseline models score around $1-2$m $\bar{E_C}$. Indicating that the models have benefitted from multiview training.

Table 7.4: Model performance on the two different test sets of trajectories. $\bar{E}_{RE}$ denotes the mean reconstruction error, $\tilde{E}_{RE}$ the median reconstruction error, $\bar{E}_C$ the mean consistency error, and $\tilde{E}_C$ the median consistency error. Finally, $\bar{E}_{rec}$ is the 3D reconstruction error.

	Image		World		
	$\bar{E}_{RE}$	$\tilde{E}_{RE}$	$\bar{E}_C$	$\tilde{E}_C$	$\bar{E}_{rec}$
TrajTrans					
Real	52.0 px	23.6 px	1.64 m	1.36 m	-
Synthetic	20.4 px	16.8 px	-	-	0.48 m
BiTCN					
Real	60.2 px	33.8 px	200 m	1.25 m	-
Synthetic	37.8 px	22.8 px	-	-	0.62 m
FNN					
Real	93.2 px	56.2 px	- m	-	-
Synthetic	65.0 px	46.6 px	-	-	0.87 m
GRU					
Real	111.0 px	84.4 px	1.91 m	1.42 m	-
Synthetic	70.8 px	58.9 px	-	-	1.16 m
LSTM					
Real	107.7 px	89.2 px	1.67 m	1.37 m	-
Synthetic	78.7 px	63.0 px	-	-	1.45 m

Outlier assesment As observed in Table 7.4, all models show a large gap between the mean and median reprojection error on the real data reprojection error. This suggests that the distribution of the trajectory-averaged pixel distance is skewed, and a small fraction of outliers strongly influence the average. In Figure 7.7 E_{RE} is plotted for the shuttleset22 data, verifying the hypothesis, as it can be observed that the distribution has a long tail of large E_{RE}, which can be considered outliers. Visual inspection shows that the samples the model scores poorly on fall into distinct categories.

A significant number of the poorly scoring samples can be attributed to poor detection of the shuttle by WASB [105], which often leads to the predicted shuttle trajectory being better than the reference detected image trajectory, but still getting a larger average reconstruction error. Likewise, there are inaccurate annotations of the player hitting the shuttle, which causes the model to predict nonsensical 3D trajecto-

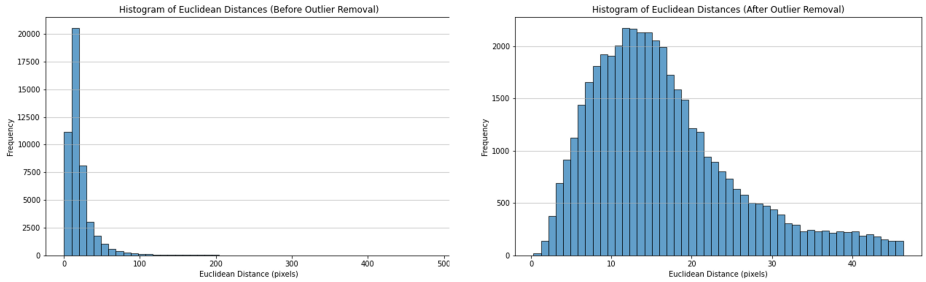


Figure 7.7: The two plots show the reprojection error distribution before and after removing clear outliers from the real-world dataset.

ries.

Removing a small fraction of the worst performing trajectories, specifically $3500/41000 \simeq 8.5\%$, leads to a much less skewed distribution of errors where the mean and median are only $\sim$ 3 pixels apart. Thus, this could be one logical way to explain the outliers. Granted, some of the removed samples were not overly flawed input trajectories but simply poor predictions by the model. An explanation for these samples was not present in the synthetic data and therefore, the models struggle to predict these samples accurately. These should, of course, not be removed as they represent the model's actual capabilities. However, it is impossible to differentiate between the two when filtering the dataset. The different cases are discussed with examples in subsection 7.3.2.

Intial condition evaluation The γ_0 predictions on the synthetic data also provide valuable insight into the model capabilities. The average absolute error for the initial position and initial velocity is shown in Table 7.5 in meters and meters per second, respectively. The model does well in estimating the starting position of the stroke in the x and z-coordinates, with the error $\sim 0.1m$ while the y-coordinate is, on average, 0.35m away from the true y-coordinate. This results in an average error distance of $\sqrt{(0.08m)^2 + (0.35m)^2 + (0.10m)^2} \simeq 0.37m$, which is an improvement compared to results on the tracknetV2 tennis dataset from Paper 3. However, the errors in the predictions of the initial velocities are larger on average for the badminton data, with an error, i.e., model uncertainty above 1m/s in y and z directions. This could partially be caused by the synthetic dataset being more complex and challeng-

Table 7.5: Average absolute differences and standard deviations on initial conditions on synthetic test data.

γ_0	Direction	Absolute Error
Position (x_0)	x	0.08 ± 0.07 m
	y	0.35 ± 0.33 m
	z	0.10 ± 0.09 m
Velocity (v_0)	x	0.90 ± 0.79 m/s
	y	3.57 ± 3.62 m/s
	z	1.21 ± 1.18 m/s

ing to model compared to Paper 3.
Another likely explanation could be the relative velocities compared to the distance covered are much higher for badminton than for tennis. Hence, the model is generally better at estimating the initial position than the initial velocity.
Consistent with the equivalent experiment in Paper 3, the y-direction still causes the most significant error for both the initial velocity and the initial launch position predictions. This is caused by the camera's optical axis being closely aligned with the y-axis in the world space coordinate system for most camera positions. However, compared to the results Paper 3, the relative error in the y-direction compared to the total error is reduced for both the velocity and position estimates. This suggests including more variety in the sampled camera positions, i.e., diversity in the viewing angle, for the synthetic data to help reduce the ambiguity in the direction of the optical axis.
Observing the 3D trajectories produced from real-world 2D input reveals some additional tendencies: The model has added difficulties determining the proper 3D start position of the shot when the trajectory starts from the side of the court furthest away from the camera. This is again due to the model finding predictions in the direction of the optical axis difficult. On the side, of the court, the furthest away from the player is a single pixel corresponding to a much larger real-world distance. This manifests in issues for not only the model-making prediction but also for the shuttle detection model WASB. Thus, it is logical that the compounding effect of these factors inhibits the model predictions further away from the shuttle.

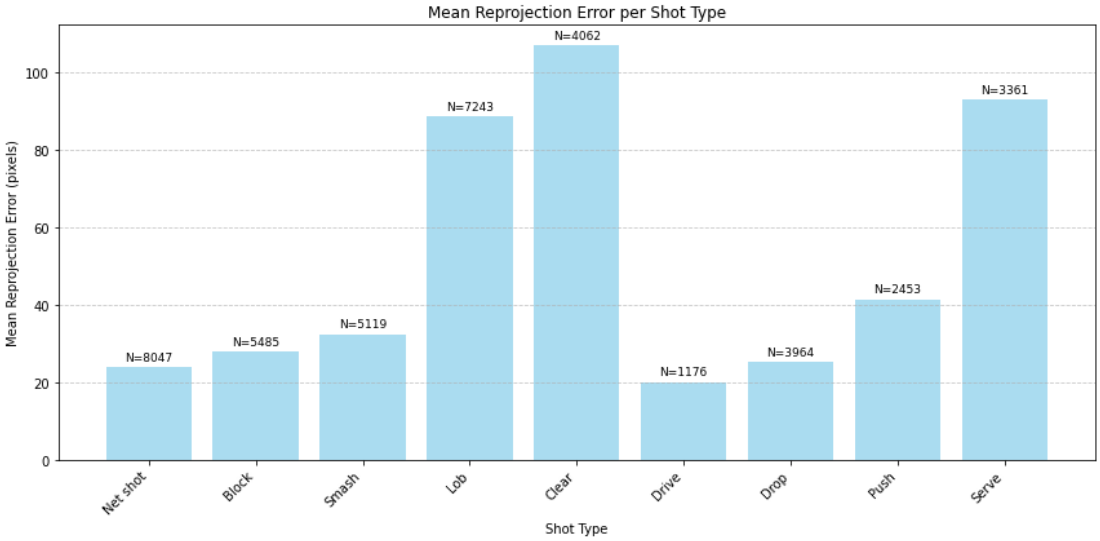


Figure 7.8: Bar plots showing the reprojection error of the different shot types of real-world matches. Lobs, Clear, and Serves, in particular, appear more difficult for the model to predict than the rest.

Shotwise reconstruction performance The average trajectory-wise reconstruction error is shown in the Figure 7.8 for the different shot types in shuttleset22 [114]. Most shot types have an average reconstruction error clearly below 40 pixels, which for a 1280×720 image resolution corresponds to a relative pixel error of 2.7%. However, the serve, clear, and lob all have errors above 100 pixels. Limitations of the physical modeling of the trajectories could cause this. The lobs and clears (and long serves) have the longest trajectories, which would result in the most significant reprojection errors if the modeling of the trajectories does not exactly capture the real shuttle trajectories. The limitations and approximation behind the modeling of the shuttle trajectories are further discussed in the limitations section. Additionally, for serve, in particular, the footage of the segmented shots is less consistent. Sometimes, the broadcast contains close-ups of the players, which results in less consistent detection of the shuttle image trajectory.

Trajectory-based Shot classification One way to estimate the quality of the inferred 3D trajectories is to perform shot recognition using those same trajectories. In contrast to chapter 5, where action recognition involved multiple modalities, the approach here relies solely on shuttle trajectories. Specifically, the TrajTrans model

Table 7.6: Showing performance metrics of comparing 2D image trajectories vs predicted 3D trajectories for shot classification on the shuttleset22 dataset. The TrajTrans model is here predicting the shot types instead of the initial conditions.

Input	Acc (%)	F1-M
2D Trajectory	71.4	59.7
2D Traj + Court	73.1	61.5
3D Trajectory	**73.4**	**63.2**

was trained on the train-test match split of the ShuttleSet22 dataset [114] to estimate action classes instead of initial conditions. It was trained with three different input types: (1) 2D WASB-detected [105] image trajectories, (2) 2D image trajectories plus 2D court corners, and (3) the inferred 3D trajectories.
As shown in Table 7.6, the 3D trajectory input achieves the best accuracy and F1-macro (F1-M) scores, yet only by a small margin—0.3% in accuracy and 1.8 in F1-M. In comparison, the TemPose model (which uses multi-modal input features) achieves an accuracy greater than 90% and an F1-M of about 78.8 on the same train-test split Table 5.7.
One interpretation is that the inferred 3D trajectories might be insufficiently accurate when coupled with the 2D input data. However, other evaluation methods have shown promising results, suggesting this may not be the primary issue. Another possibility is that TrajTrans is not the optimal architecture for shot classification. Yet, most experiments thus far suggest that choosing GCN, TCN, or transformer-based models does not drastically affect performance. Therefore, the most likely explanation is that the shuttle trajectories alone do not provide enough discriminative information to distinguish shot classes. The ablation study in Table 5.4 supports this idea, although the underlying reasons remain unclear. Intuitively, shuttle trajectories should contain nearly as much discriminative information as the skeleton motion, but that does not appear to be the case. Further studies are needed to identify the root cause of this discrepancy.

7.3.2 Qualitative Assessment of 3D Trajectories

In Figure 7.9, the image coordinates of the shuttle extracted by WASB [105] (blue) and the model's predicted 3D trajectory with its reprojection (orange) are shown

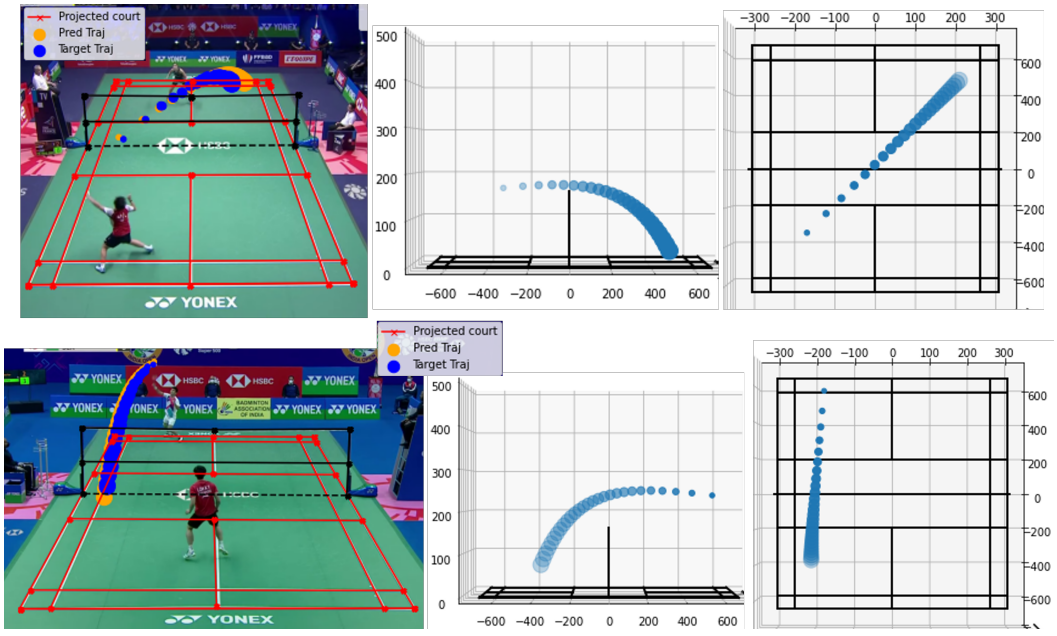


Figure 7.9: WASB image trajectory and the reprojected predicted 3D shot examples. Left: WASB image trajectory (blue) and the reprojected predicted 3D shot (orange). Middle: The predicted 3D shot viewed from the side. Right: The predicted 3D shot viewed from above.

alongside two views of the corresponding 3D prediction. These examples indicate that the reprojection closely follows the input trajectory and that the 3D prediction appears reasonable. Both examples are positive cases, where the model accurately estimates the 3D trajectory.

By contrast, Figure 7.10 highlights the challenges and limitations of the current lifting model. In the first row, two high-quality 3D predictions demonstrate the TrajTrans model's capabilities: From fairly challenging input trajectories, the model accurately predicts $\hat{\gamma}_0$, which by visual inspection seems correct. In the second row, the model still makes correct predictions despite compromised image trajectories. In the example on the left, the input trajectory has no glaring outliers but does contain missing coordinates. Because missing detections do not affect the reprojection loss, this sample has a reprojection loss close to zero, consistent with a high-quality prediction. On the right, however, the input includes several missing coordinates and outliers (e.g., a body part is occasionally misidentified as the shuttle). Despite these issues, the 3D trajectory looks accurate when visually inspected.

In these cases, the outlier coordinates do affect the E_{RE} reprojection error. As a re-

sult, E_{RE} may misrepresent prediction quality. This outcome is fairly common and shows that the model generally makes logical and sensible predictions even if E_{RE} exceeds 40 pixels. Visual inspection further suggests that certain match conditions (e.g., camera angle, lighting, or saturation) were not well-represented in the WASB [105] training data, leading to lower-quality 2D detections. It is also clear that the model would benefit significantly from better 2D inputs. Although the valid data channel and training augmentations enhance robustness to flawed inputs, there are scenarios where accurate predictions are impossible if the 2D detections are poor. Thus, improving the 2D detection pipeline is the most critical next step. Another potential solution is outlined in subsection 7.4.3.

Rows 3, 4, and 5 in Figure 7.10 show cases where the model's predictions are incorrect but appear logical given the input data. In Row 3, for example, the first few image frames fail to detect the shuttle correctly. In these situations, the model often predicts a backcourt launch position away from the camera, even when the player nearest to the camera makes the stroke. If one relies solely on the (faulty) 2D input, it becomes impossible—even for a human observer—to infer the correct starting position. Consequently, the model's mistake is unsurprising.

Regarding detection improvements, note that the shuttle travels rapidly after contact with the racket. At a launch speed of around 250km/h and a frame rate of 30fps, the shuttle moves about 2.3 meters per frame. It is, therefore, challenging to accurately detect the start of a shot. Training a detection model using high-speed cameras (60–120fps) could improve reconstruction accuracy, though this remains a conjecture as no such experiments were conducted to verify this claim in the thesis.

Another approach could incorporate side-of-court information within the TrajTrans model and train it on more samples where the initial frames of the trajectory are explicitly removed, thereby teaching it to handle missing start positions more effectively.

In another Row 3 example, the model again places the launch position on the side farthest from the camera. Although the predicted 3D trajectory fits well with the (faulty) 2D input, it is entirely inaccurate. E_{RE} remains low but does not capture the inaccuracy—demonstrating a limitation of using reprojection error in isolation. Meanwhile, E_C also fails to offer helpful information, emphasizing the importance of visual inspection. In Row 4, the annotated timestamps for the shot segments are incorrect, producing unreasonable input trajectories. This leads the model to fail, demonstrat-

ing the necessity of precise segmentation for reliable 3D reconstructions. Although segmentation was not a primary focus of this thesis, it remains crucial for accurate information extraction in badminton.
Row 5 contains more extreme examples of poor input trajectories where the model cannot accurately predict γ_0. On the right, the detection is essentially random noise, illustrating how severely flawed 2D detections can be. On the left example, there are no initial 2D detections, outliers, and slightly incorrectly annotated shot segmentation. Given these inputs, the model's erroneous prediction is still understandable.
Finally, the last row in Figure 7.10 shows suboptimal model predictions despite having relatively good input trajectories. In the left example, the hitting position is correctly estimated, but the shot's direction is off, indicating that the model still has room for improvement. In the right example, the model misidentifies which side of the court the shuttle is on—a common issue when the input trajectory is near the net or the hitting side is ambiguous due to the camera perspective. Including player positions or additional side-of-court information as part of the model, input could mitigate this issue, as also suggested for the Row 3 examples.

7.3.3 Ablation Studies

An ablation study explored the effect of using the court corners as input along with the image trajectory. The results presented in Table 7.7 show a significant performance increase in all metrics when the input includes the court corners. This suggests knowing the court position and size (in image coordinates) yields more accurate trajectories. The reason for this is likely two-fold. First, the court corners are static in the world coordinates. Hence, the changing image position of the corners provides the model with information about the camera position and trajectory perspective. Second, the court corners provide a reference boundary for the model, indicating limits for the shot placement.
A second ablation study explores the effect of multi-view data, data augmentation, and regularizing view consistency loss. The results are shown in Table 7.8. The first observation is that the trained model with Data aug + Multiview + View Consistency loss (VCL) performs best on both the real and the synthetic datasets. However, the model that leaves out the augmentations but still trains on the multiview data and views consistency loss experiences a significant drop in performance. On the

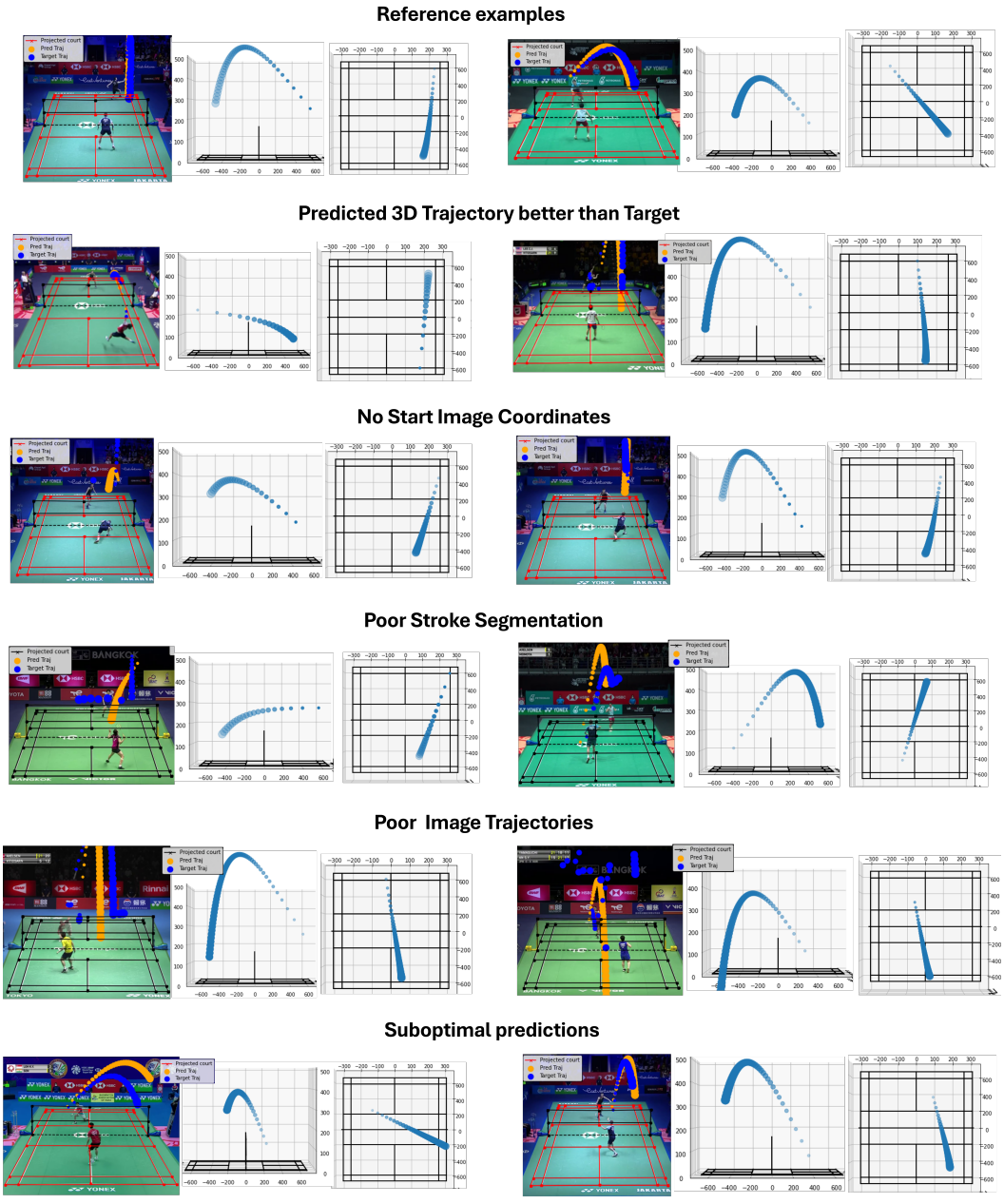


Figure 7.10: Inspection of the 3D predictions from real-world shot sequences. This figure presents six rows of prediction examples, each illustrating different attributes and limitations of the TrajTrans model. Every row contains two examples. In each example, the left image shows the first frame of the segmented shot video, with the input 2D trajectory (blue) and the reprojected 3D prediction (orange). The middle image presents the side view of the 3D prediction, while the rightmost image shows the top view.

Table 7.7: Ablation study of input features on all real-match trajectories.

Input	$\bar{E}_{RE}$ **(px)**	$\tilde{E}_{RE}$ **(px)**	$\bar{E}_C$ **(m)**	$\tilde{E_C}$ **(m)**
2D Trajectory	158.3	89.1	1.78	**1.36**
2D Trajectory + corners	**52.0**	**23.6**	**1.64**	**1.36**

Table 7.8: Performance evaluation on the Synth vs Real (Sset22) for different synthetic training initiatives. Data Aug: Training Data Augmentation, MvD: Multiview Data, VCL: View Consistency Loss

Modality			Performance	
Data Aug	MultivD	VCL	Synth:$\bar{E}_{RE}$ (px)	Real: $\bar{E}_{RE}$ (px)
			32.1	70.0
✓			92.8	110.8
	✓		25.9	58.3
	✓	✓	56.7	77.5
✓	✓	✓	20.4	52.0

other hand, the model performs well, having not experienced any regularization or training data augmentation. The multiview – data – trained model outperforms the other single-view trained model by $\Delta\bar{E}_{RE} > 6$px for both datasets. From this, it is reasonable to conclude that multiview data improves 3D recognition performance to some degree.

Models trained with data-augmented data – without VCL – also perform terribly. This indicates a balance between augmenting the data and regularizing for view consistency is reached for the best-performing model. Since the regularization amplitude and data augmentation parameters were kept the same for all models in the experiment, it is possible that the model could reach a better performance with either data augmentation or VCL regularization by tweaking the parameters. Nevertheless, the model trained with (Data aug + Multiview + VCL) performing the best suggests the training method is valid for improving shuttle reconstruction performance.

7.4 Discussion

7.4.1 Analytical Prospects

Extracting 3D player shots enables storing and examining physical properties such as speed, hitting position, and shot placement, which can vary between players and

playing styles. These properties provide rich opportunities for analysis, potentially informing training strategies, performance comparisons, and coaching methodologies. As more 3D trajectory data is collected, conducting more comprehensive and robust investigations into player-specific traits and competitive behaviors will become possible.

7.4.2 Limitations

A key limitation of the current 3D trajectory estimation pipeline is its dependence on 2D shuttle detection models. For instance, WASB [105] can misdetect or completely miss the shuttle under challenging lighting conditions or, when the shuttle moves too quickly, leading to unreliable or missing initial frames. Since the earliest segment of the shuttle's flight is crucial for estimating initial position and velocity, inaccuracies in these frames propagate through the model and reduce final prediction accuracy.
A related issue arises from side ambiguity when the camera viewpoint allows the shuttle's 2D path to be interpreted from multiple perspectives. Although incorporating a side embedding partially mitigates this problem, it remains especially troublesome if early trajectory coordinates are absent or mislabeled (see Figure 7.10, row 3).
Another set of limitations stems from the simplified physical assumptions in the Initial Value Problem (IVP) framework. Abrupt trajectory changes, such as hitting the net, are not modeled correctly. Moreover, factors like wind, spin, and the non-constant nature of the drag coefficient are not considered, thus reducing the realism of the simulated or inferred trajectories.

7.4.3 Future Work

The current 3D reconstruction framework can be improved in several directions. One avenue involves enhancing view consistency by incorporating contrastive or momentum-based learning strategies. Leveraging NeuralODE to optimize the initial value problem on real data through reprojection loss could further refine the transition from synthetic to actual match conditions.
Another critical direction lies in denoising or reconstructing 2D shuttle trajectories that are missing early frames, possibly through a masked autoencoder approach similar to what was done for skeleton data in chapter 5.

Another limitation could be addressed by extending the model architecture to incorporate information about the starting side in the model. This information is implicitly known from the player position and shot segmentation. Hence, including it in the model would not exploit difficult-to-retrieve information. Still, it would likely allow the model to avoid confusion about the direction of a shot with an ambiguous viewing angle or flawed input.
Additionally, modeling more nuanced aerodynamic factors – such as spin, wear-and-tear on the shuttle, and varying drag coefficients – could bring simulated conditions closer to real-world scenarios and improve accuracy. Finally, training on a wider variety of synthetic camera parameters, followed by targeted fine-tuning on real data, may bolster the robustness of the model to different match setups and camera perspectives.

7.5 Chapter Conclusion

In this chapter, the TrajTrans transformer model was trained on synthetic 2D trajectory and initial-condition pairs to estimate real-world 3D shuttle trajectories in badminton matches. The model achieves promising results, reflected by low reprojection errors and favorable consistency metrics on synthetic multiview datasets. Visual inspections further support the idea that the generated 3D trajectories appear realistic. Nevertheless, the model's performance is hindered by missing or inaccurate 2D detection of the shuttle in the earliest frames and ambiguity when the camera angle allows multiple interpretations. These issues most notably affect the estimation of the shuttle's initial velocity. Despite these limitations, TrajTrans demonstrates the potential to be a cost-effective and efficient solution for capturing 3D shuttle trajectories in real-world settings.
Relatively simple yet strategic improvements – such as refining data preprocessing and enhancing physical modeling – could yield significant gains in accuracy and overall reliability.

Chapter 8

Conclusion

This thesis has investigated three areas in computer vision in the context of badminton analytics: *Stroke recognition*, *Stroke forecasting*, and *3D shuttle reconstruction*. Focusing on skeleton motion, shuttle trajectories, and domain-targeted neural architectures, the work has demonstrated that computer vision and machine learning approaches can automate nuanced match analyses and generate insights at a level of detail previously only accessible through excessive manual effort or advanced multi-camera systems.
The following sections summarize key findings, highlight limitations, and propose future directions leveraging collaboration between machine learning researchers and sports practitioners.

8.1 Main Takeaways

A primary conclusion from the *stroke recognition* experiments is that skeleton-based representations, especially those integrating bone, joint, and shuttle information, surpass purely image-centric solutions in detecting fine-grained badminton strokes. Models such as the proposed TemPose capitalize on these compact – but discriminative – action-focused features to ease the task of differentiating smashes, clears, net shots, or drives. The recognition performance is close to human-level recognition. It can therefore reduce heavy annotation requirements and establish a realistic opportunity for large-scale, in-depth analyses of player patterns and stroke usage.
In *stroke forecasting*, the thesis explored how partial or complete knowledge of past

strokes informs the prediction of the next action in a rally. While players sometimes disguise upcoming shots, results indicate that even partial skeleton sequences and player context can yield meaningful next-stroke probability estimates. These insights can expand real-time coaching feedback, improve strategic preparation, and provide dynamic information for match commentary. The initial results in stroke forecasting also open opportunities for future techniques that adapt to specific match situations or player tendencies.
Finally, *3D shuttlecock reconstruction* was pursued using a monocular framework and synthetic data. A simplified aerodynamic model was employed to generate (2D, 3D) training pairs utilizing sampled camera parameters, removing the reliance on expensive ground truth 3D data. Although contingent on accurate 2D detection and a ballistic model of shuttle flight, the results demonstrated that a reconstruction pipeline can uncover shot-level information about speed, altitude, and flight path that coaches and analysts can leverage for a deeper understanding of the performance.

8.2 Limitations & Short-comings

The experimental findings and subsequent analysis have uncovered several shortcomings and limitations in relation to the investigated approaches.

Model Specialization vs. Practical Gains. Substantial effort was invested in designing or adapting neural network layers (e.g., TCNs, GCNs, or transformers) specifically for badminton tasks. Results generally showed that although such customization yields small performance differences, bigger effects stem from data factors – size, annotation quality, modality completeness – rather than fine architectural nuances. Hence, while model architectural improvements are welcome, data considerations appear to dominate the performance gains.

Data Size and Quality. Throughout, training efficacy proved sensitive to dataset scale and consistency. Where large, accurately labeled segments were available, model performance rose noticeably. In contrast, label noise, missing frames, or camera-angle inconsistencies often undermined the benefits of architectural refinements. Thus, encouraging standardized annotation protocols, collaborative data sharing, or construct-

ing comprehensive match repositories remains essential for robust, generalizable models.

Overlapping Modalities and Complementary Features. It became evident that skeleton data alone supported decent stroke classification and shuttle trajectories offered relevant information for certain tasks. However, fusing multiple input streams (e.g., skeleton + shuttle + court position) often led to incremental but consistent improvements. Each modality supplies subtle but unique differences, so more advanced and further domain-specific data integration can likely yield further gains.

Unsuccessful or Unfinished Approaches. Attempts at incorporating player Identification into the RallyTemPose stroke forecasting network did not yield the expected performance gain compared to player-unaware approaches. Different playstyles and player tendencies certainly exist. Hence, intuitively, the inclusion of such information should improve forecasting capabilities. The reasons for this are unknown. Perhaps it is caused by an imperfect model design. Another possibility is that not enough matches of all players are covered. Hence, the model captures different, more elementary shot selection rules instead.
Similarly, sophisticated block architectures that intertwined GCN and transformer layers in parallel streams provided only marginal recognition gains. The experiments indicate that broad leaps in performance may hinge more on better data, rather than highly specific model assemblies.

8.3 Future Prospects

The contributions outlined in this thesis form the basis of a badminton analytics pipeline capable of classifying strokes at scale, forecasting future actions, and reconstructing shuttle flights in 3D. Ongoing and prospective research may include:

- Fine-tuning and enhancement of aerodynamic model to close the gap between synthetic and real shuttle trajectories. (e.g., adding effect of gyro-spin and non-constant drag coefficient C_d)
- Improving the fidelity of 2D detection – for both shuttle and player skeletons – perhaps through higher-frame-rate cameras. To this end, developing a racket

detection model could prove useful. However, for the current image quality of broadcasted matches, this seems improbable, and hence, higher fidelity is needed here.

- Scaling up the datasets (in collaboration with sports organizations), fostering cross-organization unifying annotation standards.

As these capabilities mature, the sport of badminton will benefit from increasingly data-driven performance tracking, advanced practice drills informed by stroke forecasting, and fully realized 3D analyses of both player movement and shuttle motion, bridging the gap between raw footage and actionable knowledge of the game's technical and strategic elements.

8.4 Key Insights for Collaboration

Implementing advanced methods, as those described in this thesis, calls for targeted collaboration between *machine learning specialists* and *sports practitioners*. Domain experts – coaches, analysts, and experienced players – play a vital role in specifying which stroke details, shot parameters, etc., truly matter in coaching and match strategy planning and analysis. Their input ensures that the objective of the deep-learning methods remains meaningful. At the same time, educating and informing the domain users about the fundamentals of deep-learning methods is essential for fruitful collaboration.

One unfulfilled desire during this project would be the emphasis on field trials, where predicted strokes or reconstructed flights were tested against real player input to help refine data requirements (to fill persistent gaps). Since top athletes are typically very occupied, an initial target could be up-and-coming junior athletes instead.

A gap seems to exist between demonstrated machine-learning-based analytical capabilities and practical usability. In research, these methods are often in-between generic analysis and relevant analytical considerations due to the researcher's lack of domain knowledge. For this reason, players and coaches often find such tools, more tailored to a broader audience, inconvenient, if not redundant. Therefore, players and coaches can be reluctant to spend much effort developing concrete objectives for the use of autonomous machine-learning methods. Expectedly, that attitude will gradually change as AI-based analysis and services continuously integrate more and more

into society and daily life. Emphasizing intuitive visualization and interpretability of results for domain experts will be vital to encouraging this transition and adaptation. For example, suppose coaches are able to see and understand a model's stroke predictions and how specific frames or joints strongly influenced the classification. In that case, they can connect such input with their current understanding to provide crucial player feedback and integrate their new perspective into training regimes.

8.5 Overall Message

Through careful modeling of skeleton motion, stroke sequences, and ballistic shuttle flight, this thesis has demonstrated how machine learning can automate significant components of badminton analysis and reveal detailed motion and stroke dynamic insights.

While architectural refinements contributed incremental gains, the overarching lesson is that *data availability, consistency, and feature synergy* overshadow most other considerations for robust performance.

Improving data pipelines, fusing complementary modalities, and extending physical assumptions about shuttle or player motion all promise further progress. When implemented thoughtfully, these tools can empower athletes, coaches, and even spectators' experience.

References

[1] Abu Farha, Y. and Gall, J. (2019). Uncertainty-aware anticipation of activities. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0.

[2] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *ArXiv*, abs/1803.08375.

[3] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[4] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lucic, M., and Schmid, C. (2021a). Vivit: A video vision transformer. *CoRR*, abs/2103.15691.

[5] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. (2021b). Vivit: A video vision transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6816–6826.

[6] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization.

[7] Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., Schwarzschild, A., Wilson, A. G., Geiping, J., Garrido, Q., Fernandez, P., Bar, A., Pirsiavash, H., LeCun, Y., and Goldblum, M. (2023). A cookbook of self-supervised learning.

[8] Ban, K.-W., See, J., Abdullah, J., and Loh, Y. P. (2022). Badmintondb: A badminton dataset for player-specific match analysis and prediction. In *Proceedings of the 5th International ACM Workshop on Multimedia Content Analysis in Sports*, MMSports '22, page 47–54, New York, NY, USA. Association for Computing Machinery.

[9] Bertasius, G., Wang, H., and Torresani, L. (2021). Is space-time attention all you need for video understanding? In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning, ICML*

2021, 18-24 July 2021, Virtual Event, volume 139 of *Proceedings of Machine Learning Research*, pages 813–824. PMLR.

[10] Bin, Y., Chen, Z.-M., Wei, X.-S., Chen, X., Gao, C., and Sang, N. (2020). Structure-aware human pose estimation with graph convolutional networks. *Pattern Recognition*, 106:107410.

[11] Brumann, C., Kukuk, M., and Reinsberger, C. (2021). Evaluation of open-source and pre-trained deep convolutional neural networks suitable for player detection and motion analysis in squash. *Sensors*, 21(13):4550.

[12] Cai, T. T. and Ma, R. (2022). Theoretical foundations of t-sne for visualizing high-dimensional clustered data.

[13] Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., and Sheikh, Y. A. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[14] Carreira, J. and Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733.

[15] Chao, V., Jamsrandorj, A., Oo, Y. M., Mun, K.-R., and Kim, J. (2023). 3D Ball Trajectory Reconstruction of a Ballistic Shot from a Monocular Basketball Video. In *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE. ISSN: 2577-1647.

[16] Chappa, N. V. R., Nguyen, P., Nelson, A. H., Seo, H.-S., Li, X., Dobbs, P. D., and Luu, K. (2023). Spartan: Self-supervised spatiotemporal transformers approach to group activity recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 5158–5168.

[17] Chen, H.-T., Tsai, W.-J., Lee, S.-Y., and Yu, J.-Y. (2012). Ball tracking and 3D trajectory approximation with applications to tactics analysis from single-camera volleyball sequences. *Multimedia Tools and Applications*, 60(3):641–667.

[18] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. (2019). MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

[19] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6572–6583, Red Hook, NY, USA. Curran Associates Inc.

[20] Chen, Y., Shen, C., Wei, X.-S., Liu, L., and Yang, J. (2017a). Adversarial posenet: A structure-aware convolutional network for human pose estimation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1221–1230.

[21] Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., and Sun, J. (2017b). Cascaded pyramid network for multi-person pose estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7103–7112.

[22] Chen, Y., Zhang, Z., Yuan, C., Li, B., Deng, Y., and Hu, W. (2021). Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13359–13368.

[23] Cheng, B., Xiao, B., Wang, J., Shi, H., Huang, T. S., and Zhang, L. (2020). Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5385–5394.

[24] Chi, H.-G., Ha, M. H., Chi, S., Lee, S. W., Huang, Q., and Ramani, K. (2022). Infogcn: Representation learning for human skeleton-based action recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20154–20164.

[25] Chi, S., Chi, H.-g., Huang, Q., and Ramani, K. (2024). Infogcn++: Learning representation by predicting the future for online skeleton-based action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14.

[26] Chu, W.-T. and Situmeang, S. (2017). Badminton video analysis based on spatiotemporal and stroke features. In *Other Conferences*, pages 448–451.

[27] Chung, F. (1997). *Spectral Graph Theory*. Number nr. 92 in CBMS Regional Conference Series. Conference Board of the Mathematical Sciences.

[28] Contributors, M. (2020). Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose.

[29] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3844–3852, Red Hook, NY, USA. Curran Associates Inc.

[30] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, page 8.

[31] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

[32] Dong, L., Li, D., Li, S., Lan, S., and Wang, P. (2019). Tai chi action recognition based on structural lstm with attention module. In *Other Conferences*.

[33] Dormand, J. and Prince, P. (1980). A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

[34] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[35] Du, Y., Wang, W., and Wang, L. (2015). Hierarchical recurrent neural network for skeleton based action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1110–1118.

[36] Duan, H., Zhao, Y., Chen, K., Lin, D., and Dai, B. (2022). Revisiting skeleton-based action recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2959–2968.

[37] Dwivedi, V. P. and Bresson, X. (2021). A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*.

[38] Ertner, M. H., Konglevoll, S. S., Ibh, M., and Graßhof, S. (2024). Synthnet: Leveraging synthetic data for 3d trajectory estimation from monocular video. In *Proceedings of the 7th ACM International Workshop on Multimedia Content Analysis in Sports*, MMSports '24, page 51–58, New York, NY, USA. Association for Computing Machinery.

[39] Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., and Feichtenhofer, C. (2021). Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835.

[40] Farha, Y. A., Richard, A., and Gall, J. (2018). When will you do what? - anticipating temporal occurrences of activities. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5343–5352.

[41] Farin, D., Krabbe, S., De With, P. H. N., and Effelsberg, W. (2003). Robust camera calibration for sport videos using court models. In Yeung, M. M., Lienhart, R. W., and Li, C.-S., editors, *Storage and Retrieval Methods and Applications for Multimedia 2004*, volume 5307, pages 80–91, San Jose, CA. SPIE.

[42] Fazio, M., Fisher, K., and Fujinami, T. (2018). Tennis ball tracking: 3-d trajectory estimation using smartphone videos. *Department of Electrical Engineering, Stanford University*.

[43] Feichtenhofer, C., Fan, H., Malik, J., and He, K. (2019). SlowFast Networks for Video Recognition. arXiv:1812.03982 [cs] version: 2.

[44] Gammulle, H., Denman, S., Sridharan, S., and Fookes, C. (2019). Forecasting future action sequences with neural memory networks. In *British Machine Vision Conference*.

[45] Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. (2021). Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*.

[46] Ghosh, A., Singh, S., and Jawahar, C. V. (2018). Towards structured analysis of broadcast badminton videos. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 296–304.

[47] Guo, C., Zuo, X., Wang, S., Zou, S., Sun, Q., Deng, A., Gong, M., and Cheng, L. (2020). Action2motion: Conditioned generation of 3d human motions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2021–2029.

[48] Guo, H., Agarwal, N., Lo, S.-Y., Lee, K., and Ji, Q. (2024). Uncertainty-aware action decoupling transformer for action anticipation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18644–18654.

[49] Hachiuma, R., Sato, F., and Sekii, T. (2023). Unified keypoint-based action recognition framework via structured keypoint pooling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22962–22971.

[50] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

[51] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. B. (2021). Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377.

[52] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[53] Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

[54] Huang, J. and Kang, H. (2024). 3d skeleton-based human motion prediction using spatial–temporal graph convolutional network. *International Journal of Multimedia Information Retrieval*, 13(3):33.

[55] Huang, L., Li, Y., Tian, H., Yang, Y., Li, X., Deng, W., and Ye, J. (2023). Semi-supervised 2d human pose estimation driven by position inconsistency pseudo label correction module. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 693–703.

[56] Ibh, M., Graßhof, S., and Hansen, D. W. (2024). A stroke of genius: Predicting the next move in badminton. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3376–3385.

[57] Ibh, M., Graßhof, S., Witzner, D., and Madeleine, P. (2023). TemPose: a new skeleton-based transformer model designed for fine-grained motion recognition in badminton. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 5199–5208. ISSN: 2160-7516.

[58] Ioffe, S. and Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org.

[59] Javadiha, M., Andujar, C., Lacasa, E., Ric, A., and Susin, A. (2021). Estimating player positions from padel high-angle videos: Accuracy comparison of recent computer vision methods. *Sensors*, 21(10):3368.

[60] Jeong, J., Park, D., and Yoon, K.-J. (2024). Multi-agent long-term 3d human pose forecasting via interaction-aware trajectory conditioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1617–1628.

[61] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., and Zisserman, A. (2017). The kinetics human action video dataset. *CoRR*, abs/1705.06950.

[62] Ke, Q., Bennamoun, M., Rahmani, H., An, S., Sohel, F., and Boussaid, F. (2020). Learning latent global network for skeleton-based action prediction. *IEEE Transactions on Image Processing*, 29:959–970.

[63] Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *Pre-print*. arXiv:1609.02907 [cs, stat].

[64] Komorowski, J., Kurzejamski, G., and Sarwas, G. (2019). Deepball: Deep neural-network ball detector. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications.

[65] Kulkarni, K. M. and Shenoy, S. (2021). Table tennis stroke recognition using two-dimensional human pose estimation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4571–4579.

[66] Li, M., Chen, S., Zhang, Z., Xie, L., Tian, Q., and Zhang, Y. (2022a). Skeleton-parted graph scattering networks for 3d human motion prediction. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T., editors, *Computer Vision – ECCV 2022*, pages 18–36, Cham. Springer Nature Switzerland.

[67] Li, T., Liu, J., Zhang, W., and Duan, L. (2020). Hard-net: Hardness-aware discrimination network for 3d early activity prediction. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 420–436. Springer, Cham.

[68] Li, Y., Wu, C.-Y., Fan, H., Mangalam, K., Xiong, B., Malik, J., and Feichtenhofer, C. (2022b). Mvitv2: Improved multiscale vision transformers for classification and detection. In *CVPR*.

[69] Li, Y., Zhang, S., Wang, Z., Yang, S., Yang, W., Xia, S.-T., and Zhou, E. (2021). Tokenpose: Learning keypoint tokens for human pose estimation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11293–11302.

[70] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

[71] Liu, J. and Liang, B. (2022). An Action Recognition Technology for Badminton Players Using Deep Learning. *Mobile Information Systems*, 2022:1–10.

[72] Liu, P. and Wang, J.-H. (2022). MonoTrack: Shuttle Trajectory Reconstruction From Monocular Badminton Video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, page 10.

[73] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[74] Liu, Z., Zhang, H., Chen, Z., Wang, Z., and Ouyang, W. (2020). Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 140–149, Seattle, WA, USA. IEEE.

[75] Loh, S. B., Roy, D., and Fernando, B. (2022). Long-term action forecasting using multi-headed attention-based variational recurrent neural networks. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2418–2426.

[76] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[77] Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. *In ICLR*.

[78] Malawski, F. and Kwolek, B. (2019). Improving multimodal action representation with joint motion history context. *Journal of Visual Communication and Image Representation*, 61:198–208.

[79] Mazzia, V., Angarano, S., Salvetti, F., Angelini, F., and Chiaberge, M. (2022). Action transformer: A self-attention model for short-time pose-based human action recognition. *Pattern Recognition*, 124:108487.

[80] McGillem, C. and Cooper, G. (1984). *Continuous and Discrete Signal and System Analysis*. HRW series in electrical and computer engineering. Holt, Rinehart, and Winston.

[81] Newell, A., Huang, Z., and Deng, J. (2017). Associative embedding: End-to-end learning for joint detection and grouping. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[82] Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 483–499, Cham. Springer International Publishing.

[83] Ng, Y. and Fernando, B. (2020). Forecasting future action sequences with attention: A new approach to weakly supervised action forecasting. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, PP.

[84] Papandreou, G., Zhu, T., Chen, L., Gidaris, S., Tompson, J., and Murphy, K. (2018). Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. *CoRR*, abs/1803.08225.

[85] Petrovich, M., Black, M., and Varol, G. (2021). Action-conditioned 3d human motion synthesis with transformer vae. In *ICCV*, pages 10965–10975.

[86] Plizzari, C., Cannici, M., and Matteucci, M. (2021). Skeleton-based action recognition via spatial and temporal transformer networks. *Computer Vision and Image Understanding*, 208-209:103219.

[87] Post, S. L., McLachlan, J., Lonas, T., Dancs, J., Knobloch, D., Darrow, C., Sinn, E., Davis, S., Neilly, D., Funk, A., Golz, J., Phelps, A., and Goers, B. (2009). Aerodynamics of a Badminton Shuttlecock. In *Volume 7: Engineering Education and Professional Development*, pages 145–150, Lake Buena Vista, Florida, USA. ASMEDC.

[88] Rahmad, N. A. and As'ari, M. A. (2020). The new convolutional neural network (cnn) local feature extractor for automated badminton action recognition on vision based data. *Journal of Physics Conference Series*, 1529.

[89] Rahmad, N. A., As'ari, M. A., Ibrahim, M. F., Sufri, N. A. J., and Rangasamy, K. (2020). Vision based automated badminton action recognition using the new local convolutional neural network extractor. In Hassan, M. H. A., Che Muhamed, A. M., Mohd Ali, N. F., Lian, D. K. C., Yee, K. L., Safii, N. S., Yusof, S. M., and Fauzi, N. F. M., editors, *Enhancing Health and Sports Performance by Design*, pages 290–298, Singapore. Springer Singapore.

[90] Rahmad, N. A. and As'ari, M. A. (2020). The new convolutional neural network (cnn) local feature extractor for automated badminton action recognition on vision based data. *Journal of Physics: Conference Series*, 1529(2):022021.

[91] Reddy, D., Dwivedi, D., Yemula, P., and Pal, M. (2023). Data-driven approach to form energy-resilient microgrids with identification of vulnerable nodes in

active electrical distribution network. *International Journal of Data Science and Analytics*, pages 1–12.

[92] Ren, J., Orwell, J., Jones, G. A., and Xu, M. (2009). Tracking the soccer ball using multiple fixed cameras. *Computer Vision and Image Understanding*, 113(5):633–642. Computer Vision Based Analysis in Sport Environments.

[93] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 91–99, Cambridge, MA, USA. MIT Press.

[94] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

[95] Sainan, K., Mohamad, M., Mohamed, Z., Saari, S., Mat, M. F., and Khusaini, N. (2018). Athletes tracking using homography method: a preliminary study. *International Journal of Engineering and Technology(UAE)*, 7:6–10.

[96] Shen, L., Liu, Q., Li, L., and Yue, H. (2016). 3D reconstruction of ball trajectory from a single camera in the ball game. In Chung, P., Soltoggio, A., Dawson, C. W., Meng, Q., and Pain, M., editors, *Proceedings of the 10th International Symposium on Computer Science in Sports (ISCSS)*, volume 392, pages 33–39. Springer International Publishing, Cham. Series Title: Advances in Intelligent Systems and Computing.

[97] Skublewska-Paszkowska, M., Powroźnik, P., and Łukasik, E. (2020). Learning three dimensional tennis shots using graph convolutional networks. *Sensors*, 20:6094.

[98] Soroush Mehraban, Vida Adeli, B. T. (2024). Motionagformer: Enhancing 3d human pose estimation with a transformer-gcnformer network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*.

[99] Stergiou, A. and Damen, D. (2023). The wisdom of crowds: Temporal progressive attention for early action prediction. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14709–14719.

[100] Sun, K., Xiao, B., Liu, D., and Wang, J. (2019a). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5693–5703.

[101] Sun, K., Xiao, B., Liu, D., and Wang, J. (2019b). Deep High-Resolution Representation Learning for Human Pose Estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5696, Long Beach, CA, USA. IEEE.

[102] Sun, N.-E., Lin, Y.-C., Chuang, S.-P., Hsu, T.-H., Yu, D.-R., Chung, H.-Y., and İk, T.-U. (2020). Tracknetv2: Efficient shuttlecock tracking network. In *2020 International Conference on Pervasive Artificial Intelligence (ICPAI)*, pages 86–91.

[103] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

[104] Tang J., W. J. . H. J. (2023). Predicting human poses via recurrent attention network. *Visual Intelligence*, 1.

[105] Tarashima, S., Haq, M. A., Wang, Y., and Tagawa, N. (2023). Widely applicable strong baseline for sports ball detection and tracking. In *BMVC*.

[106] Tian, L., Wang, P., Liang, G., and Shen, C. (2021). An adversarial human pose estimation network injected with graph structure. *Pattern Recognition*, 115:107863.

[107] Van Zandycke, G. and De Vleeschouwer, C. (2019). Real-time cnn-based segmentation architecture for ball detection in a single view setup. In *Proceedings Proceedings of the 2nd International Workshop on Multimedia Content Analysis in Sports*. ACM.

[108] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017a). Attention is all you need. *CoRR*, abs/1706.03762.

[109] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017b). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

[110] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

[111] Wang, F., Sun, L., Yang, B., and Yang, S. (2006). Fast Arc Detection Algorithm for Play Field Registration in Soccer Video Mining. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 6, pages 4932–4936. IEEE. ISSN: 1062-922X.

[112] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. (2016). Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *European conference on computer vision*, pages 20–36. Springer.

[113] Wang, P. and Li, S. (2018). Structural-attentioned lstm for action recognition based on skeleton. In *Other Conferences*.

[114] Wang, W., Du, W., and Peng, W. (2023a). Shuttleset22: Benchmarking stroke forecasting with stroke-level badminton dataset. *CoRR*, abs/2306.15664.

[115] Wang, W., Huang, Y., Ik, T., and Peng, W. (2023b). Shuttleset: A human-annotated stroke-level singles dataset for badminton tactical analysis. In *KDD*, pages 5126–5136. ACM.

[116] Wang, W., Shuai, H., Chang, K., and Peng, W. (2022a). Shuttlenet: Position-aware fusion of rally progress and player styles for stroke forecasting in badminton. In *AAAI*, pages 4219–4227. AAAI Press.

[117] Wang, W., Yao, L., Chen, L., Lin, B., Cai, D., He, X., and Liu, W. (2022b). Crossformer: A versatile vision transformer hinging on cross-scale attention. In *International Conference on Learning Representations, ICLR*.

[118] Wang, X., Hu, J.-F., Lai, J.-H., Zhang, J., and Zheng, W.-S. (2019). Progressive teacher-student learning for early action prediction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3551–3560.

[119] Watanabe, T., Haseyama, M., and Kitajima, H. (2004). A soccer field tracking method with wire frame model from TV images. In *2004 International Conference on Image Processing, 2004. ICIP '04.*, volume 3, pages 1633–1636 Vol. 3. IEEE. ISSN: 1522-4880.

[120] West, D. (2001). *Introduction to Graph Theory*. Featured Titles for Graph Theory. Prentice Hall.

[121] Xiao, B., Wu, H., and Wei, Y. (2018). Simple baselines for human pose estimation and tracking. In *European Conference on Computer Vision (ECCV)*.

[122] Xiao, Q., Zaidi, Z., and Gombolay, M. (2024). Multi-Camera Asynchronous Ball Localization and Trajectory Prediction with Factor Graphs and Human Poses. arXiv:2401.17185 [cs].

[123] Xu, Y., Zhang, J., Zhang, Q., and Tao, D. (2022). ViTPose: Simple vision transformer baselines for human pose estimation. In *Advances in Neural Information Processing Systems*.

[124] Xu, Y., Zhang, Q., Zhang, J., and Tao, D. (2021). Vitae: Vision transformer advanced by exploring intrinsic inductive bias. *CoRR*, abs/2106.03348.

[125] Yan, F. (2005). Tennis ball tracking for automatic annotation of broadcast tennis video. *Proceedings of the British Machine Vision Conference*.

[126] Yan, H., Liu, Y., Wei, Y., Li, G., and Lin, L. (2023). Skeletonmae: Graph-based masked autoencoder for skeleton sequence pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[127] Yan, S., Xiong, Y., and Lin, D. (2018a). Spatial temporal graph convolutional networks for skeleton-based action recognition. *AAAI*, 32.

[128] Yan, S., Xiong, Y., and Lin, D. (2018b). Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

[129] Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., and Yan, S. (2022). Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10819–10829.

[130] Zhu, K., Wong, A., and McPhee, J. (2022). Fencenet: Fine-grained footwork recognition in fencing. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3588–3597.

[131] Zhu, W., Ma, X., Liu, Z., Liu, L., Wu, W., and Wang, Y. (2023). Motionbert: A unified perspective on learning human motion representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Paper 1: TemPose: A new approach skeleton-based action recognition

TemPose: a new skeleton-based transformer model designed for fine-grained motion recognition in badminton

Magnus Ibh Stella Grasshof Dan Witzner Pascal Madeleine
Machine learning group, IT University of Copenhagen Aalborg Univeristy
{ibhq, stgr, witzner}@itu.dk pm@hst.aau.dk

Abstract

This paper presents TemPose, a novel skeleton-based transformer model designed for fine-grained motion recognition to improve understanding of the detailed player actions in badminton. The model utilizes multiple temporal and interaction layers to capture variable-length multi-person human actions while minimizing reliance on non-human visual context. TemPose is evaluated on two fine-grained badminton datasets, where it significantly outperforms other baseline models by incorporating additional input streams, such as the shuttlecock position, into the temporal transformer layers of the model. Additionally, TemPose demonstrates great versatility by achieving competitive results compared to other state-of-the-art skeleton-based models on the large-scale action recognition benchmark NTU RGB+D. Experiments are conducted to explore how different model parameter configurations affect TemPose's performance. Additionally, a qualitative analysis of the temporal attention maps suggests that the model learns to prioritize frames of specific poses relevant to different actions while formulating an intuition of each individual's importance in the sequences. Overall, TemPose is an intuitive and versatile architecture that has the potential to be further developed and incorporated into other methods for managing human motion in sports with state-of-the-art results.

1. Introduction

Badminton is a fast-paced racket sport that requires a high level of skill, athleticism, and tactical awareness. As the sport's popularity grows, the need for objective and data-driven methods for evaluating player performance has become increasingly important. One area of particular interest uses automatic analysis, specifically human action recognition (HAR), to provide insights into a player's performance [13] and inform decision-making in the sport. Fine-grained action recognition deals with action classes closely related in both type (e.g. badminton strokes) and motion (i.e., strokes may look similar) and is appropriate for highly technical sports disciplines that require high precision and accuracy in movement execution. The required attention to detail in badminton results in small and subtle differences in how players execute specific movements, which are difficult to capture using RGB-based methods [42]. Skeleton motion as a primary feature in fine-grained action recognition has been effective in various sports disciplines [8, 17], including badminton [19, 21]. Skeleton-data provides a detailed representation of the body movement through spatiotemporal sequences of joint and bone positions, which enables extracting features crucial for recognizing specific actions and movements, even those that may be subtle or difficult to detect with traditional imaging techniques. While existing methods for skeleton-based action recognition have achieved good results on controlled action benchmark datasets [23, 39], many tend to lack robustness and scalability for real-world applications. In an approach to address this issue, recent research has explored the use of transformer models, which have shown excellent capabilities in natural language processing (NLP) [7] and image segmentation [9, 14], to model sequential data for video action recognition [1, 18, 22].

This paper presents *TemPose*, a new skeleton-based transformer model designed for fine-grained motion recognition in badminton. The model offers several significant contributions, including a novel factorized transformer model that combines temporal and interaction layers, multi-person interaction modeling, and improved recognition rates using fewer parameters. The proposed action recognition model, *TemPose*, is outlined in Figure 1. The model takes processed skeleton data as input and passes it through a sequence of transformer layers in the TemPose encoder. This process creates tokens of the temporal data and captures the temporal body dynamics and sequential interactions between an arbitrary number of people involved in the action. The MLP head at the top predicts the action based on the information embedded in a class token. The primary and badminton-specific version of the model includes the fu-

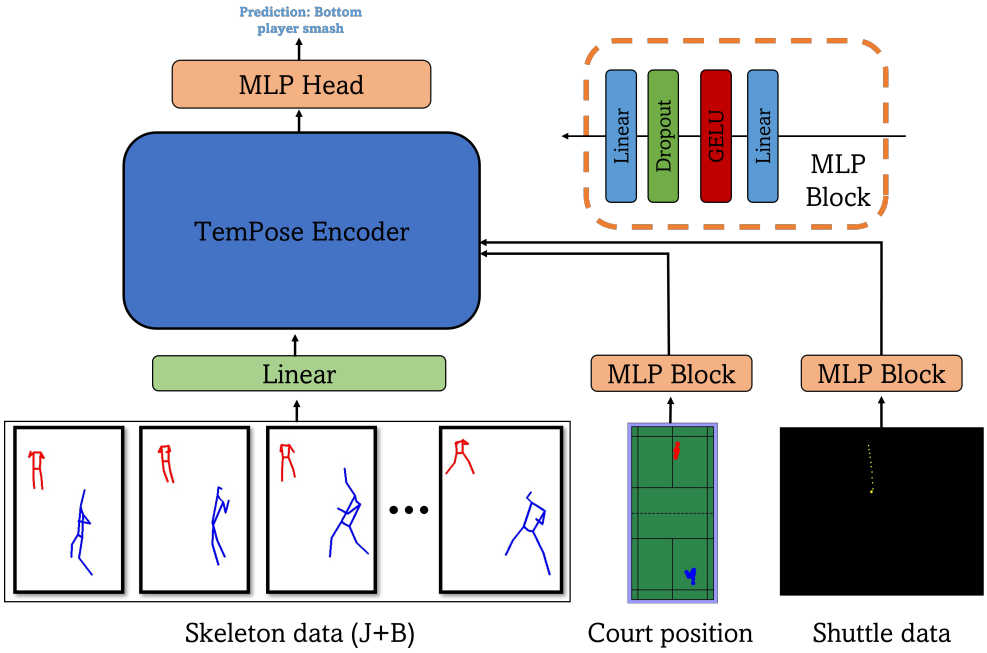


Figure 1. Illustration of our proposed action recognition framework, TemPose. The framework uses human skeleton data, consisting of joint and bone information, and incorporates additional features such as player court position and shuttlecock position from a badminton action (e.g., smash). The TemPose encoder, composed of multiple transformer layers, processes the input to embed relevant features into a class token. Finally, an MLP head utilizes these features to predict the action class. The composition of the MLP block is shown in the upper right corner.

sion of skeleton-data with player court positions (CP) and shuttlecock position (SP). We exhaustively test two different versions of TemPose, where the additional modalities are integrated at different stages of the TemPose encoder. In one version (TemPose-NF), the CP and SP sequences are tokenized and appended to the embedded skeleton-data before the interaction transformer layers. Figure 2 depicts TemPose-NF. The other version (TemPose-TF) prioritizes an early fusion of the skeleton, SP, and CP modalities.

An overview of related work is provided in Section 2, followed by a description of pose and shuttle estimation, preprocessing, and the model architecture in Section 3. The experimental results on fine-grained badminton datasets are presented in Section 4, along with testing on a standard benchmark action recognition datasets NTU RGB+D [20, 32]. The paper concludes with a qualitative analysis of the information stored in the different transformer layers and future works in 5.

2. Related Work

Action recognition in sports Most work on action recognition in badminton uses convolutional neural network (CNN) architectures for feature extraction on RGB images [29–31]. Decision-making algorithms like Support Vector Machines then use the extracted features to make predictions. Other approaches involve using handcrafted features such as Histogram of Oriented Gradients, along with temporal convolutional networks (TCN) to process the action's spatial and temporal aspects [5, 13]. Instead of using image data, skeleton data has been successfully used for the analysis and recognition tasks of other sports, such as Tai Chi [8, 10, 36] and fencing [26, 42]. But despite its potential, skeleton poses have yet to be thoroughly tested for badminton tasks. In one recent example [21], skeleton data is used in a gated recurrent unit (GRU) model to perform binary hit detection. However, like other recurrent models, GRUs can struggle with training issues. This paper proposes an architecture more suited for utilizing skeleton data for badminton recognition tasks.

Human-action recognition using skeleton data. Graph convolutional networks (GCN) are a popular method for skeleton-based action recognition [35, 40]. GCNs uses nodes to represent every human joint at every time. Connecting nodes, both spatially and temporally, to the other nodes with edges allows GCNs to capture both spatial and temporal aspects of human motion. Spatio-temporal GCNs have demonstrated promising results for skeleton-based ac-

tion recognition [23, 39, 40], but they also possess some limitations. One limitation is their limited ability to model long-term dependencies in complex actions, as they typically use a fixed-length temporal window. Moreover, they are sensitive to missing data [41] and require a carefully designed graph structure based on the recognized actions' characteristics, which can be challenging. CNNs are also commonly used for analyzing skeleton data. One approach is to stack heatmaps along the temporal dimension into a 3D input and use 3D-CNNs to extract information [3, 11]. Other studies, such as [17, 42], generate a temporal sequence of joint coordinates and use TCNs to encode the information.

Transformers for human action recognition. The emergence of ViT [9] has led to many applications of vision transformer backbones [2, 22, 37, 38]. Including vision transformers [1, 12, 18, 22] used for HAR. These works are typically trained on Kinetics-400 [16] to mitigate the issue involved with over-fitting. But only a few works have considered transformers on other modalities than RGB data, in our case, skeleton data. Utilizing transformer based models on skeleton data has, however, been attempted in other recent work. [28] combines self-attention with a GCN and TCN to model spatial and temporal attention. Similarly, [27] performs temporal encoding of the skeleton poses with a sequence of temporal transformer layers.
Unlike previous work, we present a factorized transformer encoder. Embedding individual skeleton data into temporal and interaction tokens allows the method to encode information about the motion of multiple people across the entire sequence length.

3. Model

This section first outlines the extraction process for the skeleton, CP, and SP data. We introduce the temporal transformer layer module for a single individual and subsequent action prediction. Subsequently, the model is extended to a factorized temporal and interaction encoder, which embeds information about the interaction between individuals in the class token. Last, we describe two methods of incorporating CP and SP data into the model.

3.1. Extraction of the skeleton, player position, and shuttlecock data

A visual representation of the skeleton-data retrieval is shown in Figure 3. In a video sequence with T frames, the poses of a person are given by the sequence $P = [P_1, \ldots, P_T]^{\mathrm{T}} \in \mathbb{R}^{T\times 2J}$. A pose $P_t \in \mathbb{R}^{J\times 2}$ in frame t is represented by J keypoints $(x_i^{(t)}, y_i^{(t)})$, where $x_i^{(t)}$ and $y_i^{(t)}$ are 2D joint coordinates for the joint i. The bones $B_t \in \mathbb{R}^{B\times 2}$ in frame t are represented by the keypoint differences $(x_i^{(t)} - x_j^{(t)}, y_i^{(t)} - y_j^{(t)})$, where i and j are specific joint pairs that make up the human bones. The final skeleton data sequence, S, of an individual, is defined to be

$$S = [[P_1, B_1], \ldots, [P_T, B_T]]^{\mathrm{T}} \in \mathbb{R}^{T\times 2(J+B)} \quad (1)$$

The pose extraction pipeline consists of two stages using tools from previous studies, including [4, 6] to detect humans and perform pose estimation. We employ HRnet [33], a pre-trained framework, to estimate the 2D poses. However, irrelevant individuals, such as spectators in the crowd, can limit the quality of the skeleton data. To address this issue in badminton, we calculate a homography using the court's known dimensions and map the feet of the detected individuals to the ground plane. By doing so, we only consider skeletons within the court and can identify each sequence's top and bottom player. In cases where a whole skeleton is missing, we replace it with the pose from the previous frame. Finally, we normalize the poses by centering them and scaling them to have a bounding box diagonal of 1. Additionally, we sample the players' 2D ground plane feet position (i.e., CP) for each time frame as an additional input feature. The sequence PC is represented as $\in \mathbb{R}^{T\times 2}$. The shuttlecock's position holds valuable information for categorizing the different strokes in badminton. To extract the shuttlecock's position, we use a pre-trained model from [34] to obtain its image coordinates in each frame of the video, represented as (u, v, c), where u and v are the image coordinates of the shuttlecock, and c is the confidence of the prediction. We only consider predictions with a confidence score above 0.75; we pad failed predictions with zeros.

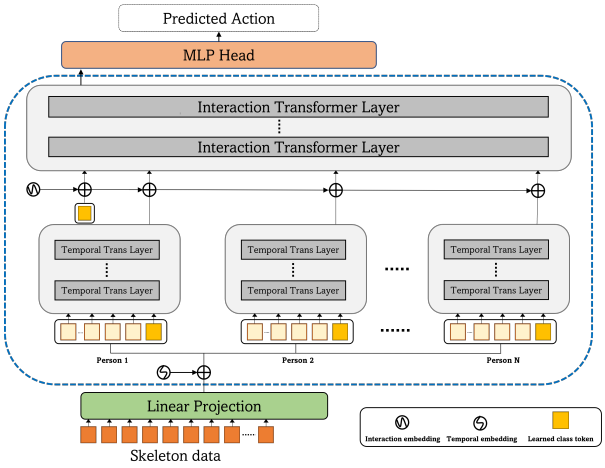


Figure 2. Illustration of *TemPose encoder* shows the factorized transformer structure. First, the temporal token for each person is encoded by the temporal transformer layer. Second, The interaction between actors is modeled based on the temporal context of each person.

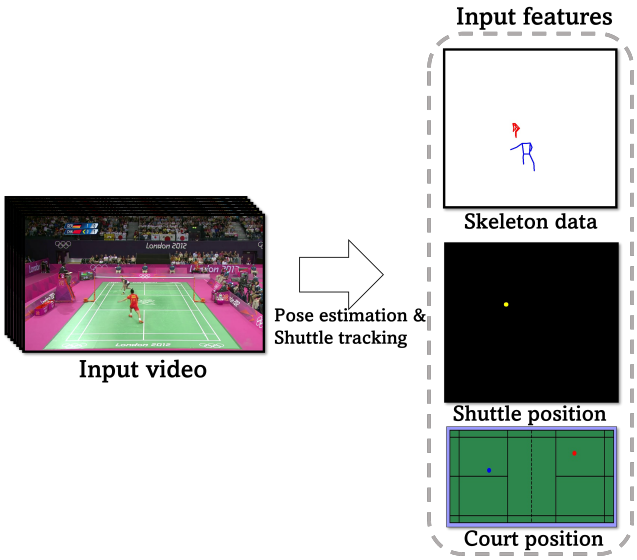


Figure 3. The figure illustrates the input data utilized by our proposed action recognition framework, *TemPose*. The framework takes in centered and normalized skeleton data of the badminton players, along with their court position and the scaled position of the shuttlecock, all of which are extracted from RGB video input. Specifically, HRNet [33] estimates the poses of the badminton players, while TrackNet [34] estimates the position of the shuttlecock.

3.2. Skeleton-based temporal self-attention for action prediction

As a first step, we consider a single-person sequence without the interaction layers from Figure 2. The skeleton data is mapped through a linear projection to a sequence of temporal tokens $[x_1, \ldots, x_T]^{\mathrm{T}} \in \mathbb{R}^{T \times D_L}$, where each token $x_t \in \mathbb{R}^{D_L}$ is the vector representation of the skeleton at that particular time frame. A learnable temporal embedding $E_T \in \mathbb{R}^{T+1 \times D_L}$ is added to the tokens to capture the underlying temporal structure better. The sum of the embedding and projection yields x, the input of the transformer layers. x is formally defined as

$$x = [x_{cls}, \mathrm{Linear}(S)]^{\mathrm{T}} + E_T \tag{2}$$

$$= [x_{cls}, x_1, \ldots, x_T]^{\mathrm{T}} + E_T, \in \mathbb{R}^{T \times D_L} \tag{3}$$

where $x_{cls} \in \mathbb{R}^{D_L}$ is a learned class token, D_L is the dimension of the embedded feature space, and Linear is a learned linear projection. The representation of x_{cls} at the final transformer layer is used by the MLP head to make predictions. The tokens defined in (3) are then passed through transformer layers, where L is the transformer depth. To distinguish between the tokens at different layers, we define them as $x^{(l)}$ after having passed through layer l. Each layer is composed of a multi-head self-attention (MHSA), layer normalization (LN), and a multi-layer perceptron (MLP), which consists of two linear projections only separated by a GELU activation [15] and dropout, see Figure 1. The design of a single transformer layer is illustrated in Figure 4 on the left. The transformer block is described by Equation 4 and Equation 5

$$\tilde{x}^{(l+1)} = x^{(l)} + \mathrm{MHSA}(\mathrm{LN}(x^{(l)})), \tag{4}$$

$$x^{(l+1)} = \tilde{x}^{(l+1)} + \mathrm{MLP}(\mathrm{LN}(\tilde{x}^{(l+1)})), \tag{5}$$

where $\tilde{x}^{(l+1)}$ is the in-between embedded obtained after the self-attention module. The following equation describes a single head of self-attention

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{D_K}}\right) V, \tag{6}$$

where $Q = W_Q x^{(l)} \in \mathbb{R}^{T \times D_A}$, $K = W_K x^{(l)} \in \mathbb{R}^{(T+1) \times D_A}$, and $V = W_V x^{(l)} \in \mathbb{R}^{(T+1) \times D_A}$ are learned linear projections of the input sequence that respectively represent the current token, the other tokens, and their associated values, used to calculate attention scores and output. After scaling and softmax activation of the input variables, (6) serves as an attention map that provides temporal context to the value-array V, where D_A is the attention head latent dimension. The output of the MHSA yields N_{heads} value vectors V weighted by the temporal attention maps. As illustrated in Figure 4, these weighted value vectors are concatenated and mapped to the updated representation of the temporal tokens $\tilde{x}^{(l+1)}$ with another learned linear projection. Note that the number of transformer layers L is an adjustable hyperparameter.
Finally, the class token x_{cls}^{L} is fed to an MLP block to predict the action category of the samples

$$x_{act} = \mathrm{MLP}(x_{cls}), \tag{7}$$

where $x_{act} \in \mathbb{R}^{D_{cls}}$ is the model prediction, and D_{cls} the number of different action categories.

3.3. Factorized temporal and interaction structure

Actions in video sequences often contain multiple people (e.g., two people in badminton singles matches) making different subactions and movements in parallel and often reacting to the other people involved in the action. Hence, in the TemPose encoder, we want to account for multiple people and their actions. *TemPose* utilizes a factorized encoder structure of transformer layers, inspired by ViViT [1], to model the interaction between people. Figure 2 depicts the TemPose encoder. The model consists of 1.) a temporal transformer layer that captures the temporal interaction between poses extracted from the same person, identical to the temporal structure outlined in section 3.2. However, the temporal encoding is now performed in parallel

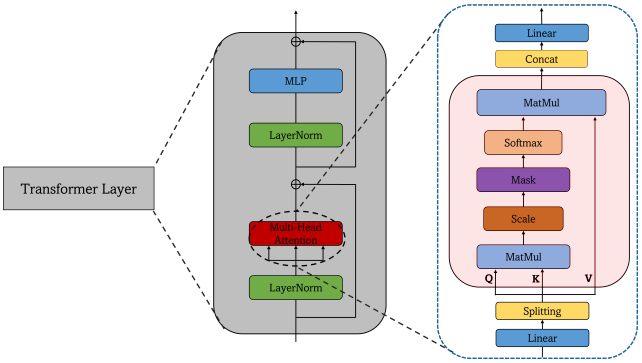


Figure 4. Illustration of transformer layers to the left. The structure of the transformer layers is identical for the temporal and interaction transformer layers. The right block shows the composition of a single-head self-attention module. The Mask leaves out the (zero) padded temporal and interaction tokens in the attention map, which allows the model to handle action sequences of varying lengths proficiently.

for up to N people involved in the action sequence, see Figure 2. As a result, the class token notation $x_{cls}^{(l)}$ is updated to $\tau_{cls,n}^{(l_T)}$. $\tau_{cls,n}^{(l_T)}$ is the temporal class token for person $n \in \{1, \ldots, N\}$, at temporal layer l_T 2.) After being processed by all L_T temporal layers, the N temporal class tokens are concatenated to $[\tau_{cls,1}^{(L_T)}, \ldots, \tau_{cls,N}^{(L_T)}]$, prepended with a interaction class token η_{cls}, and assigned an interaction embedding E_I identical to the step of Equation 3. Summing up the input for the interaction encoder becomes

$$z = [\eta_{cls}, \tau_{cls,1}, \ldots, \tau_{cls,N}]^{\mathrm{T}} + E_I, \in \mathbb{R}^{T \times D_L}. \quad (8)$$

Subsequently, the input tokens, z, are passed through L_N transformer layers to capture interactions between the embedded temporal class tokens of people participating in the action. An MLP head uses the final representation of the interaction classification token $\eta_{cls}^{(L_N)} \in \mathbb{R}^{D_L}$, see Equation 7, to predict the action class.

3.4. Player and shuttlecock position infusion

This section discusses two configurations for integrating badminton-specific CP and SP data into the encoder.

Temporal fusion: In the first fusion configuration, the SP and CP input data passes through separate TCN blocks consisting of two 1D-convolutional layers separated only by a GELU activation and dropout. The two layers have dilation 1 and 3, respectively, with a kernel size of 5 and stride 1. Through the TCN block, the channels (i.e., dimensionality) of SP and CP data are increased to the embedded dimension of the transformer layers D_L. The two new input streams are then appended to the transformer input equivalent to adding additional people (see Figure 2). The remaining architecture is identical to factorized encoder described above.

Interaction fusion In a different approach, the CP and SP data are first incorporated into the *TemPose* encoder after the temporal transformer layers. Here the SP and CP data are flattened along the temporal and coordinate dimensions and then separately passed through an MLP block. The resulting representations are appended to z in Equation 8, which is then processed identically to the original *TemPose* architecture but with two additional interaction tokens. In the experiments section, *TemPose* without fusion is referred to as *TemPose-V*. In contrast, *TemPose-TF* and *TemPose-NF* refer to temporal fusion and interaction fusion, respectively.

Temporal & multi-person padding A property of transformer architectures is the ability to handle sequences of different lengths. We implement this for TemPose by always creating a set number of temporal tokens T, corresponding to the maximal clip length for a video. Shorter videos are padded with zeros and assigned pad tokens, so they are not considered when calculating self-attention. Specifically, the MASK step in the MHSA reduces attention scores on the padded tokens to zero, see Figure 4. TemPose extends this process by choosing an upper limit N of people to model interactions from. Clips with fewer people are padded with zeros and assigned pad tokens, so they are not considered in the interaction attention.

4. Experiments

This section presents the results of our experiments to assess the performance of the factorized transformer layers compared to other state-of-the-art skeleton-based human activity recognition models. Specifically, we evaluated *TemPose* on two fine-grained badminton datasets [13] and demonstrated the versatility of the architecture by including experiments on the large-scale human motion dataset NTU RGB+D [20, 32].

Badminton Olympics (Bad OL). A fine-grained badminton dataset from 10 videos containing 15300 samples from 13 classes of badminton strokes with the following classes: top/bottom player forehand, backhand, smash, lob, react strokes, and a none class. The train/test split is match/video splitting, where all clips from one video (match) are kept as the test set.

Dataset on Badminton Stroke Placement (Bad PL). The dataset is confidential but contains 5500 samples of backcourt badminton strokes categorized into either attack or transport strokes. Additionally, the dataset includes information on the approximate location of the shuttlecock

Table 1. Hyperparameters for the TemPose training procedure. The right part of the table includes regularization and data augmentation choices.

Training	
Batch size	64
Optimizer	AdamW
Warm-up	25%
Learning rate	1e-04
LR scheduler	cosine decay

Regularization	
Label smoothing	0.1
Flipping	30%
Random shifting	30%
Dropout	0.3
Weight decay	0.01

placement for each stroke grouped into three different areas, such as left backcourt or middle midcourt, resulting in 12 different classes based on stroke type and placement. The train/test splitting is done cross-matches.

NTURGB+D. Is a large-scale human action dataset collected in a controlled setting. The dataset contains two versions, NTU-60 and NTU-120. NTU-60 has 57000 videos categorized into 60 different actions. NTU-120 has 114000 videos belonging to 120 different categories. Test and training data can be split in three different ways: cross-setup (**XSet**), cross-view (**XView**), and cross-subject (**X-sub**).

4.1. Implementation details

Table 1 list all settings and hyperparameters used for the training procedure. The choice is made based on a randomized search across the datasets. The AdamW optimization algorithm [24] is used for all training runs along with cosine-annealing [25]. Each training run is initialized with a sequence of warm-up epochs, slowly increasing the learning rate linearly from 0 to prevent overfitting. Unless specified otherwise, joint and bone data, J and B, respectively, are used together as skeleton data input.

4.2. Component studies

We analyze the individual components and different model configurations of TemPose. Unless stated otherwise, the performance is reported as classification accuracy on the Bad OL dataset. The default configuration uses the depth $L_T = L_N = 2$, $N_{heads} = 6$, embedded dimensions of $D_L = 100$ and $D_A = 128$, and lastly, an MLP scale factor of 4 between the input and hidden layers in MLP blocks.

Model configurations. To validate the multi-modal fusion approaches of the CP and SP data, we examine the performance of TemPose-V, TemPose-TF, and TemPose-NF for many different model settings shown in Table 2. AcT [27], a purely temporal skeleton-based model, is shown as the baseline model. Among the TemPose versions, TemPose-TF with $D_L = 100$ and $D_A = 128$ has the highest accuracy of 90.7% while only having 1.7 million parameters. The results suggest that temporal fusion of SP and CP is the best approach, as it achieves the highest accuracy using the fewest parameters.

Table 2. Accuracy and model size for different settings of the 3 TemPose versions. The number of attention heads $N_{heads} = 6$ and depth $L_T = L_N = 2$ are shared for all model configurations.

Model configuration	Params	Acc
Baseline (AcT [27])	2.1M	83.7%
TemPose-V		
with ($D_L = 75$, $D_A = 100$)	0.9M	85.6%
with ($D_L = 200$, $D_A = 200$)	5.2M	83.6%
TemPose-TF		
with ($D_L = 50$, $D_A = 75$)	0.5M	88.6%
with ($D_L = 100$, $D_A = 128$)	1.7M	90.7%
with ($D_L = 200$, $D_A = 256$)	6.7M	88.0%
TemPose-NF		
with ($D_L = 50$, $D_A = 75$)	2.5M	88.1%
with ($D_L = 100$, $D_A = 128$)	3.8M	89.3%
with ($D_L = 200$, $D_A = 256$)	9.0M	86.2%

Exploring joint-bone skeleton data. We investigated the impact of incorporating bone data into the joint data of the skeleton on the Bad OL and NTU RGB+D datasets for TemPose without CP and SP input. The results are presented in Table 3 and Table 6. Our findings are consistent with previous studies [11, 23]. The performance of TemPose significantly improves by utilizing both bone and joint data as input.

Importance of transformer depth. The effect of varying transformer depth is a crucial aspect of transformer models. Table 4 shows the results of a depth study on the TemPose-TF model. The model settings are kept constant throughout the study, except for the number of transformer layers, and report the model's performance for different combinations of L_T and L_N. The results show that increasing the transformer depth beyond a certain point leads to a drop in performance. The best accuracy is achieved for the combination of $L_T = 2$ and $L_N = 2$, with a top-1 accuracy of 90.7%. Increasing the depth further to $L_T = 3$ and $L_N = 3$ leads to a significant drop in accuracy to 88.3%. The performance continues to degrade as the depth is further increased. The continued drop could suggest that the performance continually worsens due to overfitting as the depth is increased. Thus, the study exemplifies the importance of finding the right balance between model complexity and data size. The overfitting can possibly be attributed to issues such as vanishing gradients or the relatively limited number of training samples in the Bad OL dataset.

Table 3. Joint + Bone architecture study

Models	Acc
Baseline (AcT [27])	
with (J)	81.8%
with (J+B)	83.7%
TemPose-V	
with (J)	81.4%
with (J+B)	85.6%

Table 4. Transformer depth study of the TemPose-TF. The remaining model settings are constant for the study, where $D_L = 100$, $D_A = 128$, and $N_{heads} = 6$. The performance of TemPose drops when the transformer depth increases.

Model	Acc
TemPose-TF	
with ($L_T = 1,\ L_N = 1$)	89.7%
with ($L_T = 1,\ L_N = 2$)	89.9%
with ($L_T = 2,\ L_N = 1$)	90.0%
with ($L_T = 2,\ L_N = 2$)	90.7%
with ($L_T = 3,\ L_N = 3$)	88.3%
with ($L_T = 4,\ L_N = 4$)	86.6%
with ($L_T = 6,\ L_N = 2$)	85.5%
with ($L_T = 2,\ L_N = 6$)	86.1%
with ($L_T = 6,\ L_N = 6$)	85.4%
with ($L_T = 8,\ L_N = 8$)	85.2%

Table 5. Top-1 accuracy results for *TemPose* with temporal (TF) and interaction (NF) fusion, to state-of-the-art (HAR) models on Badminton placement (Bad PL) and Badminton Olympics (Bad OL).

Model	Bad PL	Bad OL	Params
Bidirectional TCN [42]	80.4%	86.1%	4.1M
TCN Hog [13]	66.6%	77.0%	1.1M
ST-GCN [40]	72.3%	82.0%	3.4M
AcT-M [27]	77.9%	83.7%	2.1M
MS-G3D [23]	78.0%	83.2%	3.2M
TemPose-TF	83.9%	90.7%	2.2M
TemPose-NF	84.3%	89.3%	3.8M

4.3. Evaluation

Fine-grained sports action recognition - badminton. Table 5 shows the Top-1 accuracy results for TemPose-TF and TemPose-NF on two different Badminton datasets - Bad PL and Bad OL. The table also includes results for other state-of-the-art models on the same datasets. Overall, the results show that TemPose outperforms all other models on both datasets, with TemPose-TF achieving the highest accuracy on Bad OL and TemPose-NF achieving the highest accuracy on Bad PL. As observed in the configuration study, both fusion approaches achieve strong results, and based on our studies, no method is superior by a significant margin. However, we conclude that TemPose can accurately be used to classify the different types of movements in badminton.

Large-scale human action recognition. We showcase the versatility of TemPose, by testing TemPose on other more generic large-scale HAR benchmarks and comparing TemPose-V to other top-performing skeleton-based actions recognition models. Table 6 shows the results of *TemPose-V* on the NTU datasets. Despite being slightly worse than MS-G3D [23], and PoseConv3D [11] overall, TemPose achieves competitive results to other state-of-the-art models on all splittings of NTU RGB+D.

4.4. Qualitative analysis of temporal and interaction attention

We examine the attention maps of the transformer layers. To inspect what information is captured by the encoder. The temporal attention maps of two forehand strokes shown in Figure 6 reveal that similar patterns emerge between actions of the same class. The similar attention maps suggest that the model has learned to focus on specific temporal aspects of the actions to predict the entire sequence.
The attention maps are used to determine a temporal and interaction attention score for all actions. We define the attention score as the self-attention of the x_{cls}-token in the last transformer layer, aggregated and normalized across all attention heads. The temporal attention score is averaged over all individuals but weighted according to their interaction attention score. For a badminton smash, the attention score is depicted in Figure 5. TemPose identifies the frames around contact with the shuttlecock as the most significant section. The red and purple text represent the target and prediction class of the action, respectively. The model accurately predicts the action as a smash from the bottom player. Additionally, more attention is given to the smashing individual. The logical distribution of attention suggests that the model has developed the ability to gauge the relevance of each individual for the action based on their skeleton movement.

5. Future prospects

TemPose demonstrates top results on badminton action recognition tasks. However, in the experiments, the larger configurations of TemPose show clear signs of overfitting. The result indicates that the performance of TemPose may be further improved if additional steps to combat overfitting are taken. One prospect involves generating synthetic data to increase the amount of training data.

Table 6. Top-1 accuracy on the NTU RGB+D for state-of-art skeleton-based action recognition models.

	NTU RGB+D 120 (XSet)	NTU RGB+D 120 (XSub)	NTU RGB+D 60 (XSub)	NTU RGB+D 60 (XView)
ST-GCN [39]	73.2%	70.7%	81.5%	88.3%
ST-TR-agcn [28]	87.1%	85.1%	90.3%	96.3%
PoseConv3D [11]	89.6%	86.9%	93.7%	96.6%
MS-G3D [23]	88.4%	86.9%	91.5%	96.2%
TemPose-V (B)	85.1%	84.1%	91.0%	93.1%
TemPose-V (B+J)	88.5%	87.0%	92.7%	95.2%

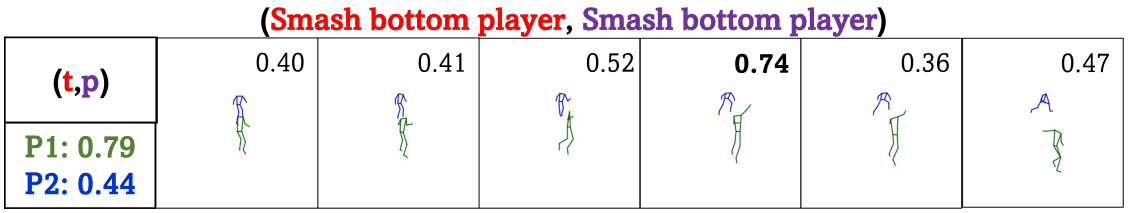


Figure 5. Prediction and attention score produced by TemPose-V for a skeleton sequence from Badminton Olympics. (t,p) refers to t as the target and p as the prediction. The interaction attention score is shown at the left, with the color corresponding to the person in the action sequence. The weighted temporal attention score is shown atop each frame in the sequence. For visual clarity, the frames are grouped by three, showing only the middle one, and the listed attention score is the average between them.

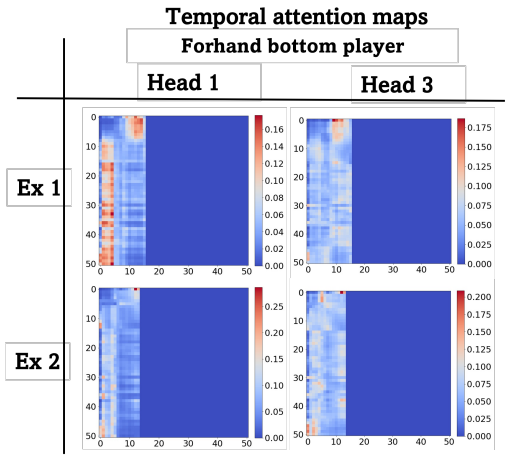


Figure 6. Temporal attention maps for a forehand by the bottom player (from Bad OL). The distribution of attention shows that TemPose prioritizes similar information when the actions are of the same class. Additionally, the attention maps also show the effect of the padding mask. The padding tokens are given no attention.

6. Conclusion

TemPose is a new skeleton-based action recognition model that uses temporal transformer layers to capture human motion dynamics and factorized interaction transformer layers to model the interaction between humans. The model outperforms existing methods in recognizing fine-grained badminton actions by fusing shuttlecock data, player court positions, and skeleton movements. It also achieves state-of-the-art performance on a large-scale action recognition dataset. Further studies will reveal how well the general TemPose architecture applies to other action recognition tasks.

Acknowledgements. We acknowledge the financial support from the Novo Nordisk Fonden as a part of Team-SPORTek, which enabled us to conduct this research. We would also like to thank Badminton Danmark and Team Danmark for their contributions to this study, including data collection and expert insights into badminton.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6816–6826, 2021. 1, 3, 4

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 813–824. PMLR, 2021. 3

[3] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017. 3

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 3

[5] Wei-Ta Chu and Samuel Situmeang. Badminton video analysis based on spatiotemporal and stroke features. In *Other Conferences*, pages 448–451, 06 2017. 2

[6] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020. 3

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. 1

[8] Lingxiao Dong, Dongmei Li, Shaobin Li, Shanzhen Lan, and Pengcheng Wang. Tai chi action recognition based on structural lstm with attention module. In *Other Conferences*, 2019. 1, 2

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1, 3

[10] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1110–1118, 2015. 2

[11] Haodong Duan, Yue Zhao, Kai Chen, Dahua Lin, and Bo Dai. Revisiting skeleton-based action recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2959–2968, 2022. 3, 6, 7, 8

[12] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021. 3

[13] Anurag Ghosh, Suriya Singh, and C. V. Jawahar. Towards structured analysis of broadcast badminton videos. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 296–304, 2018. 1, 2, 5, 7

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021. 1

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[16] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. 3

[17] Kaustubh Milind Kulkarni and Sucheth Shenoy. Table tennis stroke recognition using two-dimensional human pose estimation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4571–4579, 2021. 1, 3

[18] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *CVPR*, 2022. 1, 3

[19] Jiatong Liu and Bo Liang. An Action Recognition Technology for Badminton Players Using Deep Learning. *Mobile Information Systems*, 2022:1–10, May 2022. 1

[20] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C. Kot. Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2684–2701, 2020. 2, 5

[21] Paul Liu and Jui-Hsien Wang. MonoTrack: Shuttle Trajectory Reconstruction From Monocular Badminton Video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, page 10, 2022. 1, 2

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 3

[23] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 140–149, Seattle, WA, USA, June 2020. IEEE. 1, 3, 6, 7, 8

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. 6

[25] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. *In ICLR*, 2017. 6

[26] Filip Malawski and Bogdan Kwolek. Improving multimodal action representation with joint motion history context. *Journal of Visual Communication and Image Representation*, 61:198–208, 2019. 2

[27] Vittorio Mazzia, Simone Angarano, Francesco Salvetti, Federico Angelini, and Marcello Chiaberge. Action transformer: A self-attention model for short-time pose-based human action recognition. *Pattern Recognition*, 124:108487, 2022. 3, 6, 7

[28] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. Skeleton-based action recognition via spatial and temporal transformer networks. *Computer Vision and Image Understanding*, 208-209:103219, 2021. 3, 8

[29] Nur Azmina Rahmad and Muhammad Amir As'ari. The new convolutional neural network (cnn) local feature extractor for automated badminton action recognition on vision based data. *Journal of Physics Conference Series*, 1529, 2020. 2

[30] Nur Azmina Rahmad, Muhammad Amir As'ari, Mohamad Fauzi Ibrahim, Nur Anis Jasmin Sufri, and Keerthana Rangasamy. Vision based automated badminton action recognition using the new local convolutional neural network extractor. In Mohd Hasnun Arif Hassan, Ahmad Munir Che Muhamed, Nur Fahriza Mohd Ali, Denise Koh Choon Lian, Kok Lian Yee, Nik Shanita Safii, Sarina Md Yusof, and Nor Farah Mohamad Fauzi, editors, *Enhancing Health and Sports Performance by Design*, pages 290–298, Singapore, 2020. Springer Singapore. 2

[31] N A Rahmad and M A As'ari. The new convolutional neural network (cnn) local feature extractor for automated badminton action recognition on vision based data. *Journal of Physics: Conference Series*, 1529(2):022021, apr 2020. 2

[32] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1010–1019, 2016. 2, 5

[33] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep High-Resolution Representation Learning for Human Pose Estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5696, Long Beach, CA, USA, June 2019. IEEE. 3, 4

[34] Nien-En Sun, Yu-Ching Lin, Shao-Ping Chuang, Tzu-Han Hsu, Dung-Ru Yu, Ho-Yi Chung, and Tsì-Uí İk. Tracknetv2: Efficient shuttlecock tracking network. In *2020 International Conference on Pervasive Artificial Intelligence (ICPAI)*, pages 86–91, 2020. 3, 4

[35] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016. 2

[36] Pengcheng Wang and Shaobin Li. Structural-attentioned lstm for action recognition based on skeleton. In *Other Conferences*, 2018. 2

[37] Wenxiao Wang, Lu Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu. Crossformer: A versatile vision transformer hinging on cross-scale attention. In *International Conference on Learning Representations, ICLR*, 2022. 3

[38] Yufei Xu, Qiming Zhang, Jing Zhang, and Dacheng Tao. Vitae: Vision transformer advanced by exploring intrinsic inductive bias. *CoRR*, abs/2106.03348, 2021. 3

[39] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. 1, 3, 8

[40] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *AAAI*, 32, 2018. 2, 3, 7

[41] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 1399–1407, New York, NY, USA, 2019. Association for Computing Machinery. 3

[42] Kevin Zhu, Alexander Wong, and John McPhee. Fencenet: Fine-grained footwork recognition in fencing. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3588–3597, 2022. 1, 2, 3, 7

Paper 2: A stroke of Genius: Predicting the next move in badminton

A stroke of genius: Predicting the next move in badminton

Magnus Ibh Stella Graßhof Dan Witzner Hansen
Machine Learning Group, IT University of Copenhagen
{ibhq, stgr, witzner}@itu.dk

Abstract

This paper presents, RallyTemPose, a transformer encoder-decoder model for predicting future badminton strokes based on previous rally actions. The model uses court position, skeleton poses, and player-specific embeddings to learn stroke and player-specific latent representations in a spatiotemporal encoder module. The representations are then used to condition the subsequent strokes in a decoder module through rally-aware fusion blocks, which provide additional relevant strategic and technical considerations to make more informed predictions. RallyTemPose shows improved forecasting accuracy compared to traditional sequential methods on two real-world badminton datasets. The performance boost can also be attributed to the inclusion of improved stroke embeddings extracted from the latent representation of a pre-trained large-language model subjected to detailed text descriptions of stroke descriptions. In the discussion, the latent representations learned by the encoder module show useful properties regarding player analysis and comparisons. The code can be found at: This https url.

1. Introduction

In racket sports, players exchange strokes in a rally until one player fails to successfully return a shot, giving the point to the opponent, see Fig. 1. Predictions about strokes of players, drawing on their history of previous strokes, benefit athletes' training and match preparation and can contribute to an improved viewing experience during live broadcasts [5]. Badminton, characterized by swift shot exchanges and strategic shuttle placement, presents a challenge for deep-learning computer vision-based sports analytics. The probabilistic nature of stroke forecasting in racket sports such as badminton [1] complicates predictive analytics due to the inherent unpredictability of player decisions. At any time, players face multiple viable actions. This unpredictability originates from the dynamic interplay of factors such as the physical state, psychological condition, and tactical approach of the player, which influence the selection of potential strokes and strategies. This work aims to design a model capable of incorporating some of the uncertainty-inducing factors into the prediction process.

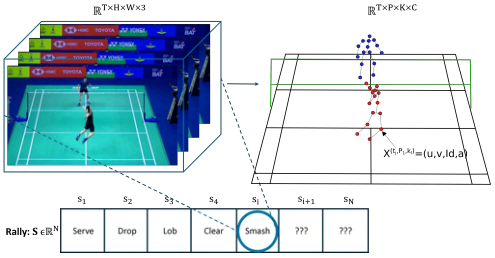


Figure 1. Datastructure overview shows that each stroke/action in a rally, i.e., stroke sequence, is provided the skeleton motion sequence of the stroke for additional context.

One approach to reducing the uncertainty associated with subsequent stroke predictions involves incorporating various otherwise uncertain contextual information into the model. These factors include player skeleton data sequences, player identification (ID), and turn-based rally awareness. Player skeleton data sequences contain the movement and positions of a player's joints over time and have shown increasingly promising results for general action recognition [14, 22, 25, 33] and sports applications [8, 18, 19, 34]. The sequences contain the motion of player strokes. This data can reveal patterns in a player's technique and movement that give information about future strokes, allowing models to account for individual players' physical capabilities and limitations. Incorporating player ID as information allows the predictive models to consider historical performance data and personal playing styles. This individual-specific approach recognizes that each player has unique characteristics (strengths, weaknesses, and strategic preferences) influencing their selection of shots in the game. Theoretically, by identifying these characteristics, the model can better predict a player's likely actions in various game situations. Finally, turn-based rally awareness introduces an extra contextual layer to the prediction [31] by specifying the actor and reactor behind each stroke. Including turn-based nuances allows the model to isolate the

individual player's motion and obtain a clear representation of each stroke. The additional context provided by skeleton data sequences, player ID, and turn-based rally awareness attempts to construct a method that moves beyond the basic statistical probability of the rally sequences and instead embraces a more holistic understanding of the game. This approach aims to predict the next stroke and simultaneously capture the underlying (player) process behind stroke selection in racket sports.

This paper builds upon concepts of our previous work [17] and presents a transformer-based model for action forecasting in badminton.[1] The primary contributions of this research include:

1. A skeleton-based spatiotemporal encoder that uses transformer and pooling blocks to learn representations for enhancing next stroke predictions in badminton.
2. An adaptive cross-attention decoder that incorporates contextual stroke descriptors from high-dimensional embeddings of a pre-trained language model (LM).
3. Examples of how the latent variables can be used for match and playstyle analysis.

The following sections will detail the methodology behind this approach, the architecture of the transformer-based model, and the findings from various experimental evaluations.

2. Related Work

2.1. Action forecasting

Prior works have attempted to develop a wide range of neural network models to forecast future action sequences from observed action labels or extracted video features. The paper [11] introduced a method using a recurrent neural network (RNN) - hidden Markov model to classify actions from video frames, followed by a convolutional neural network (CNN) or RNN that predicted the following actions in the sequence. In [20], they employed a variational multi-headed GRU to predict future actions and their duration. They showed that their approach worked for both one-hot action labels and extracted video features. Similarly to our approach, [12] suggested jointly using both frame and annotation features to improve the prediction capacity of their model. [23] employed sequence-to-sequence models using a gated recurrent unit (GRU) encoder-decoder architecture to predict future actions from RGB frames alone. To our knowledge, skeleton data is not commonly used as a modality for in-action sequence forecasting. Instead, action labels are used to condition observed skeleton sequences to generate future skeleton sequences. [13, 24] employs variational autoencoders for this task.

[1]Our code is available on github https://github.com/MagnusPetersenIbh/RallyTemPose.

2.2. Data analytical sports applications

Action recognition tasks fill up the majority of sports-focused research in the field of computer vision. Here, convolutional neural networks (CNN) have been used for feature extraction on RGB images [26]. Classification algorithms such as Support Vector Machines then use the extracted features to make predictions. Transformer models have also gained traction for spot application tasks. In [3], a Vision Transformer (ViT) [9] is used as the backbone to do group activity recognition (GAR) in Volleyball and basketball.

Skeleton data, as opposed to image data, has proven effective for the analysis and recognition of activities in various sports, including Tai Chi [8, 10, 30] and fencing [21, 34]. Skeleton-based Temporal convolutional networks (TCN) have seen use for action recognition in table tennis [18], where TCNs performed better than LSTM models. In badminton, [19] used skeleton data and shuttle trajectory data in a GRU model to perform binary hit detection. They further improved the detection rate by using badminton-specific rules. Specifically for stroke prediction [31] employed a transformer-based player and position-aware model that used prior stroke types and shuttle placement to predict future position and type of strokes. Instead of the shuttle placement, our work uses the players' skeleton and ground motion to provide a dynamic understanding of each stroke as the basis for predicting the subsequent strokes in the sequence.

3. Task formulation

In action forecasting for racket sports such as badminton, the strokes are the central actions. A stroke is the motion of a player preparing to hit the shuttle until shortly after contact between the racket and the shuttle. The exchange of strokes between players, called a rally, continues until one player fails to return the opposing player's stroke. The scientific objective is to predict the next stroke within a rally based on previously executed strokes while also considering the actual motion of players by incorporating 2D skeleton pose data. A pose $K_j^{(i)}$ within a stroke s_i (ith stroke in the sequence) captures the spatial configuration of the player's body at a given time frame j, represented by a set of keypoints that denote the 2D image positions of the body joints. Additionally, the sequence $G = g_1^{(i)}, .., g_j^{(i)}, ..g_T^{(i)}$, representing the 2D positions of the players' feet on the ground plane for each frame, is sampled and structured as $g_j^{(i)} \in \mathbb{R}^{T\times 2}$, as an additional data source. A rally S is denoted $S = [s_1, \ldots, s_N]$, where s_i is the ith stroke within the rally. Each stroke is described by a sequence of poses $K_1^{(i)}, \ldots, K_T^{(i)}$, with T representing the duration of a stroke sequence and N the number of strokes in a rally.

The goal is to predict the subsequent stroke s_{i+1} in the rally sequence S and show that leveraging both the historical sequence of strokes and motion provided by the 2D skeleton poses improves the prediction rate.

4. Model

This section describes the concepts of the autoregressive stroke prediction model, RallyTemPose. The main contribution of the model is that the encoder module takes the skeleton data and player ground condition as additional data and computes an embedded representation that conditions the rally sequence to predict the next stroke in the rally,

$$p(s_{i+1}|s_{1:i}, K_{1:i}, G_{1:i}, I) = \text{Dec}(s_{i:1}, \text{Enc}(K_{1:i}, G_{1:i}, Id)). \tag{1}$$

The overview of RallyTempose can be seen in Fig. 2.

4.1. Encoder

The encoder consists of a linear projection layer that embeds the raw data frames of player positions and skeleton poses into tokens. A learnable joint encoding (JE) is added to the tokenized data to provide information about the joint arrangement of the skeleton data. The Spatial Transform (ST) then applies a pose-wise transformer mechanism focusing on spatial relationships between keypoints in the player's movements. The ST is followed by a Grouped Pooling Block (GPB), which aggregates information, reduces dimensionality, and focuses on the relevant features of the players' movements. The Temporal Transformer (TT) focuses on the temporal dynamics, processing the pose movements over time. An important detail is that for the ST and TT blocks, both the inter-player (cross-attention) and intra-player (self-attention) attention is computed; see Fig. 3 for a visual depiction. The temporal transformer is followed by another GPB that pools over the embedded temporal representation. The final step produces (see Fig. 2) the three latent variables: a stroke representation z_s, a player 1 representation z_1, and a player 2 representation z_2. z_s merges the processed representations of both players for each stroke, providing complete context for each stroke. The player representations, on the other hand, are limited to information about one specific player.

Transformer Block:
In the transformer block, see Fig. 2, the layer normalized input is first subjected to the multi-headed self-attention (MHSA) mechanism that computes attention scores after being masked with either casual or padding mask (hides padded or future token from getting attention).

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2}$$

Q, K, and V represent Queries, Keys, and Values, all being learned linear projections of the embedded representation vectors. The transformer block employs multi-head attention by splitting Q, K, and V into multiple heads h for parallel processing: $\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_h)W^O$. Following this, a fully connected network (FC) applies nonlinear transformations to each position independently:

$$\text{FC}(x) = \text{GELU}(\text{Norm}(x)W_1 + b_1)W_2, \tag{3}$$

where both the MHSA and FC block have residual connections, GELU activations, first proposed in [16], and Norm refers to a layer normalization.

Group Pooling Block:
The GPB, shown in Fig. 2, based on [14], but here used in connection with transformer blocks instead of fully connected layers aggregates global and local information in embedded data through global and local max pooling. The pooling module operates on an embedded tensor $X \in \mathbb{R}^{G\times N\times D}$, split into select groups. N, G, and D denote the number of groups, group size, and the feature dimension, respectively. First, a global max pooling operation over the features in all groups with

$$M_d = \text{Gpool}(X)_d = \max_{n,g} X_{n,g,d}, M \in X \tag{4}$$

thus captures the most significant activations across all groups and instances for each feature. Simultaneously, local max pooling (Lpool) is executed by pooling over the group in X to create N features vector with the aggregated D features, yielding

$$Q_{n,d} = \text{Lpool}(X)_{n,d} = \max_{g} X_{n,g,d}. \tag{5}$$

The locally pooled features are then concatenated with the globally pooled features (expanded to match local dimensions), yielding a tensor $Y \in \mathbb{R}^{N\times 2D}$.

$$Y_{n,d} = \text{Concat}[\text{Lpool}(X)_{n,d}, \text{Gpool}(X)_d]. \tag{6}$$

Lastly, an FC layer maps it back to the feature dimension D.

4.2. Decoder

In the decoder, an embedding layer maps the one-hot encoded stroke sequences into stroke tokens. Subsequently, the turn-based nature of badminton is exploited by adding the specific player representations (z_1 or z_2) of the player performing the actual stroke. Through a self-attention module, the player-embedded stroke sequence is initially encoded. Subsequently, the decoder block (DB) uses cross-attention mechanisms to condition each stroke on the skeleton-based stroke representations z_s from the encoder. The final component, an MLP Head, takes the output from

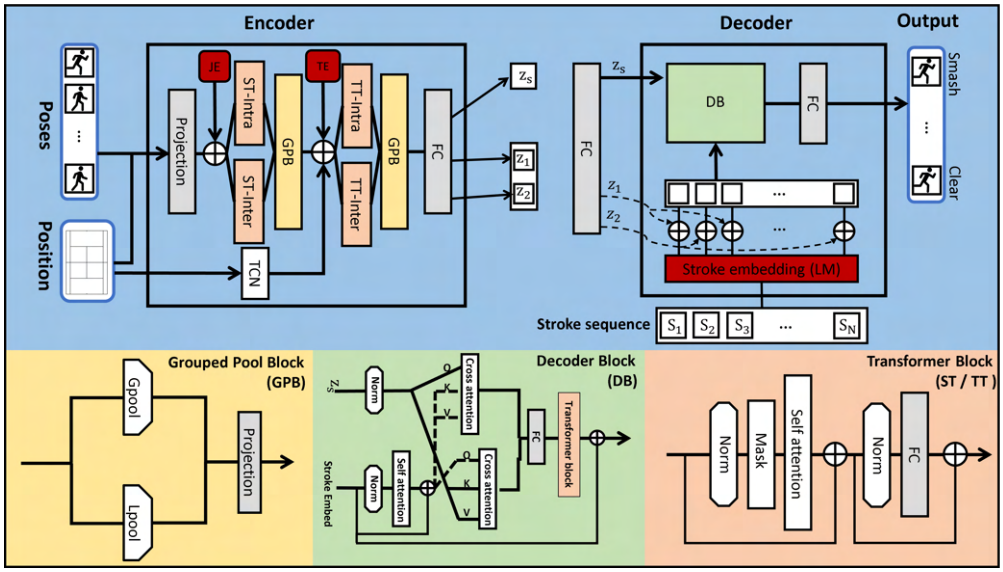


Figure 2. Overview of our approach in, with corresponding components. The abbreviations refer to the following: JE: learned joint encoding added to each pose keypoint, TE: learned temporal encoding added to the frame level tokens in a stroke, ST: spatial transformer, TT: temporal transformer, GPB: group pooling block, FC: fully connected, TCN: temporal convolutional network smoothing over the player ground positions, DB: decoder block.

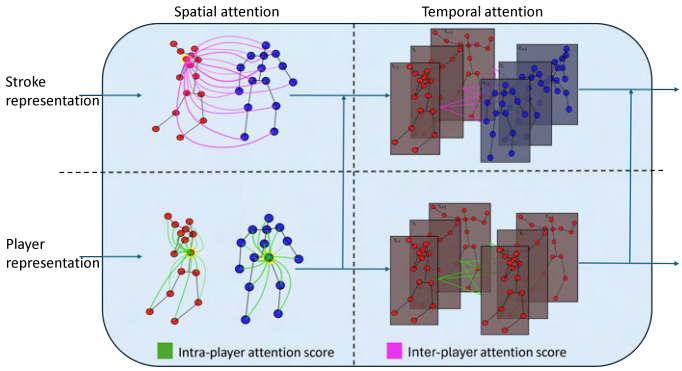


Figure 3. Illustration of the different types of attention present in the encoder module.

the DB to predict the probability of the next stroke in the sequence.

Decoder Block:

The Decoder Block (DB) combines self-attention, dual cross-attention, and an adaptive fusion mechanism. The block employs layer norms to ensure stability during the forward pass. The target embedded stroke sequence is first subjected to MHSA, after which encoder-to-decoder cross-attention and decoder-to-encoder reverse cross-attention are applied, facilitating stronger incorporation of the encoder representations z_s of the stroke motion. This is further ensured through an adaptive fusion layer that linearly combines the outputs of the dual cross-attention, which a standard Transformer Block subsequently processes for final refinement.

4.3. Enhanced Stroke Embeddings

Another aspect of our model is its pre-trained Language Model (LM) utilization. Specifically, BERT [7], for embedding various stroke types. Each stroke type is annotated with a description of its characteristics and typical use cases. From these descriptions, a high-dimensional stroke embedding is processed and extracted from the latent layer of a pre-trained BERT model. The LM representation provides more detailed embeddings than those derived from a learned embedding on a comparatively smaller dataset than the one on which the LM model was trained.

4.4. Training

The makeup of a transformer allows for $N-1$ training samples to be created from a N stroke rally S. In each training sample, the last stroke functions as the prediction target of the model, while all prior strokes in the rally serve as the observed sequence. This strategy allows for variable-length training sequences, allowing the model to observe the connection between all possible strokes in a rally during training. Sequence diversity helps the model avoid overfitting. The network is trained using two loss functions. First, we minimize the cross-entropy loss between the target and pre-

dicted strokes:

$$\mathscr{L}_{main}(s_{i+1}, \hat{p}_{i+1}) = -\sum_{j=1}^{C} s_{i+1}^{(j)} \log(\hat{p}_{i+1}^{(j)}), \tag{7}$$

where C is the number of stroke classes, s_{i+1} is the one-hot encoded target stroke, and $\hat{p}_{i+1}$ is the predicted stroke type probability vector. Second, an auxiliary objective is defined on the output of the encoder's latent stroke variable, z_s. The cross-entropy of the linear projection, $\hat{a}_i$, of the latent stroke variable $z_{s_{(i)}}$ and the corresponding stroke type s_i is minimized as

$$\hat{a}_i = W_{aux} z_{s_{(i)}} + b_{aux}, \tag{8}$$

$$\mathscr{L}_{aux}(s_i, \hat{a}_i) = -\sum_{j=1}^{C} s_i^{(j)} \log(\hat{a}_i^{(j)}). \tag{9}$$

Here, both objectives are described for a single stroke denoted by the subscript i, but in practice, the loss is the average of all strokes in a sequence. The total loss is the weighted sum of the two losses

$$\mathscr{L} = \gamma \mathscr{L}_{aux} + \mathscr{L}_{main}, \tag{10}$$

where γ is a hyperparameter, $\gamma = 0.3$ during experiments.

5. Experiments

5.1. Datasets

ShuttleSet The ShuttleSet [32] dataset contains 42 professional matches from 2018 to 2021, featuring 26 players across men's and women's singles categories. It is composed of more than 3000 rallies and 34000 strokes, with an average rally length of 10 strokes. Domain experts annotated the strokes in the dataset into 10 distinct shot types: *net shot*, *clear*, *push/rush*, *smash*, *defensive shot*, *drive*, *lob*, *dropshot*, *serve*, and *unknown/error*. The number of strokes for each type can be seen in Tab. 1. For model training and testing, the dataset is divided such that 80% of the rallies from each match are used for training, ensuring comprehensive player history, and the remaining 20% for testing.

BadmintonDB The BadmintonDB [2] dataset consists of 9 annotated video data professional men's singles matches. The dataset includes 811 rallies and 9,671 strokes, all featuring the players Kento Momota and Anthony Sinisuka Ginting. The dataset provides annotation of the strokes into 10 distinct types, that follow the recommended coaches' guide of the Badminton World Federation (BWF). The shot types are almost identical to the shuttleset data, see Tab. 1 for the stroke distribution. The same two players play in all matches, hence the 2 complete matches are reserved for testing and the remaining 7 for training.

5.2. Skeleton pose extraction

The pose extraction workflow involves two key stages: adopting techniques from [4, 6] for human detection and pose estimation and utilizing the HRNet framework [27] for precise 2D pose estimation. The presence of non-participants, like spectators, can adversely affect the skeleton data's quality by including poses of irrelevant characters. To tackle this, a homography based on the badminton court dimensions is computed to map detected individuals' feet to the ground plane, ensuring the focus is solely on players. This method also distinguishes between the top and bottom players in each sequence. Missing skeleton data is addressed by linear interpolation between the preceding and future frames. Pose normalization involves centering and scaling to standardize the bounding box diagonal to one.

5.3. Evaluation metrics

In badminton, more than one stroke is often a viable choice, which should be reflected in the evaluation metrics. The performance of the models is judged based on the accuracy (acc) of their prediction, the top-2 accuracy (acc2), and the top-3 accuracy (acc3).

5.4. Baselines

No other existing work uses stroke skeleton data to enhance future stroke prediction capabilities. Therefore, our model performance is compared to other sequence and action prediction baselines not explicitly designed for badminton. All model baselines consist of current state-of-the-art concepts for sequence prediction, and thus, while not intentionally designed for badminton stroke prediction, comparing to the baselines allows for a good estimate of the prediction capabilities of our specific model. The following baselines are used for comparison:

- Seq2Seq [28]
- Transformer [29]
- Actionformer [22] + Transformer decoder

5.5. Implementation details

The dimension of embedded representation (d) per head is set to 16, the number of heads (h) in the MHSA is set to 4, and the forward expansion in the inner dimension of feed-forward layers is set to 4 following[15]. A rally's max sequence length (s) is set to 35, and T varies for different rallies. Similarly, the max temporal length of each stroke motion (T) is set to 30. Dropout and Attention dropout are utilized in each MHSA block with a drop rate of 0.3. The models are trained with a batch size of 1 using AdamW with a learning rate set to 10^{-4}. Zero padding is performed for individual stroke motion sequences. Padding the rallies was also tested but did not improve performance.

Table 1. Distribution of the data classes for the two datasets.

	Net-Shot	Defensive-Shot	Smash	Lob	Clear	Drive	Dropshot	Push/Rush	Serve	Error
ShutSet	6716	3836	3749	4614	2440	1091	2929	3021	2060	1095
BadminDB	1756	1281	1154	1954	596	188	108	715	108	131

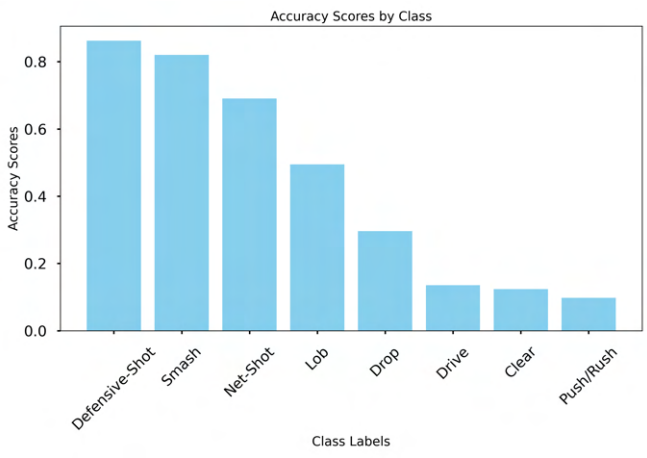


Figure 4. Comparisons of class accuracy for the different stroke types in the ShuttleSet dataset.

5.6. Main Experiments

In the comparative analysis of predictive models on the ShuttleSet and BadminDB datasets, our model outperforms the other baseline models in standard and top-3 accuracy. On the ShuttleSet dataset, it achieves an accuracy of 54.3%, a top-2 accuracy of 77.3%, and a top-3 accuracy of 92.5%, indicating its ability to rank the correct outcome within the top three predictions in over 90% of the cases. In the BadminDB dataset, our model achieves an accuracy of 62.8% and a top-3 accuracy of 93.1%. The BadminDB is much smaller than ShuttleSet, which resulted in our model often overfitting. As a result, the much simpler sequential models perform better on BadminDB comparatively, but RallyTempose still slightly outperforms them in the end.

The results show the model's prediction prowess and reflect its ability to select the most logical outcomes. For a given situation, multiple stroke candidates can be perfectly viable simultaneously. The results in both accuracy metrics, especially in the top-3 accuracy, suggest that our model's way of incorporating skeleton-motion and player-specific information improves the prediction logic compared to the baselines in the context of badminton datasets.

Logical misclassifications: In Fig. 4, the specific accuracy and misclassification ratio for all stroke types is plotted. Strokes like the smash are accurately predicted, while strokes like the clear and drives are only correctly predicted 12% and 14% of the time, respectively. However, by examination of the confusion matrix in Fig. 5, most classifications can be attributed to logical reasoning, and all misclassifications belong to sensible groups (Net-shots, push-rush, and lobs), (drives and defensive shots) and, (smash, clears and drops). For example, a clear is predominantly hit from the backcourt on shuttle trajectories and racket swings similar to a smash and, to a lesser degree, a drop. This is consistent with the faulty prediction of clears being smashes and drops, and hence the predictions follow an underlying logic of the game. Similarly, a drive can easily be confused with a defensive reaction shot. Our model can still be improved further. We hypothesize that a deeper strategic understanding of each situation can increase accuracy even more. However, the results indicate that our model, through purely next-stroke action prediction, has developed a rudimentary game understanding.

6. Discussion

6.1. Ablation Study

The impact of our skeleton-based stroke condition on the prediction capability is examined through an ablation study. The relative contribution of 1.) skeleton data, 2.) ground position of the players, and 3.) specific player embedding is determined through six different model variants. In 3 the respective prediction accuracies are shown after removing specific model inputs and their corresponding model components. The results show that the most critical factor is the inclusion of the player ground position, as leaving out this data along with the TCN block leads to a 2.6% drop in performance. The encoder version made up solely of a TCN block achieves a 51.6% accuracy. The player-specific information does not significantly boost the prediction accuracy, however, as shown in the next section, learning player-specific representations allows for introspective player analysis that can be extrapolated from the model. Since including the players' ground positions results in a significant performance boost. A potentially even greater performance increase could be obtained by including 3D skeleton data as well.

6.2. Match Analysis Prospects

The model's design allows for player comparison by analyzing the latent variables of the model. Fig. 6 and Fig. 7 show t-SNE plots of the latent variables. In the visualization z_s are colored based on the target stroke they represent,

Table 2. Accuracy (Acc), Top-2 Accuracy (Acc-2), and Top-3 Accuracy (Acc-3) of our models and other baselines on the ShuttleSet and BadminDB datasets.

	ShuttleSet			BadminDB		
Model	Acc (%)	Acc-2 (%)	Acc-3 (%)	Acc (%)	Acc-2 (%)	Acc-3 (%)
Seq2Seq (LSTM)	47.9	72.4	83.5	57.3	82.3	86.0
Transformer	49.8	73.9	87.2	61.5	85.4	92.5
POT + Trans Dec	52.1	74.1	91.2	58.4	82.0	91.7
RallyTemPose	54.3	77.3	92.5	62.8	83.5	93.1

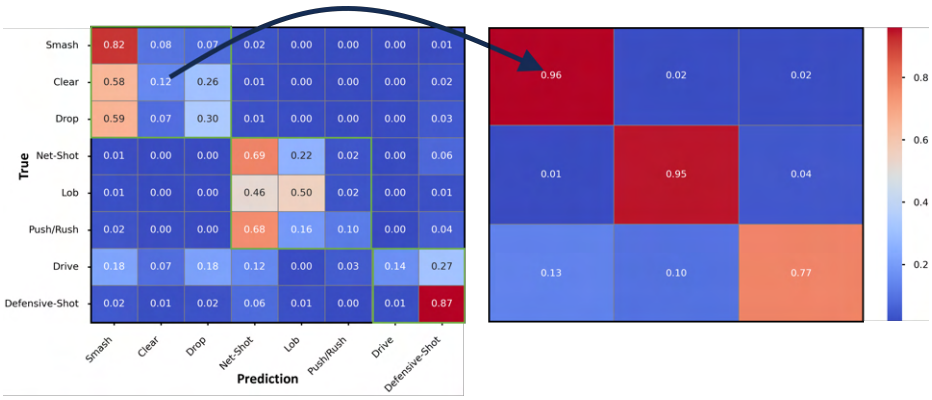


Figure 5. To the left is the confusion matrix for the shuttles data, and to the right is the confusion matrix grouped according to logical classes.

Table 3. Ablation Study of RallyTemPose model.

Keypoint	Ground	Player Rep	Accuracy (%)
			48.3
✓			49.2
		✓	46.9
	✓		51.6
	✓	✓	50.1
✓	✓		52.4
✓		✓	51.7
✓	✓	✓	54.3

whereas z_1 and z_2 are colored according to the players they represent. Clear groupings are observed for the different z_s stroke variables and partial groupings of the player variables. This indicates that z_s and, to a lesser degree, z_1 and z_2 stores relevant information about strokes and playing styles respectively. While the specific player embedding does not significantly improve the model's prediction accuracy, it allows for model intrinsic playstyle comparisons.

Player Similarity We can project the playstyle similarity of different players by looking at the cosine similarity of the player-specific latent variable for the other players in the dataset. The cosine similarity is calculated by random sampling of $N = 1000$, strokes, for each of the pair combinations of players and calculating the average cosine similarity between the latent player variables as

$$\text{Player Sim}_{i,j} = \sum_{n=1}^{N} \frac{\mathbf{z_i^n} \cdot \mathbf{z_j^n}}{\|\mathbf{z_i^n}\| \|\mathbf{z_j^n}\|}. \tag{11}$$

Tab. 4 shows the cosine similarity between the latent variables of players for 5 different players. Observe that there is a notable difference in similarity between the players. On average, the male (first 3 players) and female (last 2 players) have a lower similarity, whereas the same gender similarity scores are higher. However, the player similarity score is also quite low between the 3 males. This, however, is quite sensible since Male 3, known for a unique, endurance-based, hard-to-read playstyle, Male 2, with a very fast-paced style, and Male 1, with a physical and powerful playstyle, are very different players, and the similarity score reflects that. Future work could include categorizing distinct playstyles and attempting to interpret them as defensive, offensive, power, placement, etc.

Play-style analysis In Fig. 8, a bar plot of the average accuracy for each player in the ShuttleSet dataset is shown. There is a notable gap of more than 20% average accuracy

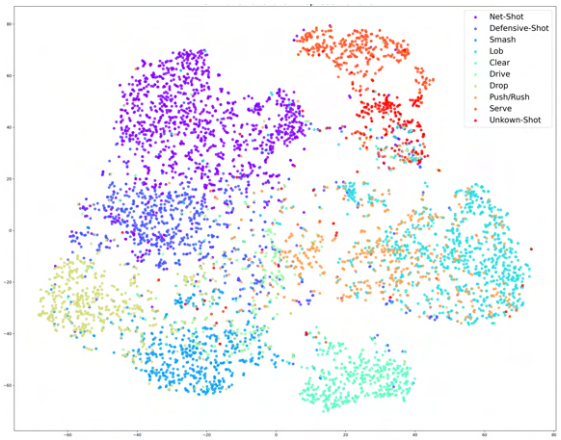


Figure 6. t-SNE plot over the latent stroke z_s representation, colored according to the observed stroke types.

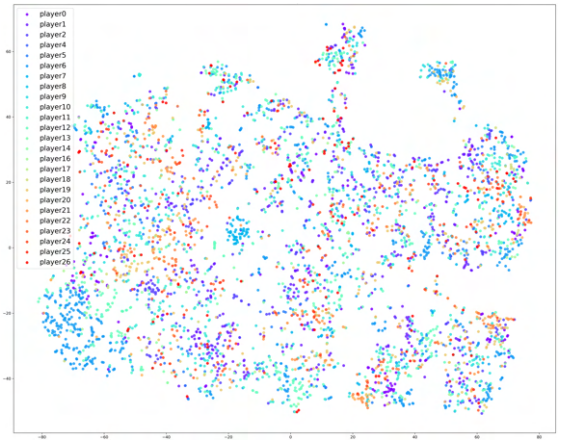


Figure 7. t-SNE plot over the latent player representation (z_1 and z_2), colored according to the target player Id. Note the lack of very distinct groupings of the player variables, which could be explained by the difference/similarity in how players perform certain strokes.

Table 4. Cosine similarity between latent player variables of different classes. (M: male, F: female)

Player sim	M1	M2	M3	F1	F2
M 1	0.61				
M 2	0.43	0.58			
M 3	0.37	0.41	0.67		
F 1	0.21	0.19	0.31	0.71	
F 2	0.23	0.51	0.49	0.57	0.65

between the players, which strokes are predicted the best compared to the player predicted the worst. The prediction accuracy of specific players could be used to indicate how well players can mask their strokes. However, the approach

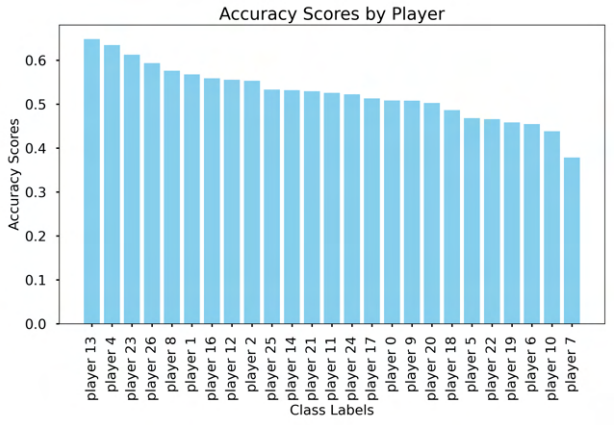


Figure 8. The average accuracy of next stroke predictions for all the players in the dataset.

assumes the model can flawlessly predict straightforward strokes, which is not yet guaranteed. Still, through continuous improvement of the model, this could be a helpful asset for player analysis.

6.3. Future prospects

Looking ahead, we aim to enhance the model's capabilities by incorporating additional variables, such as match outcomes (win/loss), to facilitate more sophisticated tactical analysis. Additionally, expanding the model to predict the skeleton sequence of the predicted strokes would be beneficial not only for sports analysis purposes but also for creating synthetic data in a field where quality annotated datasets are sparse.

7. Conclusion

This research introduced a model specifically designed for stroke prediction in badminton, utilizing an encoder-decoder architecture. The model integrates skeleton data and player-specific information using a spatiotemporal transformer encoder. Our experiments, conducted on two different real-world badminton datasets, show an increase in performance for our approach compared to other forecasting baselines. Furthermore, the extracted latent representations show potential use for player analysis and match preparation.

Acknowledgements. We are grateful for the financial support provided by the Novo Nordisk Foundation, which facilitated our research as part of the TeamSPORTek initiative. Additionally, our thanks extend to Badminton Danmark and Team Danmark for their valuable contributions, including data collection and providing expert knowledge on badminton.

References

[1] Yazan Abu Farha and Juergen Gall. Uncertainty-aware anticipation of activities. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 1

[2] Kar-Weng Ban, John See, Junaidi Abdullah, and Yuen Peng Loh. Badmintondb: A badminton dataset for player-specific match analysis and prediction. In *Proceedings of the 5th International ACM Workshop on Multimedia Content Analysis in Sports*, page 47–54, New York, NY, USA, 2022. Association for Computing Machinery. 5

[3] Naga VS Raviteja Chappa, Pha Nguyen, Alexander H. Nelson, Han-Seok Seo, Xin Li, Page Daniel Dobbs, and Khoa Luu. Spartan: Self-supervised spatiotemporal transformers approach to group activity recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 5158–5168, 2023. 2

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5

[5] Wei-Ta Chu and Samuel Situmeang. Badminton video analysis based on spatiotemporal and stroke features. In *Other Conferences*, pages 448–451, 2017. 1

[6] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020. 5

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. 4

[8] Lingxiao Dong, Dongmei Li, Shaobin Li, Shanzhen Lan, and Pengcheng Wang. Tai chi action recognition based on structural lstm with attention module. In *Other Conferences*, 2019. 1, 2

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 2

[10] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1110–1118, 2015. 2

[11] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When will you do what? - anticipating temporal occurrences of activities. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5343–5352, 2018. 2

[12] Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Forecasting future action sequences with neural memory networks. In *British Machine Vision Conference*, 2019. 2

[13] Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong, and Li Cheng. Action2motion: Conditioned generation of 3d human motions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2021–2029, 2020. 2

[14] Ryo Hachiuma, Fumiaki Sato, and Taiki Sekii. Unified keypoint-based action recognition framework via structured keypoint pooling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22962–22971, 2023. 1, 3

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021. 5

[16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3

[17] Magnus Ibh, Stella Graßhof, Dan Witzner, and Pascal Madeleine. TemPose: a new skeleton-based transformer model designed for fine-grained motion recognition in badminton. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 5199–5208, 2023. ISSN: 2160-7516. 2

[18] Kaustubh Milind Kulkarni and Sucheth Shenoy. Table tennis stroke recognition using two-dimensional human pose estimation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4571–4579, 2021. 1, 2

[19] Paul Liu and Jui-Hsien Wang. MonoTrack: Shuttle Trajectory Reconstruction From Monocular Badminton Video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, page 10, 2022. 1, 2

[20] Siyuan Brandon Loh, Debaditya Roy, and Basura Fernando. Long-term action forecasting using multi-headed attention-based variational recurrent neural networks. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2418–2426, 2022. 2

[21] Filip Malawski and Bogdan Kwolek. Improving multimodal action representation with joint motion history context. *Journal of Visual Communication and Image Representation*, 61: 198–208, 2019. 2

[22] Vittorio Mazzia, Simone Angarano, Francesco Salvetti, Federico Angelini, and Marcello Chiaberge. Action transformer: A self-attention model for short-time pose-based human action recognition. *Pattern Recognition*, 124:108487, 2022. 1, 5

[23] Yan Ng and Basura Fernando. Forecasting future action sequences with attention: A new approach to weakly supervised action forecasting. *IEEE transactions on image pro-*

cessing : a publication of the IEEE Signal Processing Society, PP, 2020. 2

[24] Mathis Petrovich, Michael Black, and Gul Varol. Action-conditioned 3d human motion synthesis with transformer vae. In *ICCV*, pages 10965–10975, 2021. 2

[25] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. Skeleton-based action recognition via spatial and temporal transformer networks. *Computer Vision and Image Understanding*, 208-209:103219, 2021. 1

[26] Nur Azmina Rahmad and Muhammad Amir As'ari. The new convolutional neural network (cnn) local feature extractor for automated badminton action recognition on vision based data. *Journal of Physics Conference Series*, 1529, 2020. 2

[27] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep High-Resolution Representation Learning for Human Pose Estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5696, Long Beach, CA, USA, 2019. IEEE. 5

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press. 5

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. 5

[30] Pengcheng Wang and Shaobin Li. Structural-attentioned lstm for action recognition based on skeleton. In *Other Conferences*, 2018. 2

[31] Wei-Yao Wang, Hong-Han Shuai, Kai-Shiang Chang, and Wen-Chih Peng. Shuttlenet: Position-aware fusion of rally progress and player styles for stroke forecasting in badminton. In *AAAI*, pages 4219–4227. AAAI Press, 2022. 1, 2

[32] Wei-Yao Wang, Yung-Chang Huang, Tsi-Ui Ik, and Wen-Chih Peng. Shuttleset: A human-annotated stroke-level singles dataset for badminton tactical analysis. In *KDD*, pages 5126–5136. ACM, 2023. 5

[33] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *AAAI*, 32, 2018. 1

[34] Kevin Zhu, Alexander Wong, and John McPhee. Fencenet: Fine-grained footwork recognition in fencing. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3588–3597, 2022. 1, 2

Paper 3: SynthNet: 3D trajectory reconstruction from synthetic training data

SynthNet: Leveraging Synthetic Data for 3D Trajectory Estimation from Monocular Video

Morten Holck Ertner
IT-University of Copenhagen
Copenhagen, Denmark
mert@itu.dk

Sofus Schou Konglevoll
IT-University of Copenhagen
Copenhagen, Denmark
sosk@itu.dk

Magnus Ibh
IT-University of Copenhagen
Copenhagen, Denmark
ibhq@itu.dk

Stella Graßhof
IT-University of Copenhagen
Copenhagen, Denmark
stgr@itu.dk

ABSTRACT

Reconstructing 3D trajectories from video is often cumbersome and expensive, relying on complex or multi-camera setups. This paper proposes SynthNet, an end-to-end pipeline for monocular reconstruction of 3D tennis ball trajectories. The pipeline consists of two parts: Hit and bounce detection and 3D trajectory reconstruction. The hit and bounce detection is performed by a GRU-based model, which segments the videos into individual shots. Next, a fully connected neural network reconstructs the 3D trajectory through a novel physics-based training approach relying on purely synthetic training data. Instability in the training loop caused by relying on Euler-time integration and camera projections is circumvented by our synthetic approach, which directly calculates loss from estimated initial conditions, improving stability and performance.
In experiments, SynthNet is compared to an existing reconstruction baseline on a number of conventional and customized metrics defined to validate our synthetic approach. SynthNet outperforms the baseline based on our own proposed metrics and in a qualitative inspection of the reconstructed 3D trajectories.

CCS CONCEPTS

• **Computing methodologies** → *Activity recognition and understanding*; *Tracking*; **Reconstruction**; *Neural networks*; • **Applied computing** → *Physics*.

KEYWORDS

3D Reconstruction, Machine Learning in Sports, Computer Vision, Ball Tracking, Neural Network, Synthetic Data, Differential Equations

ACM Reference Format:
Morten Holck Ertner, Sofus Schou Konglevoll, Magnus Ibh, and Stella Graßhof. 2024. SynthNet: Leveraging Synthetic Data for 3D Trajectory Estimation from Monocular Video. In *Proceedings of the 7th ACM International Workshop on Multimedia Content Analysis in Sports (MMSports '24), October 28-November 1, 2024, Melbourne, VIC, AustraliaProceedings of the 32nd ACM International Conference on Multimedia (MM'24), October 28-November 1, 2024, Melbourne, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3689061.3689073

MMSports '24, October 28-November 1, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-1198-5/24/10
https://doi.org/10.1145/3689061.3689073

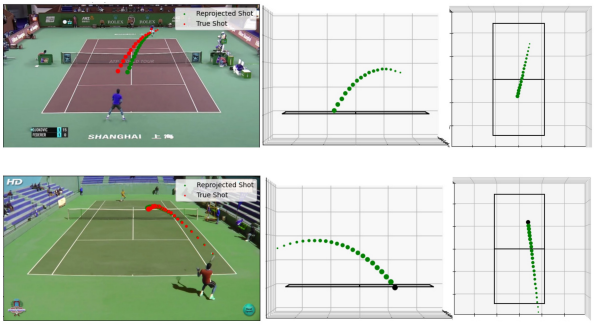


Figure 1: 3D trajectories constructed from monocular video using our end-to-end system SynthNet.

1 INTRODUCTION

In high-level tennis, accurate 3D information on the ball holds great value, both in assisting umpires in calling the game and gathering player and shot information for analytical purposes. Currently, the most popular technology is the Hawk-Eye system[1], which uses multiple cameras and triangulation to capture the ball's 3D position with an error margin of 3.6 millimeters. Such a system is expensive and thus only available for the highest level of tennis players. However, most tennis players are amateurs and often lack access to such a system. Enabling the tracking of the ball's 3D position using a single, everyday camera, such as a phone, would allow the average tennis player to access game statistics previously reserved for professionals.

Estimating 3D ball positions from monocular video has been achieved in other sports like volleyball [2], basketball [1], and badminton [12], but, to our knowledge, has never been achieved in tennis. Thus, the motivation for this paper is to estimate 3D trajectories from monocular video in tennis.
Direct 2D-to-3D lifting of ball coordinates presents a significant

[1]https://www.hawkeyeinnovations.com

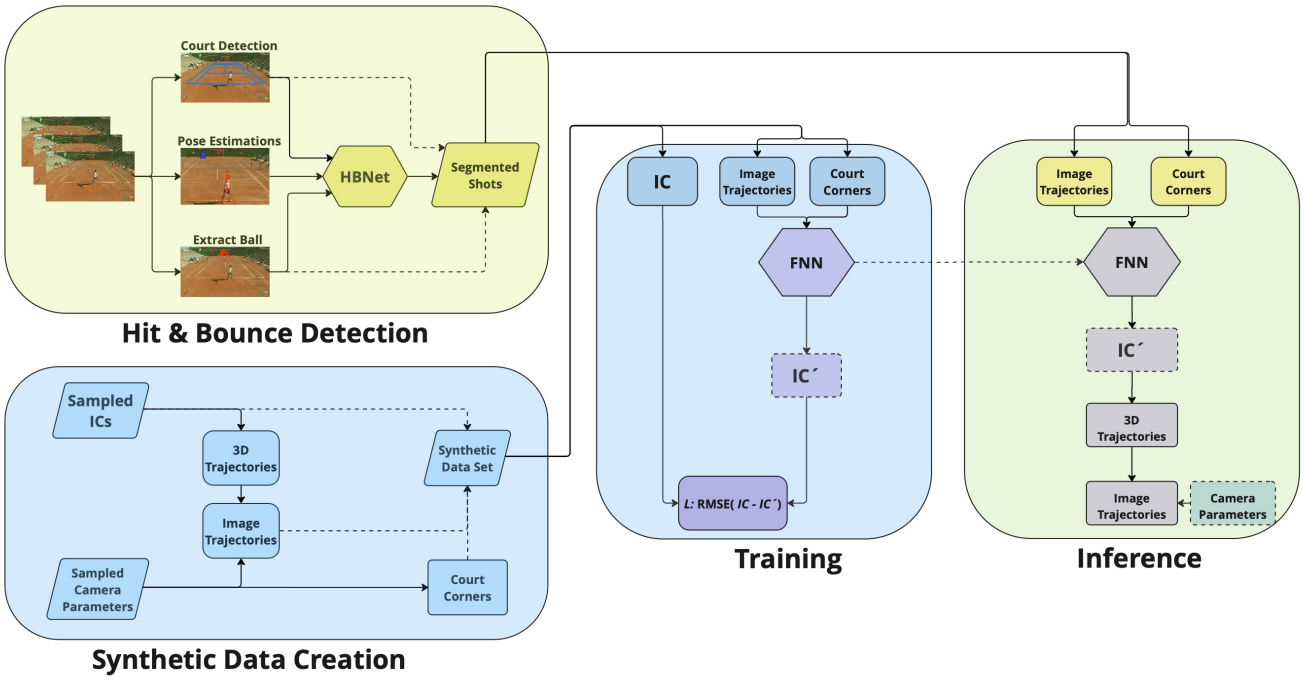


Figure 2: Complete End-to-End Pipeline. *For training*, synthetic data is generated by sampling initial conditions (IC). This data is used to create 3D trajectories and image trajectories along with sampled camera parameters. The feed-forward neural network (FNN) predicts initial conditions based on image trajectories and court corners, and trains using Root Mean Squared Error (RMSE) between true and predicted IC as the loss function. *For inference*, features are extracted from video data and fed into the hit and bounce detection model HBNet. Predictions from HBNet segment the videos into individual shots, which are then used alongside court corners as input to the trained FNN. The FNN predicts a new set of initial conditions, enabling the creation of 3D trajectories that are reprojected onto image coordinates to calculate reprojection error.

challenge since $(2D, 3D)$—pairs of ball coordinates are publicly unavailable. Instead, we segment the game into individual shots and use the physical laws of motion to set up differential equations for the trajectories, after which Euler's method is used to estimate the 3D position of the ball, which can be projected to the 2D image coordinates using the camera parameters. Usually, a training loop involving iterative time integration and subsequent 3D-2D camera projection is too unstable to converge. Instead, we propose a feed-forward neural network that predicts the **initial conditions (IC)** given the image (2D) trajectories as input, which provides a significantly smoother training routine. This is realized by, prior to training the model, random sampling of initial conditions and subsequent trajectory simulation using Euler's method. Then, only keeping trajectories, passing the net, and landing within the court allows for acquisitions of 3D shot trajectories. Lastly, the image coordinates are retrieved by projection to the image plane using randomly chosen camera parameters found from known positions of the tennis court. A model could also be trained on the synthetic (2D,3D) pairs. However, compressing input image trajectories allows for better generalization of real-world trajectories.

To segment the game into shots, we identify the start and end of each shot by detecting hits and bounces. We use the TrackNet Tennis dataset [7], consisting of videos from 10 professional tennis matches with annotated hits and bounces. From the videos, we extract three features: player poses, ball coordinates, and court corner coordinates, to use as input for our hit and bounce detection model, which we call *HBNet*. We evaluate HBNet on the ground truth labels from the TrackNet dataset and use the predictions to segment the videos into individual shots.

In the trajectory experiments, we evaluate the model on the reprojection error between the true and predicted (estimated with [19]) image trajectories. Additionally, realizing the limitation of using reprojection error as the sole metric, we propose tailored metrics to validate the real-world generalization of our synthetically trained model. The complete *SynthNet* pipeline can be seen in Figure 2.

2 RELATED WORKS

A lot of prior works in the field of computer vision and machine learning in sports have focused on individual components such as court detection [3, 21, 22], ball tracking [7, 11, 20] and pose estimation [8–10]. Previous research used those features to perform action recognition in sports. Huang et al. [5, 6] use the ball trajectories

with audio information to detect hits made in a tennis game. Similarly, Shublewska-Paszkowska et al. [18] use 3D tennis movement as input to a graph neural network to predict two actions: forehand and backhand strokes, while Cai and Tang [24] identifies 12 types of tennis shots using a Long-Short Term Memory model.

When doing 3D tracking of sports balls, the most used and most reliable way is to use a multi-camera setup [4, 15, 23, 25], where the scene is captured from multiple angles and triangulation is used to estimate the balls 3D location. While this can give accurate results, it is less cost-efficient than using a single camera as you need access to multiple cameras and the possibility and permission to set them up. Some research in 3D ball tracking from monocular video has been done in several sports such as volleyball [2], table tennis [17], basketball [1] and badminton [12], all using a similar approach. They extract features from video and use optimization techniques to find the best set of initial conditions for a reconstructed 3D trajectory. This method has several flaws. First of all, using optimization techniques means that each shot is optimized one at a time, resulting in long inference times. Secondly, the reprojection error between the reconstructed 3D trajectory and the image trajectory is used as part of the loss function, which is undesirable as this can lead to unrealistic trajectories. Instead, we propose to use a neural network to predict the initial condition, train this network on synthetic data, and use their ground truth initial condition for loss during training.

3 RECONSTRUCTING 3D TRAJECTORIES

3.1 Problem statement

In this paper, we develop an end-to-end pipeline to reconstruct 3D tennis ball trajectories from monocular video. The process involves extracting the ball, court, and player poses to predict hits and bounces during a match. These predictions segment the video into shots, defined from a hit to a subsequent bounce. We deploy a neural network trained exclusively on synthetic data to estimate initial conditions for ballistic 3D trajectories using 2D image trajectories from these segmented shots. Subsequently, we evaluate the model using reprojection error and additional proposed metrics. Training will be done solely using a synthetic dataset where ground truth initial conditions are known, and evaluation will be conducted on segmented shots using the defined metrics. To generate trajectories from initial conditions, we employ Euler's method for solving differential equations.

3.2 Hit Bounce Detection

To detect the hits and bounces in a match, we use a similar architecture to Liu et. Al [12]. We extract court coordinates with an algorithm proposed by Kosolapov Sergey [16], detect poses with RTMO [13], and track the ball with WASB [19]. The poses are filtered by searching for a pose inside each player's court. If there is no pose in the court, it finds the pose closest to the back line. Court, player poses, and the ball is then used as features for a model that predicts hits, bounces, and nonhits. To get the temporal aspect of the movement before and after a shot, we create a Gated Recurrent Unit (GRU) based architecture, called Hit-Bounce Net (HBNet). HBNet consists of an embedding layer with 32 neurons, 6 gated recurrent layers with 32 units, and 0.2 dropout. Followed by a fully connected layer and a softmax which generates confidence scores. The model takes a snippet of 21 frames at a time and predicts whether a hit or bounce occurs in the last 9 frames of a snippet. HBNet achieves an accuracy of 86% and a macro F1-score of 0.84. The shots segmented with the predictions from HBNet are called HBNet + WASB.

3.3 Synthetic Learning Procedure

Using the segmented shots from HBNet we can reconstruct 3D trajectories for each shot. We define a shot to be a hit followed by a bounce or another hit. Estimating each shot individually means we can model the trajectory as a projectile under drag:

$$\frac{d^2\mathbf{x}(t)}{dt^2} = \mathbf{g} - \frac{D}{m}|\mathbf{v}(t)|\mathbf{v}(t), \tag{1}$$

where m is the mass of the ball, $\mathbf{x}$ and $\mathbf{v}$ are the 3D position and velocity vectors, g is the gravitational constant, and D the drag coefficient. As this equation has no analytical solution, we use forward Euler's time integration to retrieve a discretized version of the shuttle position with N time steps. At each step, the acceleration is assumed constant for a small enough time step $\Delta t = t_{n+1} - t_n$, where $n \in N$ is the current step. Thus, for each small Δt, we calculate acceleration $\mathbf{a_{n+1}}$, velocity $\mathbf{v_{n+1}}$, and position $\mathbf{x_{n+1}}$, based on the acceleration, velocity, and position of the current time n step as follows:

$$\mathbf{a}_{n+1} = \mathbf{g} - \frac{D}{m}|\mathbf{v}_n|\mathbf{v}_n \tag{2}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n\Delta t \tag{3}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n\Delta t. \tag{4}$$

To create a 3D trajectory initial conditions $\mathbf{v}_0 = (\mathbf{v}_{x0}, \mathbf{v}_{y0}, \mathbf{v}_{x0})^T$ and $\mathbf{x}_0 = (x_0, y_0, z_0)^T$ is required. A visualisation of this process can be seen in Figure 3. We center the world coordinates in the middle of the tennis court. The y-axis is oriented along the length of the court, away from the camera. The x-axis is oriented across the width of the court, and the z-axis is oriented vertically, perpendicular to the ground.

To predict a set of initial conditions we use a feedforward neural network (FNN) consisting of four hidden layers. The model architecture can be seen in Figure 4. The goal of the model is to predict a set of the six initial conditions that create the best-fitting 3D trajectory. We only train the model on synthetic data, where the ground truth initial conditions are known, and we calculate the loss as:

$$L = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(IC_i - \tilde{IC_i})^2}, \tag{5}$$

where IC is the ground truth initial conditions and $\tilde{IC}$ the predicted initial conditions.

For inference, the FNN uses the extracted 2D ball coordinates and court corners (extracted from the videos) to make predictions of initial conditions. We estimate a 3D trajectory based on the initial conditions and reproject it to image coordinates using a perspective transformation. We find the camera parameters for each video with camera calibrations assuming known world coordinates and

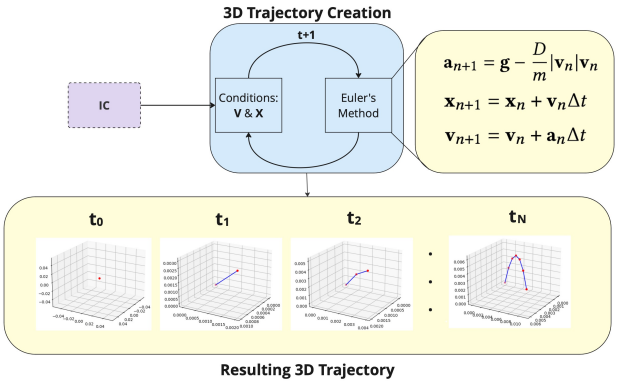


Figure 3: Visual presentation of creating 3D trajectory given a set of initial conditions. The loop runs in N iterations, and with each time step a position and velocity is created.

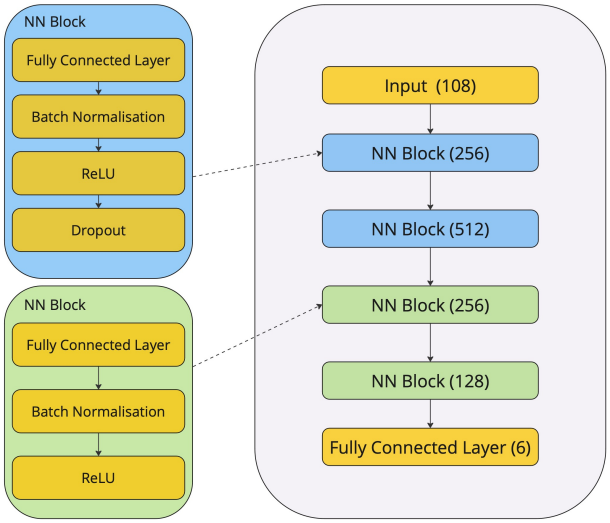


Figure 4: Model Architecture. The module has an input size of 108, 4 NN blocks, and an output size of 6.

corresponding image coordinates. The corresponding coordinates are comprised of the court corners and manually annotated net-pole coordinates. Based on this, we calculate the reprojection error (RE) between the real image trajectory (IT) and predicted image trajectory ($\tilde{IT}$) using root mean squared error (RMSE):

$$RE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(IT_i - \tilde{IT}_i)^2}, \tag{6}$$

3.4 Proposed Evaluation Metrics

We introduce three alternative evaluation metrics based on the 3D landing position of the ball: *Landing Error* (LE), *Tile Accuracy* (T.acc) and *Tile F1-score* (T.F1). To find the 3D landing position of the image trajectories, we use a homography to find the position of

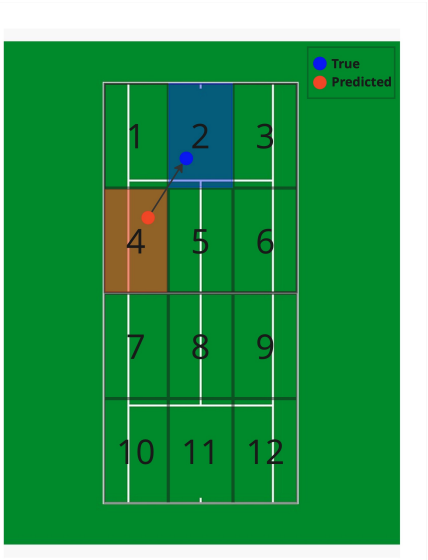


Figure 5: Explanation of Landing error and tile accuracy/F1-score. The predicted landing position, given label 2 is depicted as red dot. The blue dot is the true landing position, given label 4, found using homography on the image trajectory. The landing error is the distance between the two landing positions and tile accuracy/F1-score is calculated based on their given label.

the ball when a bounce occurs. The Landing error is the distance between the real and predicted landing position

$$LE = \sqrt{(x_l - \tilde{x_l})^2 + (y_l - \tilde{y_l})^2}, \tag{7}$$

where (x_l, y_l) is the true landing position and $\tilde{x_l}, \tilde{y_l}$ the predicted position.

To define the tile accuracy and tile F1-score we divide the court into 12 tiles, six on each side of the net, and assign the position a label corresponding to the tile they land in as seen in Figure 5. Using these labels we can calculate accuracy and F1-score.

Lastly, on synthetically created data where we have the true IC and therefore the true 3D trajectory, we can use the *reconstruction error* (RecE) as an evaluation metric. The reconstruction error is the mean Euclidean distance between the predicted and the true 3D trajectory

$$RecE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - \tilde{x_i})^2 + (y_i - \tilde{y_i})^2 + (z_i - \tilde{z_i})^2}. \tag{8}$$

4 EXPERIMENTS

4.1 Dataset and Implementation Details

We use the TrackNet tennis dataset [7], which contains 96 rallies spread across 10 different games of both men and women. All videos are statically filmed from an overhead *broadcast* view behind one of the backlines, with a resolution and frame rate of 1280x720 and

30 fps. The videos have annotated hits, bounces, and ball image coordinates for each frame, along with a visibility score for the ball. Additionally, we annotated some hits and bounces that were missing in the original annotations.

We create additional synthetic data to support the *real* data, by sampling random combinations of initial conditions, and use these conditions to create 3D trajectories. The sampling is done within boundaries such that the trajectory starts within the sidelines and not too far behind the backlines, and such that the velocity is below the highest recorded velocities in tennis.

To use the trajectories as input for our model, we reproject them down onto image coordinates using camera parameters sampled from the *real* data. The court corners corresponding to the camera parameters are used as input. We create 10000 shots, 5000 from each side of the court, all of varying lengths and ending when they hit the ground. The synthetic data allows us to measure both reconstruction and reprojection errors. The real and synthetic data are both divided into train and test, with two games reserved for testing and eight for training. With the synthetic data, we divide based on which game the sampled camera parameters are taken from. The image trajectories are padded, using -1, to have a length of 50 frames to create inputs of equal length, and trajectory and court coordinates are scaled down by the image dimensions. The model is implemented in Pytorch using a dropout of 0.2 and solely trained on the synthetic data for 25 epochs with early stopping.

4.2 Model Evaluation

To evaluate our model, we define three different sets of trajectories:

(1) the ground truth annotations and image ball trajectories from the TrackNet dataset,
(2) the predictions and image ball trajectories from our pipeline, where we utilize WASB to find the ball and our HBNet model to detect the shots (HBNet+WASB), and
(3) the synthetically created trajectories.

Table 1 shows the results of SynthNet on the ground truth, HBNet+WASB, and synthetic test data. The ground truth and HBNet+WASB are only trajectories from the test set, games 1 and 8, and the synthetic trajectories only have trajectories with camera parameters from these two games as well. The results show that SynthNet consistently performs best over all metrics on the synthetic trajectories. This is expected, as these trajectories can be perfectly estimated using our 3D projection method. Using the synthetic data yields a reprojection error of 16.1 pixels, which is more than twice as for the HBNet + WASB and ground truth trajectories. Likewise, we see that the model has half the reconstruction error on the synthetic data compared to both HBNet+WASB and ground truth trajectories.

SynthNet performs equally well on the ground truth trajectories and the HBNet+WASB trajectories in terms of reprojection error, with 41 pixels versus 41.31 pixels, while the landing error differs significantly by a meter. Interestingly, the model has a much higher accuracy, 8%, and F1-score, 0.9, on HBNet+WASB trajectories than on the ground truth trajectories. Additionally, we see a relatively large difference between the mean and median of RE and LE on ground truth and HBNet+WASB trajectories indicating that there are outliers with high RE and LE. The differences between mean and median values on the synthetic data are smaller, showing fewer outliers. Lastly, we can evaluate our model using recreation error on the synthetic data, where we get a mean error of 1.39 meters.

4.3 Effect of Knowing Camera Parameters

To examine the effect of the model having encountered the camera parameters doing training, we test the model on HBNet+WASB trajectories from all the games, i.e. both the test and train sets. Table 2 shows the results. The model is trained using camera parameters from all other games than game 1 and 8, but we don't see that it achieves the worst results on these. On game 1 it is among the best results in all the metrics, while on game 8 it is among the worst results.

Game 4 yields the worst performance based on our proposed metrics. It is noteworthy that game 4 was recorded from a much lower angle than the rest of the games. Based on those observations we conclude that it does not matter as much if the model has encountered the exact camera parameters or angle before if they are not significantly different from previously encountered ones.

4.4 Evaluating 3D Trajectories

Visually inspection of our 3D trajectories (HBNet+WASB) reveals a couple of general tendencies. Firstly, the model seems to have difficulties determining the proper 3D start position of the shot, when the trajectory starts from the side of the court furthest away from the camera. Table 3 shows the absolute error, in meters of the initial position and meters pr. second of the initial velocity, on the synthetic trajectories.

The model is good at determining the correct x-coordinate but has difficulties with especially the y-coordinate of the start position, where it is, on average, 1.6 meters away from the true y-coordinate. We observe the same tendencies on the initial velocity, where it mostly struggles with velocity in the y-direction (along the field). The model is generally better at estimating the initial position than the initial velocity.

Figure 6 shows the image coordinates of the ball found using WASB (red) and the predicted 3D trajectory and its reprojection (green). It shows a reprojection that is somewhat close to the real image trajectory, starting around the same pixels, and landing slightly away from the true position. However, the real 3D shot starts from around the backline, while the predicted shot starts closer to the net. This problem arises, as the perspective makes it more difficult to distinguish and determine distances when it is further away from the camera. Furthermore, when finding the camera parameters, the only reference points that are not on the ground plane are the poles in the middle of the court. This lack of elevated 3D reference points from the rest of the court could make it more difficult to project elevated 3D points onto the image accurately.

Secondly, SynthNet has difficulties with outlier shots such as high-velocity shots, shots from weird angles, shots that have a lot of spin, shots that last only a couple of frames, etc. Figure 7 shows a serve from game 5, with a high velocity, that flies at a strange angle. If we only see the trajectory, it could look like it is flying across instead of along the court, starting close to the net and ending near it as well. As it is a serve, the ball starts high up into the air, which again can be difficult to determine from image coordinates due to

Table 1: Model performance on the three different test sets of trajectories. $\overline{RE}$ denotes the mean RE, $\widetilde{RE}$ the median RE, $\overline{LE}$ the mean LE, and $\widetilde{LE}$ the median LE.

	Image			World			
Image Trajectories	$\overline{RE}$	$\widetilde{RE}$	T.acc	T.F1	$\overline{LE}$	$\widetilde{LE}$	RecE
Ground Truth	41.01 px	29.12 px	46.08%	0.307	2.73 m	2.17 m	-
HBNet + WASB	41.31 px	31.82 px	53.85%	0.398	3.78 m	2.4 m	-
Synthetic	**16.1 px**	**14.65** px	**75.1%**	**0.751**	**1.35 m**	**1.11 m**	1.39 m

Table 2: Results of SynthNet on HBNet+Trajectories trajectories for each game.

	Image	World		
	RE	T.acc	T.F1	LE
game1	35.29 px	64.15%	0.488	3.19 m
game2	35.27 px	56.45%	0.402	**2.25 m**
game3	54.78 px	54.54%	0.294	3.59 m
game4	36.72 px	28.07%	0.187	7.07 m
game5	38.26 px	45.45%	0.202	3.30 m
game6	43.43 px	**66.67%**	0.388	3.40 m
game7	47.76 px	54.00%	0.3540	2.91 m
game8	47.56 px	43.14%	0.320	4.40 m
game9	**34.17 px**	59.52%	0.396	2.61 m
game10	39.56 px	61.54%	**0.5523**	3.02 m
average	41.28 px	53.35%	0.358	3.58 m

Table 3: Average absolute differences and standard deviations on initial conditions on synthetic test data.

IC	Direction	Absolute Error
Position (x_0)	x	0.25 ± 0.23 m
	y	1.63 ± 1.25 m
	z	0.40 ± 0.29 m
Velocity (v_0)	x	0.64 ± 0.53 m/s
	y	3.11 ± 2.67 m/s
	z	0.58 ± 0.41 m/s

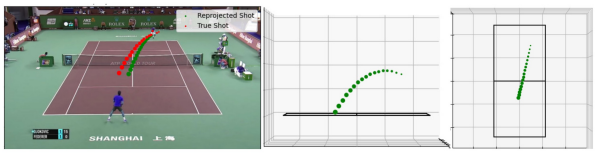


Figure 6: WASB image trajectory and reprojected predicted 3D shot. Left: WASB image trajectory (Red) and reprojected predicted 3D shot(green). Middle: Predicted 3D shot seen from the side. Right: Predicted 3D shot seen from above.

the perspective. As a result, the predicted trajectory starts too close

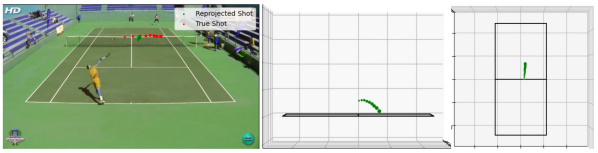


Figure 7: WASB image trajectory and reprojected predicted 3D shot. Left: WASB image trajectory (Red) and reprojected predicted 3D shot(Green). Middle: Predicted 3D shot seen from the side. Right: Predicted 3D shot seen from above.

to the net, does not have the correct trajectory, and does not land close to the actual position. In such cases, the model appears to find the least worst trajectory, which is a short trajectory in the middle of the court. This is a general tendency with outlier shots.

Thirdly, SynthNet also tends to create initial conditions such that the trajectories end before they hit the ground. This is of course a problem, as we calculate our proposed metrics under the assumption that the trajectory ends on the ground.

Furthermore, we observe a tendency to create trajectories that are directed more toward the vertical middle line of the court than the actual shot is. This observation is supported by the F1-scores of each of the 12 defined tiles. Figure 8 show SynthNet achieves the best results on the tiles along the vertical middle line. Additionally, we see that there is a slightly better score at the side of the court furthest away from the camera, which might be due to the camera angle, where the ball's position on the court closer to the camera is more difficult to estimate. Interestingly, on tile 7, the model achieves an F1-score of 0.

4.5 Ablation Study

We conducted an ablation study to explore the effect of using the court corners as input along with the image trajectory. The results presented in Table 4 show a clear performance increase in all metrics when using the court corners. We observe that knowing the court size (in image coordinates) yields more accurate trajectories. The court corners are static in the world coordinates. Thus, the changing image position of the corners provides the model with information about camera position and perspective. Additionally, we hypothesize this is because, with the court corners, the model knows the court's limits and, therefore, finds it easier to make shots within these boundaries. We note that the video material is from professional tennis players. These players rarely shoot the ball far

Table 4: Ablation study of input features on all HBNet+WASB trajectories.

Input	Image RE	T.acc	World T.F1	LE
Image Trajectory	48.25 px	41.35%	0.289	4.13 m
Image Trajectory + court corners	**41.31 px**	**53.85%**	**0.398**	**3.78 m**

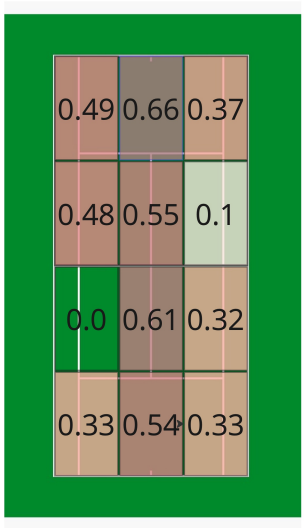


Figure 8: Tile F1-scores of each of the 12 defined tiles on HBNet+WASB trajectories.

outside the court, and the synthetically created trajectories always land within the court. Therefore, it might be valuable for the model to have additional information on where the trajectory should be within the court.

4.6 SynthNet vs. Baseline

We compare our method of estimating initial conditions with a simplified version of the previously used approach. The problem is formulated as an optimization task minimizing only the reprojection error, which we solve by Powell's method [14]. Small experiments using additional loss terms like start/landing loss were made but yielded no definitive results, thus we chose to use this simplified version. Table 5 shows the results for the optimizer and SynthNet on HBNet+WASB trajectories. On one hand, we see the optimizer generally yields a much lower reprojection error. This is expected based on the design which employs the reprojection error as its loss function and SynthNet does not. On the other hand, SynthNet generally results in better tile accuracy, tile F1-score, and landing error, indicating that it performs better than the previous optimizer. Manual inspection of the predicted trajectories supports the conclusion that SynthNet generally outperforms the simplified Baseline method. Additional practical benefits of SynthNet over the reference method are that our proposed approach is more suitable for real-life scenarios because it does not need the camera parameters of the trajectory which the optimizer relies on. This implies that SynthNet does not require estimates of the camera parameters for the video. It instead extracts the aforementioned features. Additionally, since SynthNet is already trained, running inference is much faster than with the optimizer as it needs to search for the optimal set of initial conditions for every shot individually.

4.7 Limitations

While the unique approach of SynthNet demonstrates encouraging results for monocular 3D reconstruction in tennis, several limitations and challenges persist.

Considering that some players can shoot the ball with a velocity of up to 70 m/s, the ball can travel 2.3 meters in one frame. This implies that the provided frame rate of 30 frames per second (FPS) introduces limitations. When the ball moves faster than the FPS can capture, the bounces and hits are likely not exactly in the annotated frames but somewhere between two frames. In this scenario, the landing error is misleading because

we assume that the ball hits the ground in the annotated bounce. Therefore, the error could be decreased with a higher frame rate.

Another limitation is the metrics to evaluate the models. The only metric that can reliably describe the performance of the 3D-constructed trajectory is the reconstruction error, which we do not have for the real data. Since the reprojection error does not capture important aspects, we introduced new metrics to gauge the performance of our method more accurately. However, these additional metrics only examine the trajectory ending, not its starting position.

Lastly, we observed the situation where the ball bounces on the net poses challenges. This situation is annotated as a non-hit but appears very similar to a bounce. Furthermore, 3D trajectory reconstruction fails hard in these cases because 3D trajectory reconstruction does not consider these cases when modeling with Euler's method from initial conditions. This has a fatal effect on performance and yields unusually high reprojection errors which we choose to filter out in the worst cases.

While bouncing on the net is an uncommon occurrence in real-world scenarios, a model tailored to this domain should be able to handle those edge-cases.

4.8 Future Works

In this study, we choose to estimate hit-to-bounce trajectories. A possible addition is a method that models the bounce-to-hit trajectories. Another future avenue entails fine-tuning the synthetically trained model on real data. Additionally, adding more data with varying camera angles would likely make it more robust to different camera angles and out-of-distribution shots. Alternatively, creating synthetic camera parameters could have prevented the issues with the initial conditions in the y-direction. Lastly, experimenting with incorporating spin in the reconstructed 3D trajectories by adding 3D additional initial conditions to the model output yielded no conclusive positive results. However, this will be a priority in future approaches, as spin is an essential factor in tennis.

Table 5: Optimizer and SynthNet performance on HBNET + WASB trajectories

	Image			World		
Model	$\overline{RE}$	$\widetilde{RE}$	T.acc	T.F1	$\overline{LE}$	$\widetilde{LE}$
Optimizer	**12.20 px**	**6.55 px**	40.71%	0.371	5.11 m	3.09 m
SynthNet	41.31 px	31.82 px	**53.85%**	**0.398**	**3.78 m**	**2.4 m**

5 CONCLUSION

We proposed an end-to-end pipeline that reconstructs 3D tennis ball trajectories from monocular video.

The first part of the pipeline is a GRU-based model, which uses image trajectory, player positions, and court corners to detect hits and bounces in tennis matches. Using these predictions to define shots and their trajectories, we test a neural network to predict initial conditions for a 3D reconstruction of the shots' image trajectory. The method achieves good results in both reprojection error, in our own proposed metrics, and when visually inspecting the created trajectories. However, it has difficulties determining the exact start positions, especially for the player furthest from the camera, as well as struggles with outlier shots.

We compare our results to a simplified established approach from previous research and conclude that our methods achieve better results. Although our method achieves a worse reprojection error, it performs better on landing error, tile accuracy & tile f1-score, and 3D reconstruction error. Furthermore, SynthNet is more efficient and generalizes well, since it does not rely on camera parameters. Thus, we believe our approach is more reliable for the reconstruction of 3D trajectories.

REFERENCES

[1] Vanyi Chao, Ankhzaya Jamsrandorj, Yin May Oo, Kyung-Ryoul Mun, and Jinwook Kim. 2023. 3D Ball Trajectory Reconstruction of a Ballistic Shot from a Monocular Basketball Video. In *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 1–6. https://doi.org/10.1109/IECON51785.2023.10312079 ISSN: 2577-1647.

[2] Hua-Tsung Chen, Wen-Jiin Tsai, Suh-Yin Lee, and Jen-Yu Yu. 2012. Ball tracking and 3D trajectory approximation with applications to tactics analysis from single-camera volleyball sequences. *Multimedia Tools and Applications* 60, 3 (Oct. 2012), 641–667. https://doi.org/10.1007/s11042-011-0833-y

[3] Dirk Farin, Susanne Krabbe, Peter H. N. De With, and Wolfgang Effelsberg. 2003. Robust camera calibration for sport videos using court models. In *Storage and Retrieval Methods and Applications for Multimedia 2004*, Minerva M. Yeung, Rainer W. Lienhart, and Chung-Sheng Li (Eds.), Vol. 5307. SPIE, San Jose, CA, 80–91. https://doi.org/10.1117/12.526813

[4] Megan Fazio, KS Fisher, and Tori Fujinami. 2018. Tennis ball tracking: 3-D trajectory estimation using smartphone videos. *Department of Electrical Engineering, Stanford University* (2018).

[5] Q Huang, S Cox, F Yan, TE deCampos, D Windridge, J Kittler, and W Christmas. 20110831 - 20110903. Improved Detection of Ball Hit Events in a Tennis Game Using Multimodal Information, In International Conference on Auditory-Visual Speech Processing. *11th International Conference on Auditory-Visual Speech Processing (AVSP)* (20110831 - 20110903).

[6] Qiang Huang, Stephen Cox, Xiangzeng Zhou, and Lei Xie. 2012. Detection of ball hits in a tennis game using audio and visual information. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE, 1–10.

[7] Yu-Chuan Huang, I.-No Liao, Ching-Hsuan Chen, Tsì-Uí İk, and Wen-Chih Peng. 2019. TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications. http://arxiv.org/abs/1907.03698 arXiv:1907.03698 [cs, stat].

[8] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. Ultralytics YOLO. Retrieved may 1, 2024 from https://github.com/ultralytics/ultralytics

[9] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. Ultralytics YOLO. Retrieved may 1, 2024 from https://github.com/ultralytics/ultralytics

[10] S.X. Ju, M.J. Black, and Y. Yacoob. 1996. Cardboard people: a parameterized model of articulated image motion. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*. IEEE Comput. Soc. Press, Killington, VT, USA, 38–44. https://doi.org/10.1109/AFGR.1996.557241

[11] Jacek Komorowski, Grzegorz Kurzejamski, and Grzegorz Sarwas. 2019. DeepBall: Deep Neural-Network Ball Detector. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications. https://doi.org/10.5220/0007348902970304

[12] Paul Liu and Jui-Hsien Wang. 2022. MonoTrack: Shuttle trajectory reconstruction from monocular badminton video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 3512–3521. https://doi.org/10.1109/CVPRW56347.2022.00395

[13] Peng Lu, Tao Jiang, Yining Li, Xiangtai Li, Kai Chen, and Wenming Yang. 2024. RTMO: Towards High-Performance One-Stage Real-Time Multi-Person Pose Estimation. http://arxiv.org/abs/2312.07526 arXiv:2312.07526 [cs].

[14] M. J. D. Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.* 7, 2 (Jan. 1964), 155–162. https://doi.org/10.1093/comjnl/7.2.155

[15] Jinchang Ren, James Orwell, Graeme A. Jones, and Ming Xu. 2009. Tracking the soccer ball using multiple fixed cameras. *Computer Vision and Image Understanding* 113, 5 (2009), 633–642. https://doi.org/10.1016/j.cviu.2008.01.007 Computer Vision Based Analysis in Sport Environments.

[16] Kosolapov Sergey. 2023. *TennisCourtDetector*. https://github.com/yastrebksv/TennisCourtDetector Accessed: March 2024.

[17] Lejun Shen, Qing Liu, Lin Li, and Haipeng Yue. 2016. 3D reconstruction of ball trajectory from a single camera in the ball game. In *Proceedings of the 10th International Symposium on Computer Science in Sports (ISCSS)*, Paul Chung, Andrea Soltoggio, Christian W. Dawson, Qinggang Meng, and Matthew Pain (Eds.). Vol. 392. Springer International Publishing, Cham, 33–39. https://doi.org/10.1007/978-3-319-24560-7_5 Series Title: Advances in Intelligent Systems and Computing.

[18] Maria Skublewska-Paszkowska, Paweł Powroźnik, and Edyta Łukasik. 2020. Learning Three Dimensional Tennis Shots Using Graph Convolutional Networks. *Sensors* 20 (10 2020), 6094. https://doi.org/10.3390/s20216094

[19] Shuhei Tarashima, Muhammad Abdul Haq, Yushan Wang, and Norio Tagawa. 2023. Widely Applicable Strong Baseline for Sports Ball Detection and Tracking. http://arxiv.org/abs/2311.05237 BMVC2023.

[20] Gabriel Van Zandycke and Christophe De Vleeschouwer. 2019. Real-time CNN-based Segmentation Architecture for Ball Detection in a Single View Setup. In *Proceedings Proceedings of the 2nd International Workshop on Multimedia Content Analysis in Sports*. ACM. https://doi.org/10.1145/3347318.3355517

[21] Fei Wang, Lifeng Sun, Bo Yang, and Shiqiang Yang. 2006. Fast Arc Detection Algorithm for Play Field Registration in Soccer Video Mining. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 6. IEEE, 4932–4936. https://doi.org/10.1109/ICSMC.2006.385087 ISSN: 1062-922X.

[22] T. Watanabe, M. Haseyama, and H. Kitajima. 2004. A soccer field tracking method with wire frame model from TV images. In *2004 International Conference on Image Processing, 2004. ICIP '04.*, Vol. 3. IEEE, 1633–1636 Vol. 3. https://doi.org/10.1109/ICIP.2004.1421382 ISSN: 1522-4880.

[23] Qingyu Xiao, Zulfiqar Zaidi, and Matthew Gombolay. 2024. Multi-Camera Asynchronous Ball Localization and Trajectory Prediction with Factor Graphs and Human Poses. http://arxiv.org/abs/2401.17185 arXiv:2401.17185 [cs].

[24] Jia xin Cai and Xin Tang. 2018. RGB Video Based Tennis Action Recognition Using a Deep Historical Long Short-Term Memory. *arXiv: Computer Vision and Pattern Recognition* (2018). https://api.semanticscholar.org/CorpusID:52824593

[25] Fei Yan. 2005. Tennis ball tracking for automatic annotation of broadcast tennis video. *Proceedings of the British Machine Vision Conference* (2005). https://www.semanticscholar.org/paper/Tennis-ball-tracking-for-automatic-annotation-of-Yan/d135b747e99e6a06f5ecac5462b53c1b7bd259e2?p2df