Football Players Tracking from Multiple Cameras using Deep Learning

Youssef George Fouad Saad 19P9824

Anthony Amgad Fayek Selim 19P9880

Kerollos Wageeh Youssef Saleeb 193468

Marc Nagy Nasry Sorial 19P3041

Youssef Ashraf Mounir Showeter 19P4179

Supervisor: **Prof. Dr. Mahmoud I. Khalil**

Sponsored by: **KoraStats**

Ain Shams University, Faculty of Engineering ICHEP

Computer Engineering and Software Systems

Spring 2024

Abstract

In this comprehensive thesis, we present the culmination of our work on developing a sophisticated deep learning and computer vision system tailored for football player tracking during matches. Building on the foundations laid in the first semester, this thesis details the enhancements and finalizations made to the system, focusing on crucial aspects such as path prediction, player re-identification, occlusion resolution, and similarity matching.

The core outcome of our system is the extraction of high-fidelity spatio-temporal trajectories of players, unlocking a myriad of possibilities for statistical analysis. These trajectories serve as the foundation for generating in-depth player statistics, offering valuable insights for sports enthusiasts and analysts alike. Our system not only automates the tracking process but also enhances the user experience through a dynamic and graphically rich interface. While our system excels in autonomous tracking, we acknowledge that certain scenarios may require human intervention to rectify miss-trackings, highlighting the collaborative nature of our approach.

This thesis begins with a detailed discussion on the intricacies of image processing, elaborating on the methods employed. Our systematic approach ensures a robust and accurate player tracking system, capable of adapting to the fast-paced and dynamic nature of football matches.

The experimental phase of the first semester laid the groundwork for a fully functional system. In the second semester, we refined and completed the system, culminating in the development of a dynamic user interface, which stands as the crowning achievement of this project. Through this thesis, we aim to contribute significantly to the intersection of technology and football analytics, revolutionizing player tracking and analysis. Our journey is a testament to the fusion of technological innovation and sports expertise, promising a transformative impact on the landscape of football analytics.

Keywords: Football Analytics, Player Tracking, Computer Vision, Deep Learning.

Acknowledgments

We would like to express our sincere gratitude to all those who have contributed to the completion of this project and thesis.

First and foremost, we extend our appreciation to our supervisor, Prof. Dr. Mahmoud I. Khalil, for his invaluable guidance, support, and encouragement throughout the entire project. His expertise and insights have significantly enriched our understanding and enhanced the quality of this work.

We are grateful to **KoraStats** for their sponsorship and collaboration during this project. Special thanks to Eng. Ahmed Bahgat, the Research and Development Officer, for his technical support and insights that hugely contributed to the success of our work. Additionally, we appreciate the support of Nasr Ali, the HR Manager, for facilitating the collaboration and providing organizational support.

We express our deepest appreciation to the faculty and staff at Ain Shams University, especially the Computer Engineering and Software Systems department, for providing an environment conducive to learning and research.

Finally, we want to thank our families and friends for their unwavering support and encouragement throughout this academic journey. Thank you to everyone who has been a part of this endeavor.

List of Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
NN	Neural Network
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
FCN	Fully Convolutional Network
VAE	Variational Auto Encoder
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
ROI	Region of Interest
YOLO	You Only Look Once
SSN	Semantic Segmentation Network
SDS	Superpixel-based Discriminative Segmentation
SAM	Segment Anything Model
MNC	Multi-task Network Cascades
ReLU	Rectified Linear Unit
KB	Knowledge Base
bbox	Bounding Box

Contents

Abstract i

Acknowledgments ii

List of Abbreviations iii

1 Introduction 1

1.1 Motivation . . . 1
1.2 Problem Statement . . . 2
1.3 Outline . . . 3

2 Literature Review 4

2.1 Object Detection in Football Match Analysis . . . 4
2.1.1 RetinaNet: Capturing Multi-Scale Features on the Football Field . . . 4
2.1.2 Focal Loss for Class Imbalance . . . 5
2.2 Semantic Segmentation in Sports Analytics . . . 6
2.2.1 U-Net . . . 6
2.2.2 DeepLab . . . 6
2.2.3 Meta SAM . . . 7
2.2.4 A Comparative Analysis . . . 8
2.3 Visual Object Tracker: Masking Single Player Across Sequential Frames . . . 8
2.3.1 Mask R-CNN: Precise Segmentation Tracking . . . 8
2.3.2 Online Adaptation for Dynamic Player Appearance . . . 8
2.3.3 Attention Mechanisms for Region Refinement . . . 9
2.4 Path Prediction in Sports Dynamics . . . 9
2.4.1 KB Approaches . . . 9
2.4.2 DL Approaches . . . 10
2.4.3 Challenges and Opportunities . . . 11
2.5 Similarity Matching . . . 11
2.5.1 Person Re-identification with Deep Learning . . . 12
2.5.2 VAE for Molecular Similarity . . . 12
2.5.3 Swin Transformer: Hierarchical Vision Transformer . . . 13

3 Neural Network Architecture **15**
3.1 Object Detection Model . . . 16
3.1.1 Loss Function . . . 16
3.1.2 Model Configuration . . . 16
3.2 Semantic Segmentation Model . . . 17
3.2.1 Fine-Tuning Meta SAM for Football Player Tracking . . . 17
3.2.2 Loss Function . . . 18
3.3 Visual Tracking Model . . . 18
3.3.1 FCN Architecture . . . 18
3.3.2 Initial Training Approach . . . 20
3.3.3 Combined Loss . . . 20
3.4 Path Prediction Model . . . 21
3.4.1 Previous Models . . . 22
3.4.2 Fully Connected MLP Architecture . . . 23
3.4.3 Training and Optimization . . . 23
3.5 Similarity Model . . . 24
3.5.1 Previous Approaches . . . 24
3.5.2 VAE Architecture . . . 24
3.5.3 Swin Transformer Architecture . . . 24
3.5.4 Loss Computation . . . 26

4 Solution Architecture **28**
4.1 System Overview . . . 28
4.2 A Modular Architectural Design . . . 29
4.2.1 Object Detection Module . . . 30
4.2.2 Semantic Segmentation Module . . . 31
4.2.3 Visual Tracker Module: Persistent Player Trajectory Analysis . . . 31
4.2.4 Path Prediction Module: Foreseeing Occlusions, Optimizing Efficiency . . . 33
4.2.5 Similarity Module: Dynamic Player Re-Identification Framework . . . 34
4.3 Initial Preprocessing: . . . 34
4.4 Integration and Data Flow . . . 35
4.4.1 Parallel Visual Tracking: . . . 35
4.4.2 MQTT for Processes Communication . . . 36
4.4.3 Message Queue Database Integration . . . 37
4.5 Post-processing . . . 38
4.5.1 Interpolation of Missing Frames . . . 38
4.5.2 Transformation and Mapping . . . 38
4.6 Statistics Generation and Visualization . . . 43
4.6.1 Data Retrieval API . . . 43
4.6.2 Statistical Calculations . . . 44

4.6.3 Visualization of Statistics . . . 46
4.7 Benefits of Integrated Architecture . . . 47
4.8 Limitations: Acknowledging the Boundaries . . . 49
4.8.1 Challenging Conditions . . . 49
4.8.2 Computational Efficiency . . . 49
4.8.3 Data and Model Limitations . . . 50

5 Software Requirements Specification (SRS) 51
5.1 Scope . . . 51
5.2 Functional Requirements . . . 52
5.3 Use Cases . . . 53
5.4 Non-Functional Requirements . . . 54
5.5 Software and Frameworks . . . 56
5.6 Constraints . . . 56

6 Results and Evaluation 58
6.1 Object Detection Model Refinement . . . 58
6.1.1 Enhancements in Annotation Approach . . . 58
6.1.2 Results Samples . . . 59
6.2 Visual Object Tracker . . . 60
6.2.1 Tracking Approach . . . 61
6.2.2 Results Samples . . . 61
6.2.3 Evaluation Metrics . . . 64
6.2.4 Evaluation Results . . . 64
6.2.5 Discussion . . . 65
6.3 Evaluating the Path Prediction Module . . . 65
6.3.1 Evaluation Metrics . . . 65
6.3.2 Evaluation Results . . . 66
6.3.3 Cosine Similarity Across Frames . . . 66
6.3.4 Visual Comparisons . . . 67
6.3.5 Discussion . . . 67
6.4 Evaluating the Similarity Module . . . 69
6.4.1 Precision, Recall, and F1-Score . . . 69
6.4.2 Confusion Matrix . . . 70
6.4.3 Overall Performance . . . 70

Conclusion 72

List of Figures

2.1 RetinaNet Architecture 5
2.2 RetinaNet Steps 5
2.3 DeepLab V3+ for Semantic Segmentation 7
2.4 Meta SAM 7
2.5 Social Force Model for Pedestrian Dynamics 10
2.6 Spatio-Temporal Graph Transformer Network 10
2.7 DeepCRF for Person Re-identification 12
2.8 VAE-SIM Molecular Similarity 13
2.9 Swin Transformer Architecture 13

3.1 Meta SAM Model Architecture 17
3.2 FCN Model Architecture 19
3.3 FCN Training Input 20
3.4 FCN Background Sample Training Input 21
3.5 UNET Architecture 22
3.6 MLP Architecture for Path Prediction 23
3.7 MLP Train Samples 25

4.1 Modular System Components 29
4.2 Data Flow & System Architecture 35
4.3 Chessboard Pattern for Undistortion Approximation 39
4.4 Undistortion Result of an Image 39
4.5 Transformation Points on target 2D plane 40
4.6 Transformation Points on camera pictures 40
4.7 Transformed Camera Views 41
4.8 Sample Transformed Player Position 42
4.9 Interactive Speed Chart 45
4.10 Heatmap View 46
4.11 Bounding Boxes View 46
4.12 Screenshot from Match Tracked Players Camera View 47
4.13 Choose Player to View His/Her Data 47
4.14 2D Plane Player Tracking View and Speed 48

4.15 Player Summary Stats 48
4.16 Multiple Players 2D View 48

6.1 Failed Sample of Object Detection 60
6.2 Success Sample of Object Detection 60
6.3 Sample 1 of Visual Player Tracking 62
6.4 Sample 1 of Visual Player Tracking, Full Frame 62
6.5 Sample 2 of Visual Player Tracking 63
6.6 Sample 2 of Visual Player Tracking, Full Frame 63
6.7 Cosine Similarity Across Frames: Evaluates the angular closeness of predicted positions to actual positions over 64 frames. 67
6.8 Path Comparison 1: Input path, predicted path, and actual path. 68
6.9 Path Comparison 2: Input path, predicted path, and actual path. 68
6.10 Path Comparison 3: Input path, predicted path, and actual path. 69
6.11 Normalized Confusion Matrix of the Similarity Model 71

Chapter 1

Introduction

The realm of football analytics has witnessed significant advancements with the advent of deep learning and computer vision technologies. This thesis presents the culmination of our extensive research and development efforts in creating a sophisticated system for tracking football players during matches. Building on the foundation laid in the first semester, we have refined, enhanced, and completed our project, focusing on delivering a robust and accurate player tracking system.

Our project aims to automate the process of football player tracking, leveraging advanced image processing techniques and deep learning models. The primary objective is to extract high-fidelity spatio-temporal trajectories of players, which can be utilized for detailed statistical analysis. These trajectories provide invaluable insights into player performance, tactics, and strategies, revolutionizing the way football analytics is conducted.

In the first semester, we concentrated on the initial development and experimentation phases. We explored various methodologies for object detection, segmentation, and path prediction, implementing and experimenting different models.

The second semester focused on refining these components, incorporating significant changes across different modules to enhance the system's performance and integration in addition to building a webapp that allows the users to view and interact with the tracked players data in addition to the in depth statistics calculated from the player tracks.

1.1 Motivation

Football, often hailed as the beautiful game, captivates millions with its dynamic play and strategic intricacies. Yet, the manual methods employed in tracking and analyzing player movements within the sport are labor-intensive, error-prone, and fail to capture the essence of the game comprehensively.

Our motivation arises from the convergence of two passions: our love for football and our commitment to pushing the boundaries of technological innovation. As fervent football enthusiasts, we recognize the immense potential for technology to enhance the understanding and analysis of the sport we hold dear.

The shortcomings of existing manual tracking methods fuel our drive to develop a sophisticated deep learning and computer vision system. By seamlessly integrating deep learning advancements with other computer vision techniques, our goal is to provide accurate positional data for every player, at every moment, across multiple camera feeds. This system aims not only to automate and enhance player tracking but also to redefine how we perceive and analyze the beautiful game.

Guided by our tech-savvy and football-loving ethos, our thesis embarks on a journey where deep learning advancements take center stage. We envision a system comprised of five main subsystems, each harnessing the power of deep learning models seamlessly integrated with other computer vision techniques. These subsystems represent the convergence of our technical prowess and football acumen, aiming to redefine player tracking and analysis in football matches.

In essence, our motivation is rooted in the belief that the marriage of technology and football can usher in a new era of sports analytics. Through our dedication to this project, we seek to contribute to the evolution of player tracking, enrich the understanding of in-game dynamics, and provide a toolset that empowers coaches, analysts, and fans alike.

1.2 Problem Statement

In the realm of football analytics, the traditional methods of manual player tracking and analysis present several challenges that hinder the depth and accuracy of performance assessment. These challenges include:

1. **Labor-Intensive Processes:** Current practices rely heavily on manual efforts for tracking player movements and analyzing game dynamics. This approach is time-consuming, requires significant human resources, and is prone to errors.

2. **Limited Granularity:** Manual tracking often provides limited granularity in capturing the intricate details of player movements, especially in fast-paced and dynamic situations during a football match.

3. **Occlusion Issues:** Occlusions, where players obstruct each other's view, pose a substantial challenge to accurate tracking. Existing methods struggle to handle such scenarios effectively.

These challenges underscore the need for an automated and technologically advanced solution that can overcome the limitations of manual tracking. As we delve into the development of our project, which comprises distinct subsystems integrating deep learning and computer vision techniques, we aim to address these issues comprehensively. Our objective is to provide a more efficient and accurate approach to player tracking and analysis in football matches, paving the way for a new era of football analytics.

Our project aims to tackle these challenges head-on by developing a computer vision system that seamlessly tracks players across multiple camera feeds, resolves occlusions, and delivers accurate positional data. By doing so, we aspire to usher in a new era of football analytics that is not only more precise but also more accessible and less reliant on manual labor.

1.3 Outline

This thesis is structured into the following chapters, each addressing specific aspects of the project research, implementation, integration, and development:

1. **Chapter Two: Literature Review**

   In this chapter, we delve into existing literature and research relevant to football analytics, player tracking, deep learning, and computer vision. We explore the advancements, methodologies, and challenges documented in the academic landscape.

2. **Chapter Three: NN Architecture Review**

   Building upon the literature review, we provide an in-depth analysis of neural network architectures pertinent to our project. This chapter lays the groundwork for the DL advancements that will be integrated into our deep learning computer vision system.

3. **Chapter Four: Solution Architecture**

   Here, we detail the architecture of our proposed computer vision system. Each subsystem, powered by deep learning models and computer vision techniques, is introduced and explained. This chapter serves as a blueprint for the technical implementation.

4. **Chapter Five: Software Requirements Specifications (SRS)**

   We outline the software requirements for the development of our system. This includes a comprehensive overview of the functional and non-functional requirements, as well as detailed use cases.

5. **Chapter Six: Results and Discussion**

   The final results obtained from the project implementation are presented and analyzed in this chapter. We evaluate the system's performance against predefined metrics and assess its effectiveness and implications of our findings in real-world football match scenarios.

6. **Chapter Seven: Conclusion**

   Finally, we draw conclusions from our research and development efforts. This chapter summarizes the key findings, discusses the implications of the results, and suggests directions for future research work.

By following this structured outline, we aim to provide a comprehensive exploration of our project, from the theoretical foundations to the practical implementation and evaluation.

Chapter 2

Literature Review

In this chapter, we delve into the existing literature and research pertinent to the five subsystems of our project: object detection, visual tracking, segmentation, similarity matching, and path prediction. The review encompasses both foundational works and recent advancements in computer vision, deep learning, and sports analytics.

2.1 Object Detection in Football Match Analysis

Object detection plays a pivotal role in understanding and analyzing dynamic scenarios, and in the context of a football match, the combination of RetinaNet and Focal Loss, which was inspired by papers like "Focal Loss for Dense Object Detection" by Tsung-Yi Lin[18], proves to be particularly effective. The exploration dives into how these techniques, implemented for football match analysis, contribute to the accurate identification and tracking of the players and the ball. Notable works also include the R-CNN family of methods, such as Faster R-CNN [22], which introduced region proposal networks to improve detection speed. YOLO (You Only Look Once) [21] is another influential approach known for its real-time object detection capabilities. Research in sports analytics has explored object detection in various sports contexts, but few focus specifically on football. A notable exception is the work by Lea et al. [17], which proposes a temporal model for player detection in soccer.

2.1.1 RetinaNet: Capturing Multi-Scale Features on the Football Field

RetinaNet, with its single-stage architecture[18] and Feature Pyramid Network (FPN), offers a robust solution for detecting objects at different scales. In the context of a football match, where players and the ball vary in size and move dynamically across the field, RetinaNet's ability to capture multi-scale features becomes invaluable. The anchor mechanism helps ensure accurate localization, even in situations where players may be partially occluded, or the ball is in motion.

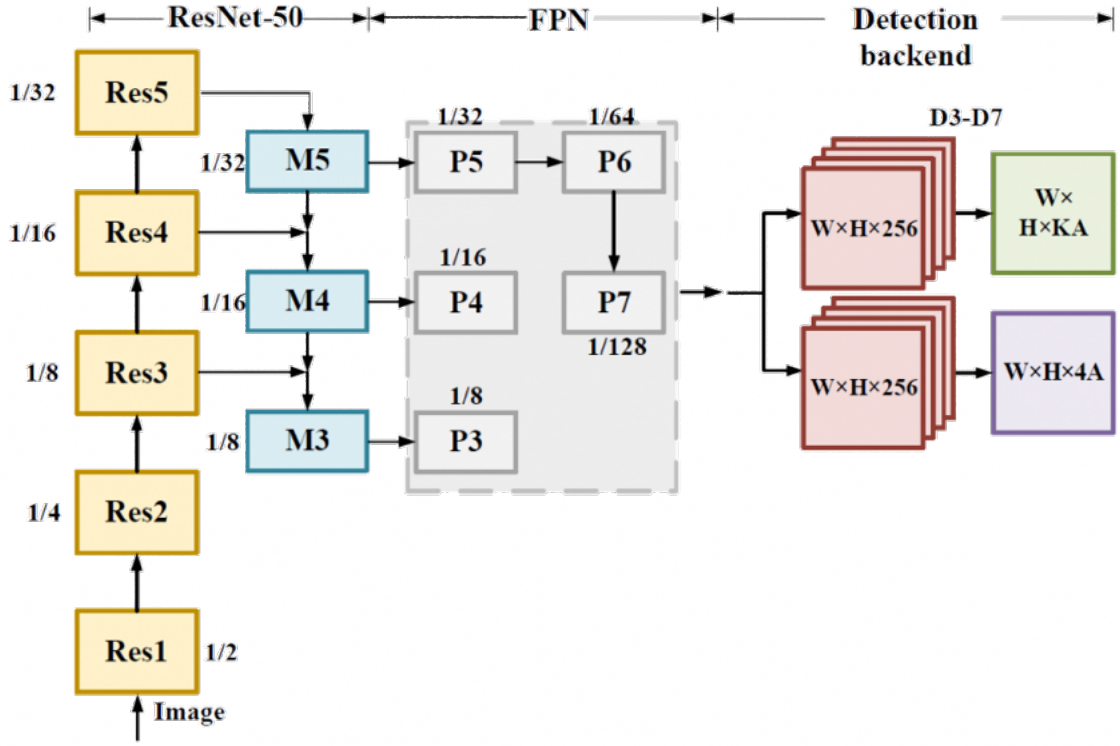

Figure 2.1: RetinaNet Architecture

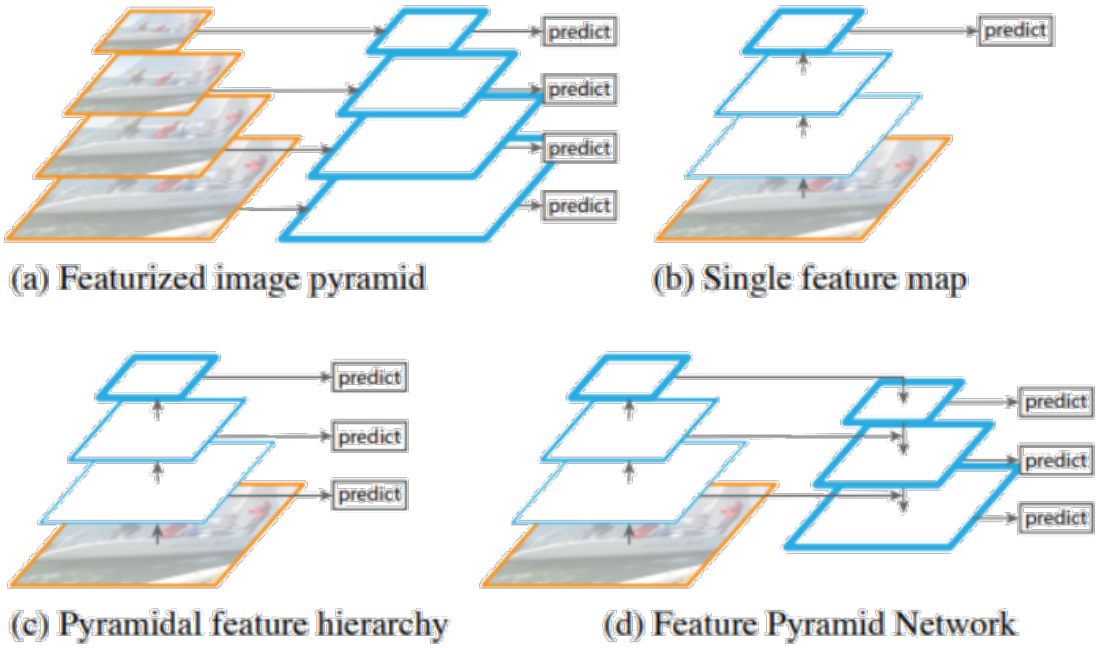

Figure 2.2: RetinaNet Steps

2.1.2 Focal Loss for Class Imbalance

The imbalanced distribution of classes in football match footage, characterized by a multitude of frames containing background information compared to instances of player or ball presence,

poses a significant challenge. Focal Loss, introduced in conjunction with RetinaNet, proves to be instrumental in addressing this class imbalance. By assigning higher weights to challenging instances, such as the ball or players in crowded regions, Focal Loss enhances the model's ability to concentrate on critical elements, thereby improving the accuracy of object detection [18].

2.2 Semantic Segmentation in Sports Analytics

Semantic segmentation plays a pivotal role in the field of sports analytics, especially for applications like player tracking, field marking detection, and tactical analysis. By assigning a label to every pixel in an image, semantic segmentation enables the precise delineation of different objects and regions of interest within the sports field. This section explores some of the state-of-the-art models used for semantic segmentation, focusing on U-Net, DeepLab, and Meta SAM, and their applications in sports analytics.

2.2.1 U-Net

U-Net, introduced by Ronneberger et al. (2015), is a convolutional neural network designed for biomedical image segmentation but has since found extensive applications in various fields, including sports analytics. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. This design allows U-Net to perform well with limited training data, making it suitable for sports scenarios where annotated data may be scarce.

U-Net's effectiveness in sports analytics comes from its ability to accurately segment players and field markings, which are crucial for tracking and tactical analysis. For instance, in a football match, U-Net can help identify and differentiate players from the background, even in complex scenes with occlusions and varying lighting conditions.

2.2.2 DeepLab

DeepLab,[5] represents a series of models that leverage atrous (dilated) convolutions to capture multi-scale contextual information without increasing the computational burden. The most recent versions, such as DeepLabv3 and DeepLabv3+ [4], incorporate atrous spatial pyramid pooling (ASPP) and encoder-decoder structures to enhance segmentation accuracy and robustness.

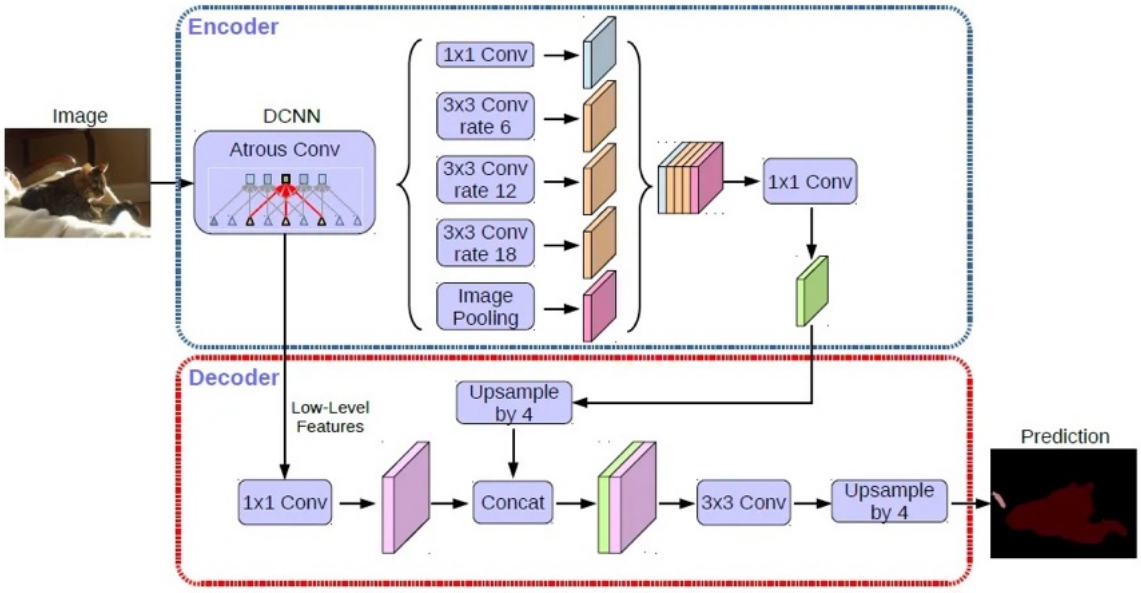

Figure 2.3: DeepLab V3+ for Semantic Segmentation

In sports analytics, DeepLab models are particularly effective due to their ability to handle high-resolution images and intricate details. This capability is essential for accurately segmenting players and other objects in a sports field, allowing for detailed analysis of player movements, interactions, and formations. DeepLab's performance in various segmentation benchmarks has established it as a reliable choice for sports-related applications.

2.2.3 Meta SAM

Meta SAM, a more recent innovation, leverages self-attention mechanisms to enhance segmentation performance. By fine-tuning Meta's SAM (Segment Anything Model) for specific applications, this model achieves high accuracy in identifying and segmenting objects within the sports field. The self-attention mechanism allows Meta SAM to focus on relevant features and relationships within the image, providing superior performance in complex scenes [15].

In the context of sports analytics, Meta SAM's ability to adapt to specific tasks through fine-tuning makes it highly versatile. For example, fine-tuning Meta SAM for football player segmentation can lead to more precise and consistent identification of players, even in challenging conditions such as overlapping players or rapid movements. The model's adaptability and accuracy make it a valuable tool for advanced sports analytics.

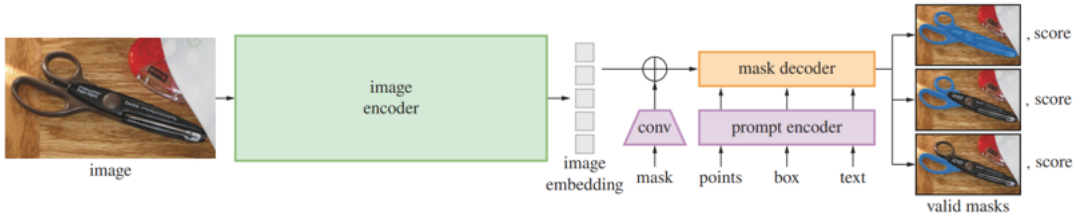

Figure 2.4: Meta SAM

2.2.4 A Comparative Analysis

The integration of semantic segmentation models like U-Net, DeepLab, and Meta SAM in sports analytics has revolutionized the way data is processed and analyzed. These models enable the extraction of detailed and accurate information about players, field markings, and other relevant objects, facilitating advanced tactical and performance analysis.

For instance, in football, these segmentation models can be used to create heatmaps of player movements, analyze passing patterns, and evaluate team formations. The insights derived from such analyses can help coaches and analysts make informed decisions, enhance player performance, and develop effective strategies.

Moreover, the continuous improvement of these models, driven by advancements in deep learning and computer vision, promises even greater accuracy and efficiency in sports analytics. As these technologies evolve, their application in sports will undoubtedly expand, providing deeper insights and contributing to the overall advancement of sports science and performance optimization.

2.3 Visual Object Tracker: Masking Single Player Across Sequential Frames

In sports analytics, Visual object tracking is essential for maintaining the identity of players as they move across the pitch. The precise masking of a single player across a sequence of frames in dynamic scenarios, such as football matches, is crucial for in-depth analysis.
The literature review explores a variety of methodologies, such as the correlation filters [3], Kalman filters [14], and more recent deep learning-based trackers like GOTURN [11] in addition to the work by Zhang et al. [35], providing insights into tracking and masking a moving player.

2.3.1 Mask R-CNN: Precise Segmentation Tracking

Mask R-CNN [9] stands out as a powerful methodology for instance segmentation, providing accurate masks for individual players in sports footage. Its simultaneous prediction of object bounding boxes and segmentation masks makes it particularly effective in isolating and tracking players in the dynamic context of football matches.

2.3.2 Online Adaptation for Dynamic Player Appearance

Jianglong Ye et al. [30] propose online adaptation techniques to address challenges posed by dynamic changes in player appearance. By continuously updating the initial mask based on evolving player characteristics, these adaptive approaches ensure the persistence of accurate player masks across frames, even when facing changing appearances.

2.3.3 Attention Mechanisms for Region Refinement

Vaswani et al. [27] inspire the use of attention mechanisms for refining the masked region around the player. These techniques focus on relevant regions within the initial mask, adapting it to the player's movement. This attention-driven refinement contributes to the precision of the player mask in dynamic sports scenarios.

2.4 Path Prediction in Sports Dynamics

Path prediction aids in anticipating the future movements of players or the ball, contributing to occlusion resolution. Recurrent Neural Networks (RNNs) [24] and Long Short-Term Memory (LSTM) networks [12] are prevalent in trajectory prediction. It plays a crucial role in understanding the movement of entities, with applications ranging from crowd management to sports analytics. We explore the methodologies of knowledge-based (KB) and deep learning (DL) in the application of path prediction in sports scenarios.

In sports analytics, studies like [19] have applied path prediction to player movement analysis. Examining these works provides insights into the challenges and solutions related to path prediction in football scenarios.

2.4.1 KB Approaches

Traditional models, such as those based on social force principles [10] and cellular automata [6], contribute valuable insights into crowd dynamics. These models incorporate parameters for social interactions and environmental factors. While highly interpretable, KB models may face challenges in accommodating diverse pedestrian behaviors.

KB models find applications in sports analytics by applying collision avoidance and spatial anticipation principles. These models consider factors such as player positions, ball dynamics, and strategic elements. Challenges in adapting to dynamic sports scenarios may exist.

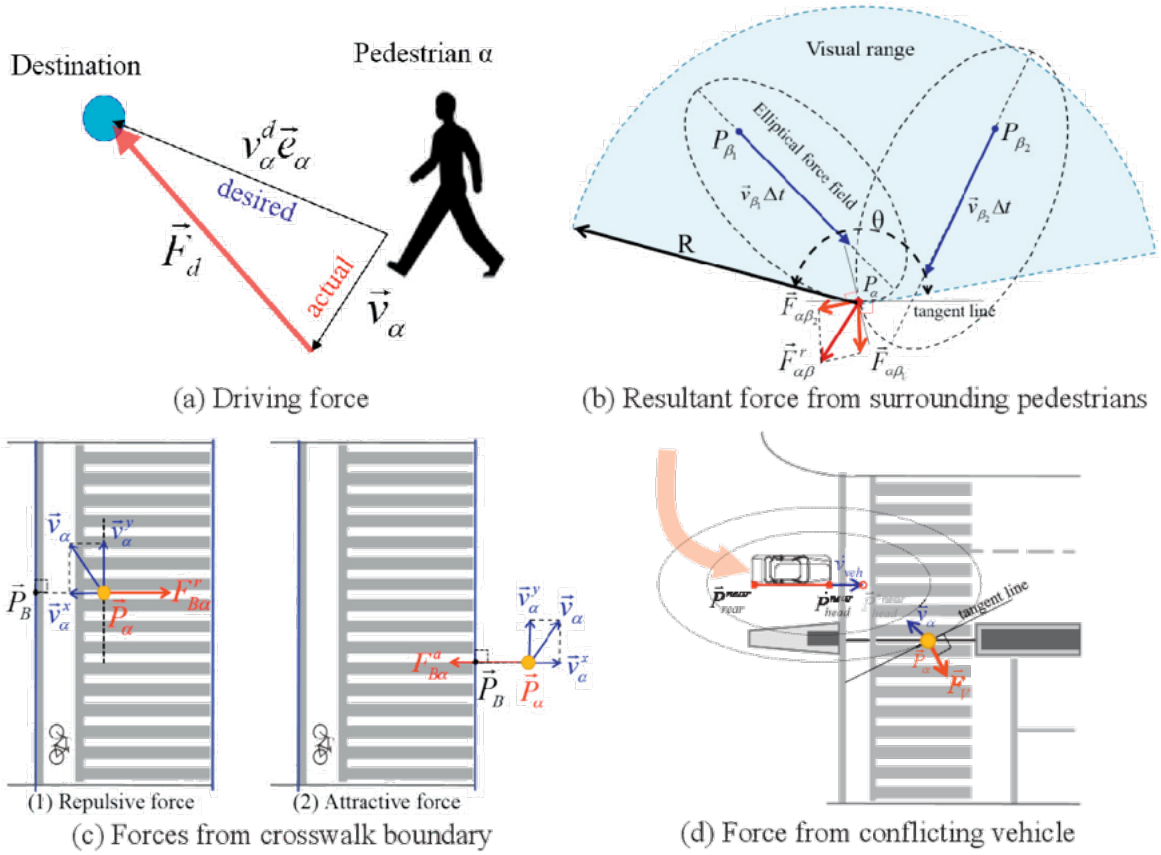

Figure 2.5: Social Force Model for Pedestrian Dynamics

2.4.2 DL Approaches

Deep Learning, specifically leveraging RNNs, LSTMs, and CNNs, has demonstrated remarkable efficacy in the intricate task of predicting pedestrian trajectories [2, 31]. RNNs and LSTMs excel in sequence modeling tasks due to their inherent ability to capture temporal dependencies within sequential data. LSTMs, with purpose-built memory cells, prove highly effective in various sequential prediction domains, such as speech recognition, machine translation, and trajectory forecasting [8, 26, 1, 28].

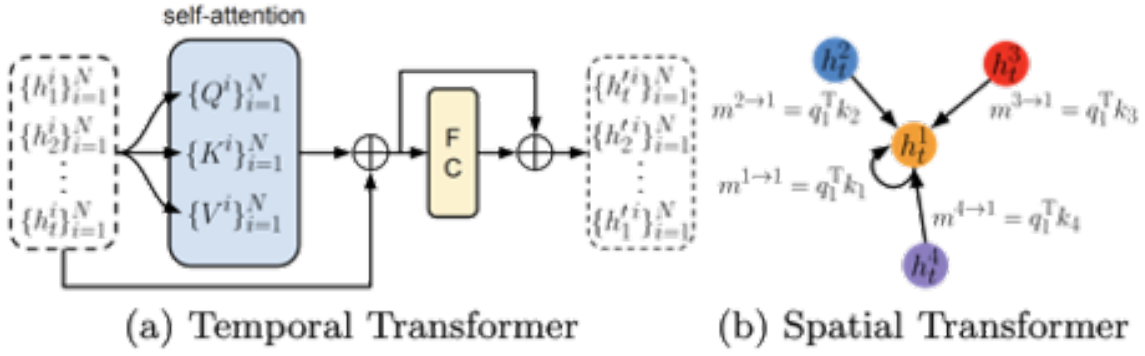

Figure 2.6: Spatio-Temporal Graph Transformer Network

Despite the undeniable success of DL models, concerns linger regarding their interpretability and reproducibility, with the intricate nature of neural network architectures making it challenging to provide transparent insights into decision-making processes [20].

In the context of sports path prediction, DL approaches, particularly those utilizing recurrent and convolutional architectures, showcase significant promise. RNNs play a pivotal role in capturing temporal dependencies in sports trajectories, allowing models to discern intricate player movements, including passes, dribbles, and shots [32, 23]. The adaptability of DL models is particularly beneficial in understanding the nuanced and dynamic nature of sports scenarios. Concurrently, CNNs are adept at extracting spatial features from sports footage, facilitating the comprehension of player positioning on the field and interactions with teammates and opponents. The synergy of RNNs and CNNs in DL models empowers them to not only predict sports trajectories with high accuracy but also comprehend the complex interplay of temporal and spatial dynamics inherent in sports scenarios.

2.4.3 Challenges and Opportunities

The trade-offs between KB and DL approaches involve considerations of interpretability, reproducibility, and adaptability. Challenges, including data requirements, DL interpretability, and the need for domain knowledge in KB models, must be carefully navigated in the context of sports path prediction.
Path prediction in pedestrian dynamics and sports analytics presents a multidimensional challenge. Integrating KB and DL approaches provides a comprehensive perspective, leveraging their respective strengths for accurate and adaptable path predictions across diverse domains.

2.5 Similarity Matching

Similarity matching is critical for associating players across different poses, pictures, and camera viewpoints. The ability to maintain accurate player identification under occlusion scenarios is crucial for effective player tracking systems. This underscores the critical role of the similarity module, which facilitates re-identification of players after occlusion. Our proposed module utilizes Variational Autoencoders (VAEs) to encode each player into a unique latent vector before occlusion. Subsequently, we measure the dissimilarity between these latent vectors and those extracted from post-occlusion frames to establish matches.

To ensure the efficacy of this approach, we have conducted a comprehensive literature review encompassing two key areas:

- **Person Re-identification with Deep Learning:** We delve into existing deep learning-based re-identification approaches to glean insights into how they address occlusion challenges.

- **VAE for Molecular Similarity:** We explore the application of VAEs in characterizing molecular similarity, seeking transferable knowledge and potential adaptations for player re-identification under occlusion.

2.5.1 Person Re-identification with Deep Learning

Jun Xiang, Ziyuan Huang, Xiaoping Jiang, and Jianhua Hou [29] propose a novel deep learning conditional random field (Deep-CRF) graph for person re-identification, treating the task as a CRF node labeling problem. Their approach considers group-wise similarities within a batch of images and exhibits inference consistency in both training and testing stages. The Deep-CRF outperforms existing methods on three large-scale person re-ID datasets, surpassing the previous deep CRF framework by 8% in Rank1 accuracy on the CUHK03 dataset.

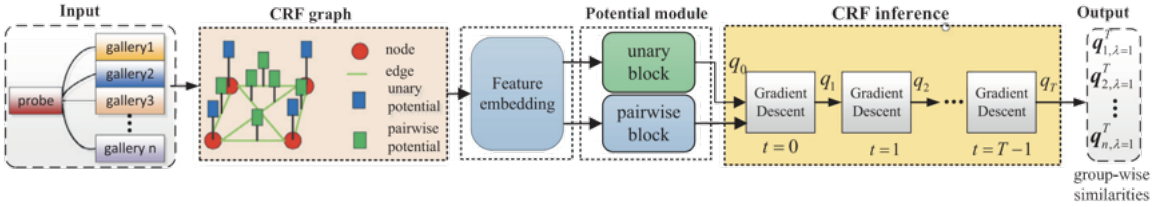

Figure 2.7: DeepCRF for Person Re-identification

Haque Ishfaq and Ruishan Liu [13] propose a novel structure named Triplet based Variational Autoencoder (TVAE) to enhance latent embedding learning by integrating deep metric learning with variational autoencoder (VAE). The TVAE method addresses the limitations of traditional VAEs in handling label or feature information, utilizing a triplet loss on the mean vectors of VAE in conjunction with reconstruction loss. Demonstrated on the MNIST dataset, TVAE achieves a high triplet accuracy of approximately 95.60%, showcasing its effectiveness, and the authors further validate the method on the Zappos50k shoe dataset, highlighting its efficacy in real-world applications.

2.5.2 VAE for Molecular Similarity

Soumitra Samanta, Steve O'Hagan, Neil Swainston, Timothy J. Roberts, and Douglas B. Kell [25] present VAE-Sim, a molecular similarity measure based on a variational autoencoder (VAE). The VAE is trained on over six million druglike molecules and natural products, utilizing a "bowtie"-shaped artificial neural network with a bottleneck layer for encoding and decoding. VAE-Sim offers a rapid and easily calculable metric for molecular similarity, providing a valuable contribution to cheminformatics.

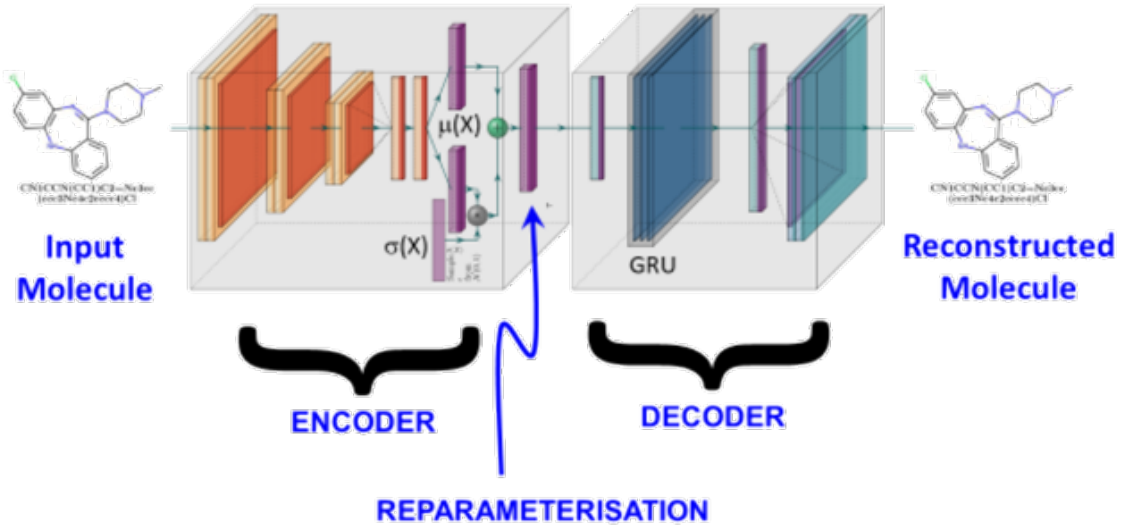

Figure 2.8: VAE-SIM Molecular Similarity

2.5.3 Swin Transformer: Hierarchical Vision Transformer

The Swin Transformer, introduced by Liu et al. [33], represents a significant advancement in vision transformers. Unlike traditional transformers that suffer from quadratic complexity with image size, the Swin Transformer employs a hierarchical architecture with shifted windows, allowing it to scale efficiently to high-resolution images. This architecture captures both local and global contextual information, making it particularly effective for dense prediction tasks such as player re-identification.

Building on the success of the Swin Transformer, a specialized version was developed specifically for soccer player re-identification [34]. This enhanced model incorporates modifications that address the unique challenges of the sports environment, such as varying lighting conditions, occlusions, and the need for real-time processing.

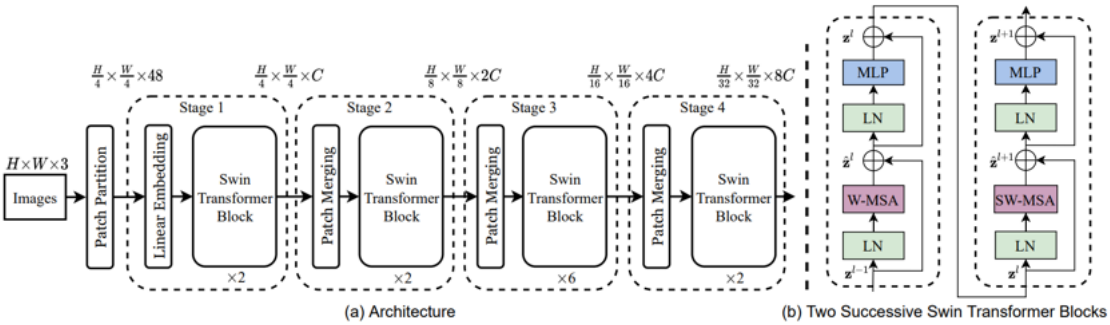

Figure 2.9: Swin Transformer Architecture

The enhanced Swin Transformer employs additional techniques such as domain-specific data augmentation and fine-tuning on large-scale soccer datasets. These enhancements improve the model's ability to generalize across different matches and scenarios, ensuring consistent

performance. The model's superior accuracy in re-identifying players has been demonstrated in several studies, making it a valuable tool for sports analysts and coaches.

Chapter 3

Neural Network Architecture

In this chapter, we provide a brief presentation of built neural networks is provided in the context of image and video processing. Inspired by the human brain, NNs are able to learn and generalize from experience. One major application area of a certain type of NNs is the convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Both are advanced information processing paradigms inspired by the human brain, adapted specifically for tasks related to image and video analysis, such as image recognition, classification, segmentation, and temporal pattern recognition.

CNNs are specialized neural networks designed for processing grid-like data, such as images. Inspired by the human visual system, CNNs excel at capturing hierarchical features and patterns in images. Their ability to automatically learn and generalize from examples makes them particularly effective in tasks like image classification and object detection. CNNs are adept at discerning intricate relationships in data, even when the underlying structures are complex or challenging to articulate. Moreover, they demonstrate a remarkable capacity for extrapolating predictions to unseen aspects of the input data, enabling them to forecast future behaviors in image datasets.

RNNs, on the other hand, are tailored for handling sequential data, making them well-suited for video processing tasks. Inspired by the human brain's ability to recognize patterns over time, they possess a memory component that allows them to retain information about previous inputs. This makes them particularly effective in tasks such as video classification, where temporal dependencies play a crucial role. RNNs can learn and generalize from sequential examples, capturing nuanced relationships within video data even when the precise underlying connections are elusive. Their universal functional approximation capabilities empower RNNs to model complex nonlinear relationships without prior knowledge of the intricate interplay between input frames and output predictions.

In summary, CNNs and RNNs in image and video processing leverage their respective architectural strengths to learn, generalize, and forecast patterns within visual data, showcasing their adaptability and effectiveness in handling complex tasks inspired by human cognitive processes.

3.1 Object Detection Model

The chosen architecture for our object detection model is RetinaNet, a robust framework initialized using `torchvision.models.detection.retinanet_resnet50_fpn_v2`. It combines a ResNet-50 backbone with a Feature Pyramid Network (FPN) for effective multi-scale feature extraction.

The RetinaNet classification head has been customized to suit the specific requirements of the task:

- The number of anchors has been adjusted based on the model's default configuration.
- A group normalization layer with 32 groups has been introduced.
- The loss function is set to a custom Focal Loss, designed to handle class imbalance in object detection.

3.1.1 Loss Function

The Focal Loss employed in this model is tailored to address class imbalance in object detection. It includes tunable hyperparameters:

- `alpha` for balancing class weights.
- `gamma` for controlling the focus on hard-to-detect instances.

3.1.2 Model Configuration

During model instantiation using the `create_model` function, specific parameters are configured to tailor the model's behavior according to the task requirements.

- The model is designed to detect objects belonging to two classes only: 'Player' and 'Ball.'
- Batch size is set to 16 for efficient GPU memory utilization.
- Base image resolution for transformations is set to 640 x 640.
- The model is trained for 75 epochs to capture diverse patterns in the data.
- Parallel data loading with 4 workers enhances training efficiency.
- The initial learning rate is set to 0.001 for effective weight updates.

Multi-Resolution Training

While our model primarily processes images at the base resolution, we explore the nuances of multi-resolution training and its implications. This allows us to understand the trade-offs associated with resolution variations during the training process.

Multi-resolution training is disabled (`RESOLUTIONS = None`). The model processes images at the base resolution and does not vary the resolution during training.

3.2 Semantic Segmentation Model

Semantic segmentation is a crucial component of our system, enabling precise delineation of players and other objects within each frame. It involves classifying each pixel in an image into a predefined category. For our football player tracking system, this translates to accurately segmenting players from the background, providing detailed information about player positions and movements.

3.2.1 Fine-Tuning Meta SAM for Football Player Tracking

For our project, we fine-tuned the Meta SAM model to specialize in segmenting football players from match footage. The process involved adapting the model to handle the unique challenges of sports video data, such as varying lighting conditions, occlusions, and dynamic backgrounds.

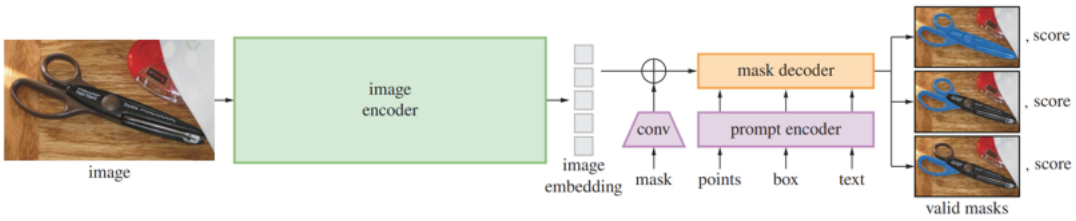

Figure 3.1: Meta SAM Model Architecture

The Meta SAM architecture features several key components that make it particularly effective for our application:

- **Backbone:** SAM uses a powerful backbone network (e.g., a ResNet or a Vision Transformer) pre-trained on large-scale datasets. This backbone extracts rich feature representations from input images, capturing both local and global contexts.
- **Segmentation Head:** The segmentation head processes the feature maps from the backbone and produces pixel-wise classifications. It employs a combination of convolutional layers and upsampling techniques to generate high-resolution segmentation masks.
- **Multi-Scale Feature Fusion:** SAM integrates multi-scale feature fusion to enhance its ability to handle objects at different scales. This is particularly important for accurately segmenting players who may appear at varying sizes within the frame.

3.2.2 Loss Function

The fine-tuning process involves optimizing the model using a combination of loss functions designed to improve segmentation accuracy and robustness:

Cross-Entropy Loss: This loss function is widely used for classification tasks and is calculated for each pixel to encourage correct class predictions.

$$L_{ce} = -\sum_{i=1}^{C} y_i \log(p_i) \tag{3.1}$$

Dice Loss: measures the overlap between the predicted segmentation mask and the ground truth mask, focusing on the similarity between sets of predicted and actual pixels. It is particularly effective for handling imbalanced datasets.

$$L_{dice} = 1 - \frac{2\sum_{i}(inputs_i \times targets_i) + \epsilon}{\sum_{i} inputs_i + \sum_{i} targets_i + \epsilon} \tag{3.2}$$

Boundary Loss: This loss function emphasizes accurate segmentation boundaries, which is critical for precisely delineating players in complex scenes. It penalizes deviations from the true boundary.

$$L_b = \frac{1}{T}\sum_{c=1}^{C}\int_{\Omega} \omega_c(x)||\nabla u_c(x) - \nabla v_c(x)||^2 dx \tag{3.3}$$

3.3 Visual Tracking Model

The Visual Tracking Module has a pivotal role within the player tracking system, deploying an intricate custom fully convolutional network (FCN) architecture. This module is dedicated to the generation of precise monochromatic masks representing individual players, thereby enabling meticulous tracking across successive frames.

The masks generated by this module are fundamental inputs for subsequent phases of the player tracking process. These masks significantly contribute to the overall system's efficacy in precisely locating and tracking each player throughout the temporal evolution of a football matches.

3.3.1 FCN Architecture

At the core of the Visual Tracking Module lies a FCN that transform input images into informative player masks. The employed deep learning model for binary segmentation, a variant of the Fully Convolutional Network (FCN) architecture. It is designed to process input images and predict a pixel-wise segmentation mask, where each pixel is classified into a specific category.

The iterative refinement of the architecture and training strategies underscores the mod-

ule's adaptability to diverse player appearances and dynamic on-field scenarios. This continual improvement culminates in heightened precision and reliability within the player tracking subsystem.

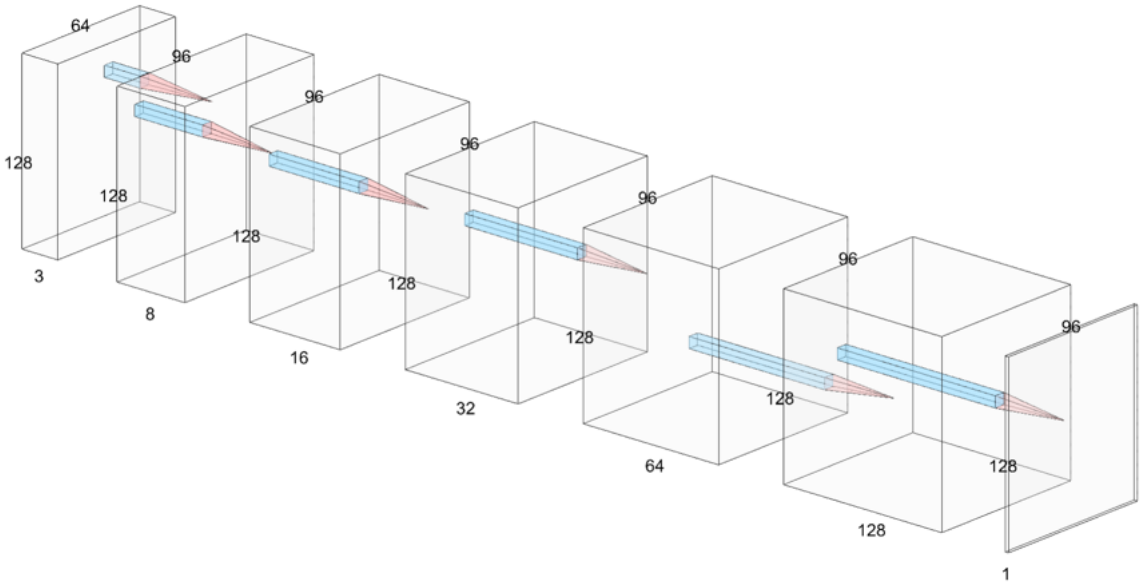

Figure 3.2: FCN Model Architecture

- **Convolutional Layers:** The network employs a strategically designed series of convolutional layers with increasing feature channel depths (number of filters) from 3 (RGB) to 8, 16, 32, 64, 128. These layers progressively extract higher-level features from the input image.
- **ReLU Activation:** Employing ReLU activation functions after each convolutional layer introduces non-linearity, enhancing feature representation and allowing it to learn more complex relationships within the data.
- **Output Layer:** The final convolutional layer has only one filter, resulting in a single-channel feature map. This map represents the predicted segmentation mask. A sigmoid activation function is applied to the output, generating values between 0 and 1 for each pixel. These values indicate the probability of a pixel belonging to the foreground class (likely assigned value 1) or the background class (likely assigned value 0).

Model Advantages

- **Fully Convolutional:** The model relies solely on convolutional layers, allowing it to process images of any size while maintaining spatial information crucial for segmentation tasks.
- **Simple and Efficient:** The architecture is relatively straightforward and avoids complex operations like pooling layers, potentially leading to faster training and inference compared to some other segmentation models.

- **Learns Hierarchical Features:** The increasing number of filters in the convolutional layers allows the network to extract features at different levels of abstraction, from low-level edges and textures to high-level object shapes and relationships.

3.3.2 Initial Training Approach

The Visual Tracking Module is initially trained using multiple images for each player and its corresponding mask, in addition to some examples of the background coupled with completely black masks, aimed at optimizing the model's parameters for accurate player binary mask generation. The training process involves the utilization of a complex weighted combined loss of three loss functions as the optimization criterion to measure the difference between the predicted and ground truth masks, coupled with the Adam optimizer to efficiently update the model parameters.

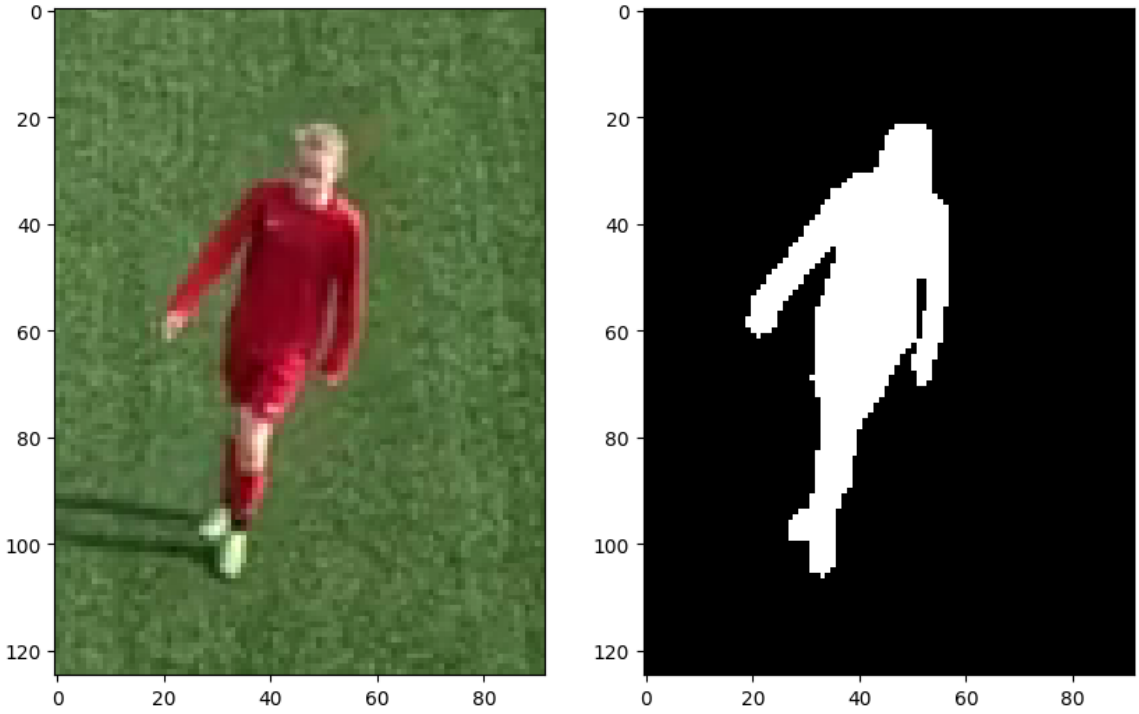

Figure 3.3: FCN Training Input

3.3.3 Combined Loss

The FCN training process involves minimizing the combined loss function as an optimization criterion. This formula incorporates three individual loss components: Focal Loss, Dice Loss, and Intersection over Area (IoA) Loss, where the weighted sum of all of them is calculated.

A lower **Dice Loss** value indicates better overlap between the predicted and ground truth

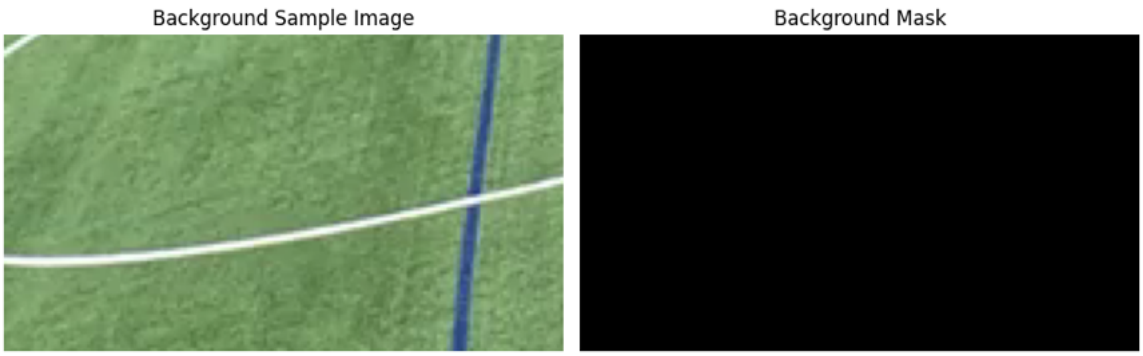

Figure 3.4: FCN Background Sample Training Input

segmentation masks.

$$L_{dice} = 1 - \frac{2\sum_{i}(inputs_i \times targets_i) + \epsilon}{\sum_{i} inputs_i + \sum_{i} targets_i + \epsilon} \tag{3.4}$$

The **focal loss** addresses the class imbalance issue that might be present in semantic segmentation datasets, where the background class often dominates. It provides higher weight to misclassified foreground pixels compared to background pixels.

$$L_{focal} = \alpha_t(1 - p_t)^{\gamma} \times L_{bce} \tag{3.5}$$

The **IoA Loss**, similar to Dice Loss, measures the overlap between the predicted and ground truth segmentation masks. However, it uses a slightly different formulation:

$$L_{IoA} = 1 - \frac{\sum_{i}(inputs_i \times targets_i)}{\sum_{i} targets_i + \epsilon} \tag{3.6}$$

Combining the Losses:

$$L_{combined} = 0.65 \times (L_{focal} + L_{dice}) + 0.35 \times L_{iou} \tag{3.7}$$

The combined loss function utilizes weighted summation to combine the individual losses. The specific weights, can be tuned to prioritize specific aspects based on the dataset characteristics and desired model behavior. In this work, the focal loss and Dice loss are given more weight (0.65) compared to the IoU loss (0.35). This might be due to the potential benefits of focal loss in handling class imbalance and Dice loss in promoting accurate foreground segmentation.

3.4 Path Prediction Model

The path prediction module is needed for cost efficiency, as it allows the use of less expensive models when not needed. That is why path prediction is required to be able to detect if any occlusion (players covering each other from being viewed from the camera) will occur to

substitute models with each other in addition to boosting the visual tracker performance by guiding it through different possible connected components to choose from.

Various attempts were made using knowledge-based systems like the Kalman Filter Predictor or using LSTM NN to be able to predict the upcoming movement of the player. However, these systems did not produce reliable enough accuracies to be useful. Our updated approach leverages a fully connected Multi-Layer Perceptron (MLP) architecture for this task, informed by the Kolmogorov-Arnold representation theorem [16]. This theorem posits that any continuous function can be represented by a composition of continuous functions, suggesting that an MLP can effectively model complex relationships within the data.

3.4.1 Previous Models

Initially, our path prediction module utilized Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs) like U-Nets. While these models demonstrated reasonable performance, they had limitations in capturing the intricate dynamics and dependencies over extended sequences. The LSTM model, although capable of learning long-term dependencies, often struggled with the computational complexity and convergence issues. Similarly, U-Nets, though powerful in capturing spatial features, faced challenges in modeling temporal dynamics effectively.

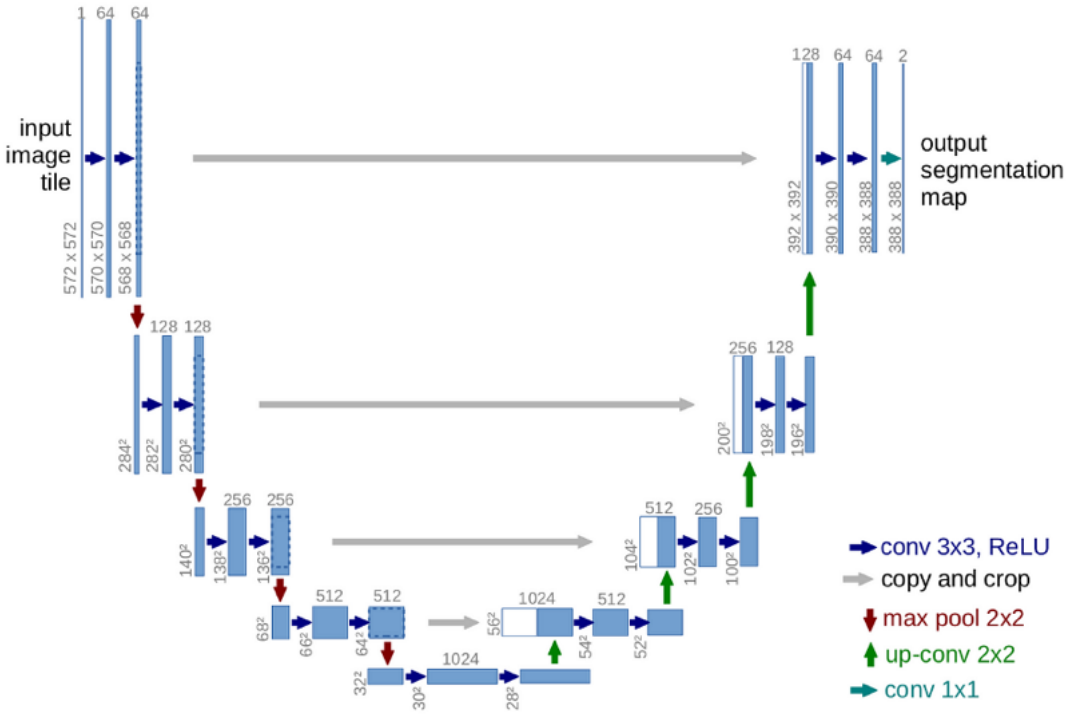

Figure 3.5: UNET Architecture

3.4.2 Fully Connected MLP Architecture

To address these limitations, we transitioned to a fully connected MLP for path prediction. The architecture consists of several densely connected layers, each followed by non-linear activation functions. This design choice is grounded in the Kolmogorov-Arnold representation theorem, which guarantees that an MLP can approximate any continuous function given a sufficient number of neurons and layers.

The choice of a fully connected MLP is motivated by its flexibility and powerful function approximation capabilities, as described by the Kolmogorov-Arnold representation theorem. This approach enables the model to learn complex mappings from historical position data to future trajectories, surpassing the performance of previous LSTM and CNN-based models.

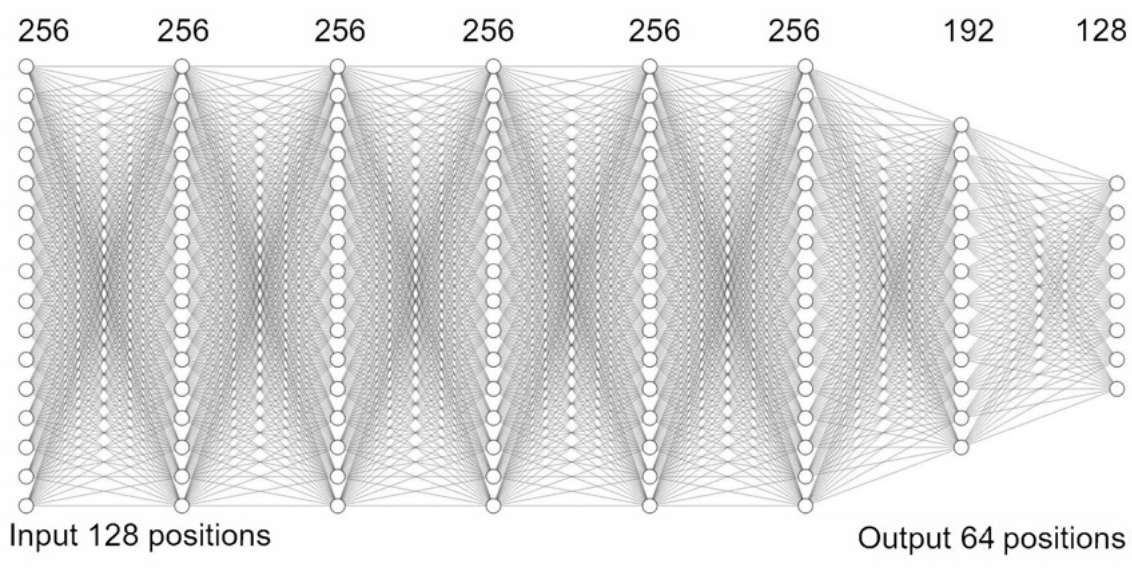

Figure 3.6: MLP Architecture for Path Prediction

- **Input Layer:** The model takes as input the latest 128 (x , y) positions of a player, represented as a vector of 256 values.
- **Hidden Layers:** The network includes multiple hidden layers, each with a substantial number of neurons to capture the complex relationships between past positions and future movements. These layers employ ReLU (Rectified Linear Unit) activation functions to introduce non-linearity.
- **Output Layer:** The output layer produces the predicted 64 next (x , y) positions, represented as a vector of 128 values.

3.4.3 Training and Optimization

The MLP is trained using a mean squared error (MSE) loss function, which measures the average squared differences between the predicted and actual future positions. We employ the

Adam optimizer, known for its efficiency and adaptive learning rate capabilities, to minimize the loss function during training.

$$\text{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.8}$$

A model for each camera view was trained on history of tracks of all players from this camera, these samples can be visualized as follows where each picture shows the input 128 positions in orange and the output 64 positions track in green.

Samples of how the training data visually looks like are shown in figure 3.7.

3.5 Similarity Model

The similarity module is crucial for the re-identification of players in various frames, ensuring consistent tracking even when players are occluded or temporarily out of the camera's field of view. In our updated system, we have integrated the Swin Transformer, a state-of-the-art vision transformer architecture, to enhance the performance of the similarity module. This section details the architecture, advantages, and implementation of the Swin Transformer for player re-identification.

3.5.1 Previous Approaches

Initially, our similarity module relied on convolutional neural networks (CNNs) for feature extraction and comparison. While effective to some extent, CNNs often struggled with capturing long-range dependencies and contextual information, which are critical for accurately distinguishing between players in complex scenes.

3.5.2 VAE Architecture

Initially, our similarity module relied on Variational Autoencoders (VAEs) for feature extraction and comparison. While VAEs were effective in learning compact representations of player appearances, they often struggled with capturing long-range dependencies and contextual information, which are critical for accurately distinguishing between players in complex scenes.

3.5.3 Swin Transformer Architecture

The Swin Transformer, introduced by Liu et al. [33], represents a significant advancement in the field of vision transformers. Its hierarchical architecture with shifted windows allows it to process images efficiently while maintaining a balance between local and global feature extraction. This makes it particularly suitable for the task of player re-identification, where both fine details and broader contextual information are essential.

(a) MLP Train Sample 1

(b) MLP Train Sample 2

Figure 3.7: MLP Train Samples

- **Hierarchical Representation:** The Swin Transformer processes images in a hierarchical manner, starting with small patches and progressively merging them into larger patches. This allows the model to capture fine-grained details at lower levels and broader contextual information at higher levels.

- **Shifted Window Mechanism:** The unique shifted window approach ensures that the model can efficiently capture long-range dependencies without the quadratic complexity associated with traditional transformers. This is achieved by shifting the window partitions between successive layers, enabling cross-window connections and enhancing the model's ability to capture global context.

- **Multi-Head Self-Attention:** The model employs multi-head self-attention mechanisms within each window to capture intricate relationships between different parts of the image. This allows for more nuanced feature extraction, crucial for distinguishing between similar-looking players.

3.5.4 Loss Computation

Following the feature generation phase, the resultant features are sent to the fusion loss stage, where three distinct loss functions—cross-entropy, triplet loss, and focal loss—are calculated. These loss functions are then combined and sent to a fully connected (FC) layer for ID prediction.

Cross-Entropy Loss encourages correct classification by calculating the probability of the i-th sample's predicted person classification score against the true label of the i-th sample.

$$L_{ce} = -\sum_{i=1}^{C} y_i \log(p_i) \tag{3.9}$$

Triplet Loss focuses on inter-sample similarities, facilitating the acquisition of intricate image details throughout the learning process. It involves inputting three paired images: an anchor image (a), a positive sample (p) with the same ID as (a), and a negative sample (n) with a different ID.

$$L_t = \max\{d(a,p) - d(a,n) + m, 0\} \tag{3.10}$$

Focal Loss addresses class imbalance issues by increasing the weight of hard-to-separate samples in the overall loss and decreasing the influence of easy-to-separate samples.

$$L_{focal} = \alpha_t (1 - p_t)^{\gamma} \times L_{bce} \tag{3.11}$$

The Fusion Loss uses a combination of these three loss functions. The goal of using fusion loss is to improve the network by letting different loss functions limit each other, thereby enhancing the model's ability to learn representative characteristics.

$$L_{Fusion} = L_{CE} + L_t + L_{focal} \tag{3.12}$$

By leveraging this fusion loss, the proposed model learns not only to accurately classify but also to capture fine-grained similarities and effectively handle hard and imbalanced data, which increases the model's discriminative strength and facilitates the model's ability to learn effectively. The model parameters are updated to minimize this combined loss using the Adam optimizer.

Chapter 4

Solution Architecture

4.1 System Overview

The football analytics system is designed to revolutionize player tracking and analysis in the dynamic world of football. Our system is a sophisticated blend of cutting-edge technology and a deep understanding of the sport, aiming to provide comprehensive insights into player movements and team dynamics.

This chapter delves deep into the intricate architecture of our multi-module player tracking system, designed to dominate the dynamic world of football matches. The system orchestrates five specialized modules in a distributed dance, each contributing unique skills to achieve robust and accurate player tracking.

At its core, the system seeks to automate and enhance the traditional manual methods of tracking player movements during football matches. The purpose is to provide accurate positional data for every player and the ball at every moment, across multiple camera feeds. This data forms the basis for in-depth player and team performance analysis.

The system's functionality is driven by the integration of deep learning advancements with other computer vision techniques. By harnessing the power of artificial intelligence, our goal is to overcome the limitations of manual tracking, such as labor-intensive processes, limited granularity, and occlusion issues.

Goals

Our primary goals include:

- Develop a modular system with specialized modules for player tracking, object detection, path prediction, semantic segmentation, and similarity analysis.
- Provide accurate and detailed player statistics for performance assessment.
- Create a dynamic user interface for presenting the output statistics in an intuitive and graphically rich format.

- Ensure the system's efficiency in tracking all players and the ball, even in fast-paced scenarios.
- Facilitate human-agent interaction for updating player identification in cases of uncertainty.

Through this system, we aim to redefine how football analytics is conducted, offering a valuable and smart toolset for coaches, analysts, and football enthusiasts to gain deeper insights into the beautiful game.

4.2 A Modular Architectural Design

The system architecture is designed to seamlessly integrate multiple modules, rather than relying on a single conductor each specializing in a crucial aspect of the tracking process. The architecture is modular, allowing for flexibility, scalability, and efficient communication between components.

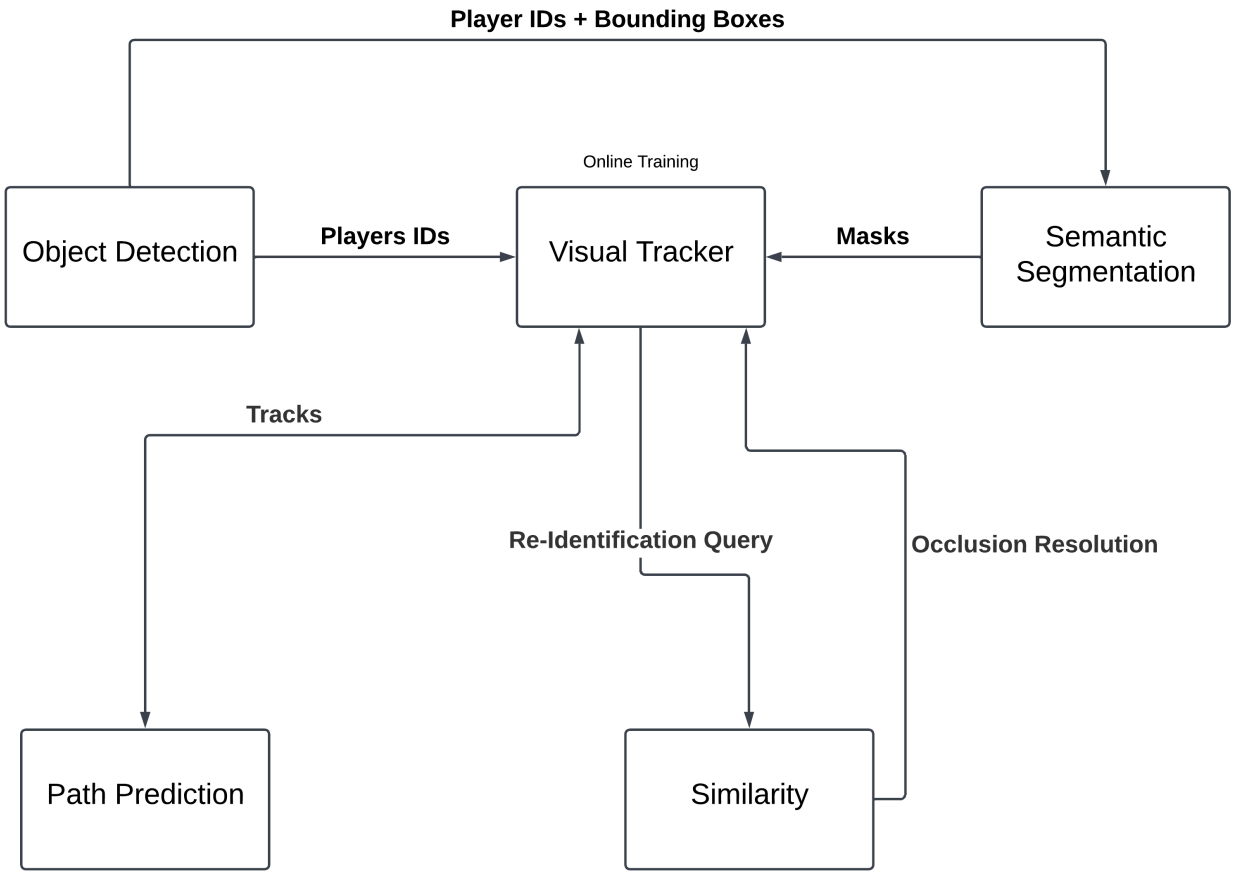

Figure 4.1: Modular System Components

The modules communicate seamlessly to ensure the flow of data across the system. A high-level communication diagram is presented in Figure 4.1. The modular architecture ensures that

each component can be individually upgraded or replaced without affecting the entire system. This flexibility allows for the incorporation of advanced technologies and methodologies as they emerge in the field of football analytics. Five modules, each a virtuoso in its own right, work together in seamless harmony.

4.2.1 Object Detection Module

Within our player tracking symphony, the object detection module acts as a spotlight operator, precisely illuminating the players and the ball on the pitch. It wields ResNet50 as its discerning lens, identifying these key objects and outlining their positions within the scene. When new players enter the stage, this module steps up, using the sharp eyes of ResNet50, trained on 10,000 football match labeled frames. It assigns unique IDs and generates initial masks, setting the scene for the other modules to shine. It identifies and locates objects such as players and the ball within the camera feeds.

Detection and Localization:

- Receives video frames from the system's input pipeline.
- Employs ResNet50 to detect players and the ball within each frame.
- Generates bounding boxes surrounding each detected player and ball.
- Assigns unique IDs to each detected player.

Module Output

Provides the following information to subsequent modules:

- Bounding boxes for each detected player and the ball, specifying their coordinates and dimensions.
- Unique player IDs, enabling identification and tracking.

Activation:

The object detection module is called upon for initial identification of players and the ball at the start of the analysis process. In addition, It is engaged when new players enter the field of view or when occlusions require re-identification. It also collaborates with the visual tracker module to maintain tracking.

Key Considerations:

- **Accuracy:** Precise detection and localization are essential for accurate tracking and analysis.

- **Speed:** Real-time processing demands efficient inference.
- **Robustness to Occlusions:** Handling partial visibility is crucial for maintaining tracking integrity.
- **Integration with Other Modules:** Seamless collaboration with the visual tracker, path prediction, semantic segmentation, and similarity modules is vital for comprehensive player tracking.

4.2.2 Semantic Segmentation Module

For semantic segmentation, the system employs a fine-tuned Meta SAM (Segment Anything Model). This model provides precise segmentation of players, which is essential for distinguishing individual players from the background and other elements on the field working hand in hand with the object detection module to start and furhter improve the visual tracker processing and functionality.

4.2.3 Visual Tracker Module: Persistent Player Trajectory Analysis

Within our player analysis symphony, the visual tracker module plays the role of a persistent and steadfast conductor, orchestrating the spotlight monitoring on each player's movements across the pitch. The module processes subsequent video frames, employing a multi-stage approach where it wields the intricate language of fully convolutional networks, connected components analysis, and moving window algorithms to maintain a continuous understanding of player trajectories.

Functionality

1. **Initializing Player Representations and Activation:**
    - Initiated after initial player identification and segmentation by the object detection and the semantic segmentation modules respectively.
    - Trains for 500 epochs on the player masks in addition to some random background crops.
    - Initialize a moving window for every player using the given bounding box, where:

$$\begin{aligned} \text{width of the window} &= 2 \times \text{bbox width}, \\ \text{height of the window} &= 2 \times \text{bbox height}, \\ \text{center of the window} &= \text{center of the bbox}. \end{aligned}$$

2. **FCN Feedforward:**

- Applies the trained FCN to generate masks for each tracked player to generate binary (black and white) masks of the given window.

3. **Connected Component Analysis:**

   - Applies connected components analysis to discern individual players within the generated masks, and draw bbox on the edges of the largest connected component in the window mask.

4. **Update Moving Window:**

   - Employs a moving window algorithm to dynamically follow players across frames, to minimize the processing time and effort needed where the FCN works on a fraction of the frame instead of processing the full frame.
   - The window center point (x,y), width, and height are all updated at different factors taking into consideration the nature of the game dynamics.
   - This moving window algorithm is scale invariant as it adapts to changing sizes of the players as they move closer or further from the camera.

5. **Online Adaptation:**

   - Executes periodic updates to the DL model through continuous training of the FCN model to refine player representations and adapt to variations in appearance, lighting, or camera angles.

6. **Module Output:**

   - Provides updated bounding boxes and masks for each tracked player, reflecting their positions and movements in each frame.
   - Provides complete or partial tracks to be used by the path prediction module to predict the next positions.

Key Considerations

- **Accuracy:** Precise tracking of player trajectories is essential for reliable analysis of player behavior and interactions.
- **Robustness to Occlusions:** Handling partial visibility and overlapping players effectively is crucial for maintaining accurate tracks.
- **Efficiency:** While not real-time, the visual tracker module utilizes an efficient tracking algorithms which is super important to minimize processing time.
- **Adaptability:** The ability to adapt to appearance changes and variations in camera views is crucial for maintaining tracking integrity.

4.2.4 Path Prediction Module: Foreseeing Occlusions, Optimizing Efficiency

Within our player tracking orchestra, the path prediction module plays the role of a cautious prophet, peering into the future to anticipate occlusions and guide the system towards cost-effective tracking. It utilizes the powerful MLP architecture as a seer of player trajectories, enabling strategic resource allocation. By predicting future trajectories, it warns the other modules of potential occlusions, allowing for smoother transitions and boosts accurate tracking.

By utilizing the path prediction module's foresight, the system can navigate the cost-efficiency terrain effectively, allocating resources wisely while maintaining accurate player tracking throughout the analysis.

Functionality

1. **Visual Tracker Guidance:**
    - Receives the last 128 positions of the player.
    - Employs the trained MLP model to predict the next 64 positions of the player.
    - Guides the visual tracker as it chooses between different connected components available in the FCN output mask by weighing in the factor of the component distance from the predicted position.
    - Guides the window centroid update as the visual tracker update its window using a weighted sum of the previous window, new bounding box, and the predicted position.
2. **Cost-efficient Resource Allocation:**
    - Leverages occlusion predictions to determine the appropriate tracking model for each player in the next frame.
    - Substitutes computationally expensive models with simpler, yet sufficient, alternatives for players unlikely to be occluded.
    - This dynamic switching optimizes processing time and resource consumption while maintaining accurate tracking.

Module Output

- The next 64 positions x, y pairs of the next 64 frames per player.

Activation

- Operates concurrently with the visual tracker module, analyzing input data periodically.
- Predictions inform model selection choices before tracking updates are performed.

Key Considerations

- **Prediction Accuracy:** Reliable prediction of potential occlusions is crucial for effective resource allocation.
- **Cost Optimization:** Balancing tracking accuracy with computational efficiency is key for maximizing cost-effectiveness.
- **Integration with Other Modules:** Seamless collaboration with the visual tracker is vital for accurate tracking even when occlusions occur.

4.2.5 Similarity Module: Dynamic Player Re-Identification Framework

A bridge across time and space, this module recognizes players even when their appearance changes due to occlusions or camera handoffs. Using the art of swin transformer architecture, it matches players across these gaps, ensuring continuous tracking throughout the match. It analyzes similarity between player looks, helping the re-identification process using similarities.

In the realm of player tracking, ensuring robust identification of individuals under occlusion remains a formidable challenge. Addressing this challenge necessitates the development of innovative solutions that can effectively re-identify players before and after occlusion. This section introduces a novel similarity module designed to leverage swin transformer architecture for the purpose of re-identifying players in the dynamic context of occlusion scenarios.

Model Feedforward

In the feedforward process, the model encodes an input image through its convolutional layers, leading to the extraction of high-level features. Subsequently, during the decoding phase, these encoded features are used to classify the input image.

4.3 Initial Preprocessing:

The initial phase of the system involves the main process, which is responsible for processing the first tens of frames of the video to initialize the tracking system. This phase includes object detection and semantic segmentation to accurately identify and locate players in the frame.

- **Object Detection:** The object detection module uses the RetinaNet model, which has been fine-tuned with optimized hyperparameters and training data specific to football matches. RetinaNet's architecture, known for its ability to detect objects at multiple scales, is particularly well-suited for the dynamic and varied scenes in football.

- **Semantic Segmentation:** For semantic segmentation, the system employs a fine-tuned Meta SAM. This model provides precise segmentation of players, which is essential for distinguishing individual players from the background and other elements on the field.
- **Output:** Bounding boxes, masks, and assigned IDs for detected players and ball.

4.4 Integration and Data Flow

The efficacy of our player tracking system is contingent upon the orchestrated collaboration of its constituent modules, intricately interweaving information through a systematically designed data flow. This section delves into the technical underpinnings of the integration mechanisms and data exchange protocols that govern the seamless functioning of these modules, revealing the meticulous ballet orchestrating precise player analysis.

In this section, we present a comprehensive overview of the system designed to track football players using advanced neural networks. The architecture incorporates multiprocessing, message queuing, and MQTT (Message Queuing Telemetry Transport) to ensure efficient, real-time processing and communication between the various components of the system. This design aims to achieve scalability, reliability, and high performance in a demanding real-time environment.

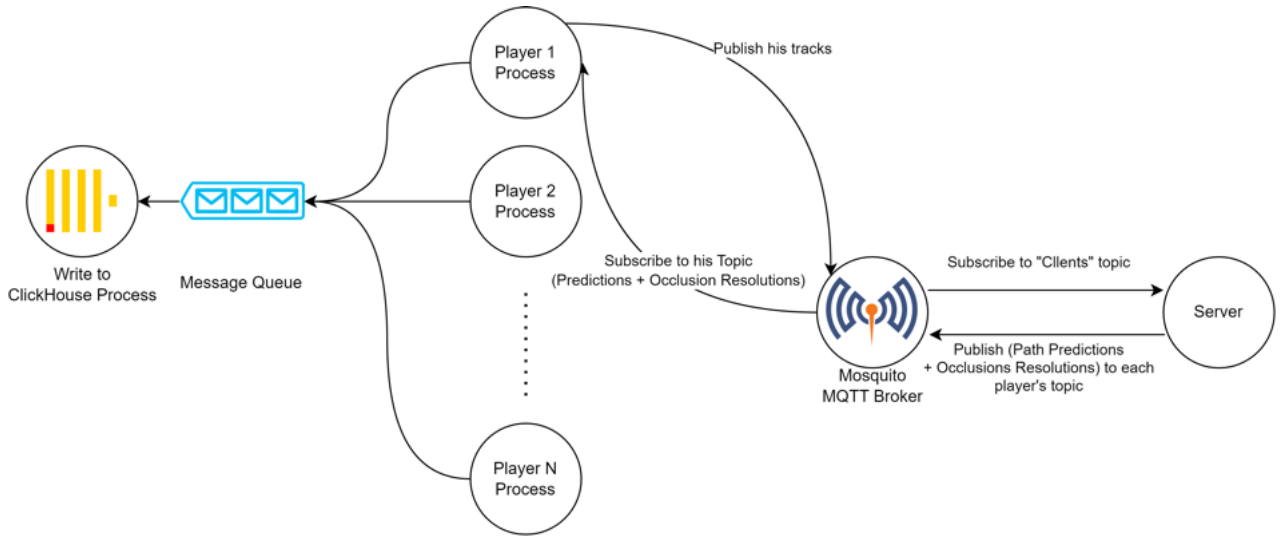

Figure 4.2: Data Flow & System Architecture

At its core, the architecture leverages the strengths of multiprocessing to manage the heavy computational load, message queuing to facilitate smooth data flow, and MQTT for robust and low-latency communication. The architecture is divided into several key components, each playing a specific role in the overall system.

4.4.1 Parallel Visual Tracking:

After the initial processing, the system launches individual player processes. Each player process is dedicated to tracking a single player throughout the match. These processes run concurrently,

leveraging the benefits of multiprocessing to handle multiple players simultaneously.

- **Visual Tracker:** Each player process uses a fully convolutional network-based visual tracker. This network maintains the height and width dimensions of the layers, avoiding pooling to preserve spatial resolution. The tracker continuously updates the position and movement of the player, ensuring accurate tracking data.
- **Publishing Tracks:** The player processes publish their tracking data to the MQTT broker. This data includes the player's position, bounding boxes, and processing window.
- **Output:** Updated bounding boxes and masks for each tracked player every frame.

4.4.2 MQTT for Processes Communication

MQTT is a lightweight messaging protocol designed for constrained environments and low-bandwidth, high-latency networks. It plays a crucial role in ensuring real-time, reliable communication between the different components of our football player tracking system.

Mosquitto Broker:

At the heart of the MQTT-based communication is the MQTT broker, implemented using Mosquitto. The broker acts as a central hub, facilitating message exchange between various processes in the system. It enables decoupling of the data producers (player processes) and data consumers (server and clients), ensuring that each component can operate independently and efficiently.

Player Processes:

Each player process is responsible for tracking a single player and publishing its tracking data to the MQTT broker. This data includes the player's current position, velocity, and other relevant metrics. By using MQTT, the system ensures that the tracking data is transmitted in real-time, allowing for immediate processing and analysis.

- **Publishing Tracking Data:** The player processes publish their tracking data to specific topics on the MQTT broker. For example, Player 1 publishes its data to a topic named player1/tracks. This hierarchical topic structure allows for organized and scalable communication.
- **Subscribing to Topics:** The server subscribes to these player-specific topics to receive the tracking data. This subscription mechanism ensures that the server gets updated information as soon as it is published by the player processes.

The Server:

The server plays a central role in processing the tracking data received from the MQTT broker. It subscribes to all player-specific topics to aggregate the tracking data and perform further analysis, including path prediction and occlusion resolution.

- **Path Prediction:** The server uses the aggregated tracking data to predict the future positions of players. The predicted paths are then published back to the MQTT broker on topics like player1/predictions, ensuring that all components can access the latest predictions.
- **Occlusion Resolution:** When occlusions occur, the server leverages the similarity module to resolve them. The resolved positions are published on topics such as player1/occlusions, ensuring that the tracking data remains accurate and consistent.

Fault Tolerance:

MQTT enhances the system's fault tolerance. If a player process or the server experiences a failure, the MQTT broker can buffer the messages, ensuring that no data is lost. Once the system recovers, the buffered messages are delivered, maintaining data integrity and continuity.

4.4.3 Message Queue Database Integration

Message queues provide a reliable and efficient mechanism for handling the continuous stream of tracking data generated by the player processes, ensuring that it is systematically stored and readily available for analysis. We also explore the role of message queues in facilitating the integration of tracked data into ClickHouse, a high-performance columnar database designed for real-time analytics.

Data Publishing by Player Processes:

Each player process is tasked with tracking a specific player and generating a continuous stream of tracking data. This data includes positional coordinates, and bounding boxes. The player processes publish this data to the message queue, ensuring that it is promptly captured and ready for subsequent storage.

The tracking data is published to the message queue using a predefined structure. Each message contains essential tracking information, formatted in a consistent manner to facilitate easy parsing and processing by the ClickHouse integration component.

Integration with ClickHouse:

A dedicated process, as shown in Figure 4.2 referred to as the ClickHouse Write process, is responsible for consuming the data from the message queue and inserting it into the ClickHouse

database. This process subscribes to the message queue, retrieves the buffered messages, and performs the necessary transformations to match the ClickHouse schema.

4.5 Post-processing

4.5.1 Interpolation of Missing Frames

In the context of player tracking, occasional loss of tracking data can occur due to various factors such as occlusions, rapid player movements, or temporary loss of visibility. To maintain the continuity and accuracy of the tracking data, it is essential to interpolate the missing frames. Interpolation helps in estimating the positions of the players during the frames where data is not available.

Our Approach:

The linear interpolation technique is based on the assumption that the player's movement between two known positions is uniform. While this may not always be accurate, it provides a simple and effective way to estimate missing positions in the absence of more sophisticated prediction models. For most practical purposes, linear interpolation is sufficient to fill in short gaps in the tracking data.

We employ a linear interpolation method to estimate the positions of players in the missing frames. Given two known positions of a player at times t_1 and t_2, where $t_1 < t < t_2$, the position of the player at time t is calculated as follows:

$$\mathbf{p}(t) = \mathbf{p}(t_1) + \frac{t - t_1}{t_2 - t_1} \left(\mathbf{p}(t_2) - \mathbf{p}(t_1)\right) \tag{4.1}$$

Where:

- $\mathbf{p}(t)$ is the interpolated position at time t,
- $\mathbf{p}(t_1)$ is the known position at time t_1,
- $\mathbf{p}(t_2)$ is the known position at time t_2.

This method ensures that the trajectory of the player remains smooth and continuous, providing a reasonable estimate of the player's movement between the known positions.

4.5.2 Transformation and Mapping

To enhance the analysis and visualization of the tracking data, it is often beneficial to transform the data from the camera perspective to a top-down, bird's-eye view. This transformation enables a clearer understanding of player positions and movements on the field.

This subsection describes the methodology used to track player positions from video frames and map them onto a playground image. The process involves several key steps, including approximating necessary calibration matrices, undistorting and rectifying points, applying homographies, and generating output in the form of transformed (x, y) points which are saved in the database.

Calibration and Undistortion

We begin by approximating the camera calibration matrix $\mathbf{K}$ and distortion coefficients $\mathbf{d}$ for both the right and left cameras. These are essential for correcting lens distortions in the video frames.

We approximate the matrices as we do not have access to the camera itself. The calibration approximation is done using the well-known chessboard patterns method [7].

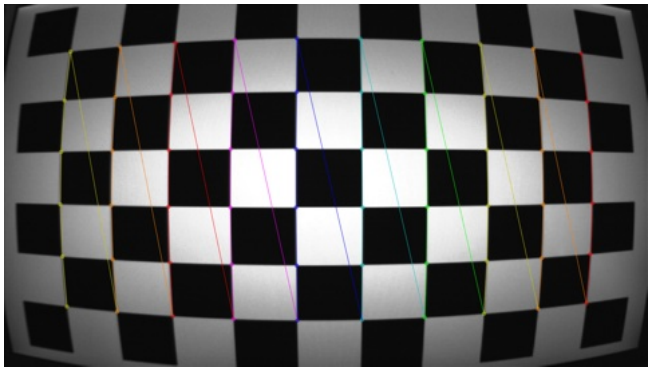

Figure 4.3: Chessboard Pattern for Undistortion Approximation

A sample result of the undistorted picture is shown in Figure 4.4.

(a) Distorted Image (Raw) (b) Undistorted Image

Figure 4.4: Undistortion Result of an Image

Homography Mapping

To map points from the camera view to the playground view, we use homography matrix $\mathbf{H}$. This matrix transform points from the rectified camera coordinates to the playground coordinates.

The homography matrix H is a 3×3 matrix that defines the relationship between the coordinates in the camera's perspective and the top-view coordinates. The transformation of a point (x, y) from the camera to the playground coordinates is given by:

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.2}$$

Where:

- $\mathbf{x} = [x, y, 1]^T$ is the original position in the camera's perspective,
- $\mathbf{x}' = [x_t, y_t, 1]^T$ is the transformed position in the top-view perspective.

To compute the homography matrix H, we use a set of corresponding points between the camera's perspective and the desired top-view perspective. These points can be manually selected as shown in Figures 4.5 for the 2D plane picture and Figure 4.6 for the points shown on the pitch in both cameras.

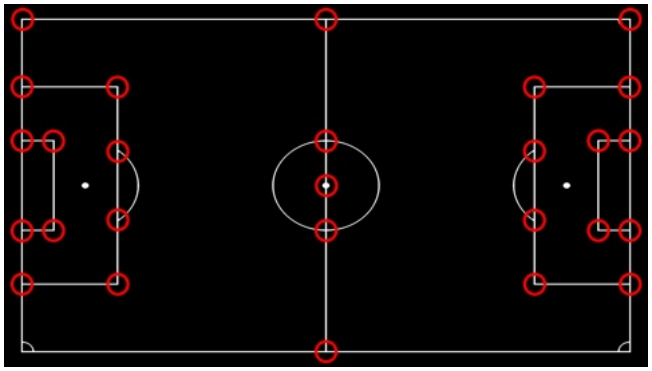

Figure 4.5: Transformation Points on target 2D plane

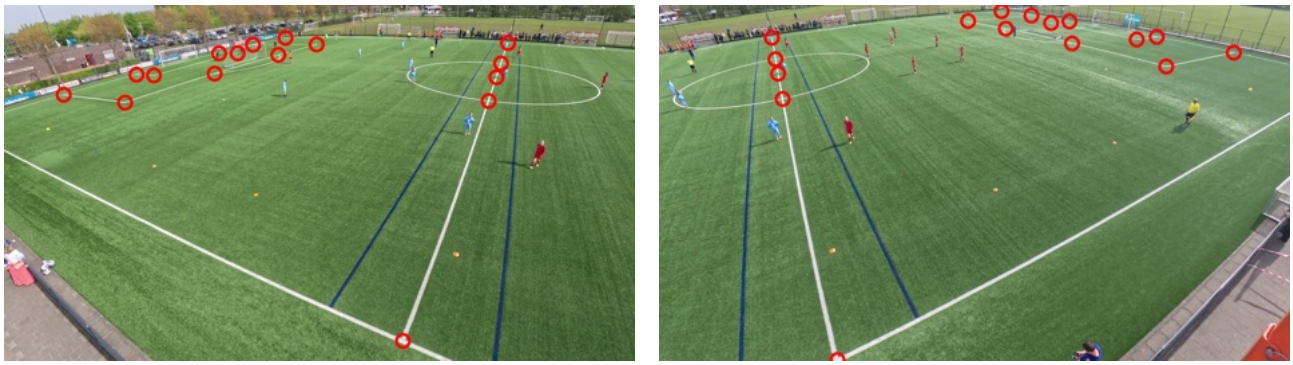

Figure 4.6: Transformation Points on camera pictures

The homography transformation can be applied on the image as a whole or for each of the tracked positions points. The transformed pictures from both cameras are shown in the following figure 4.7.

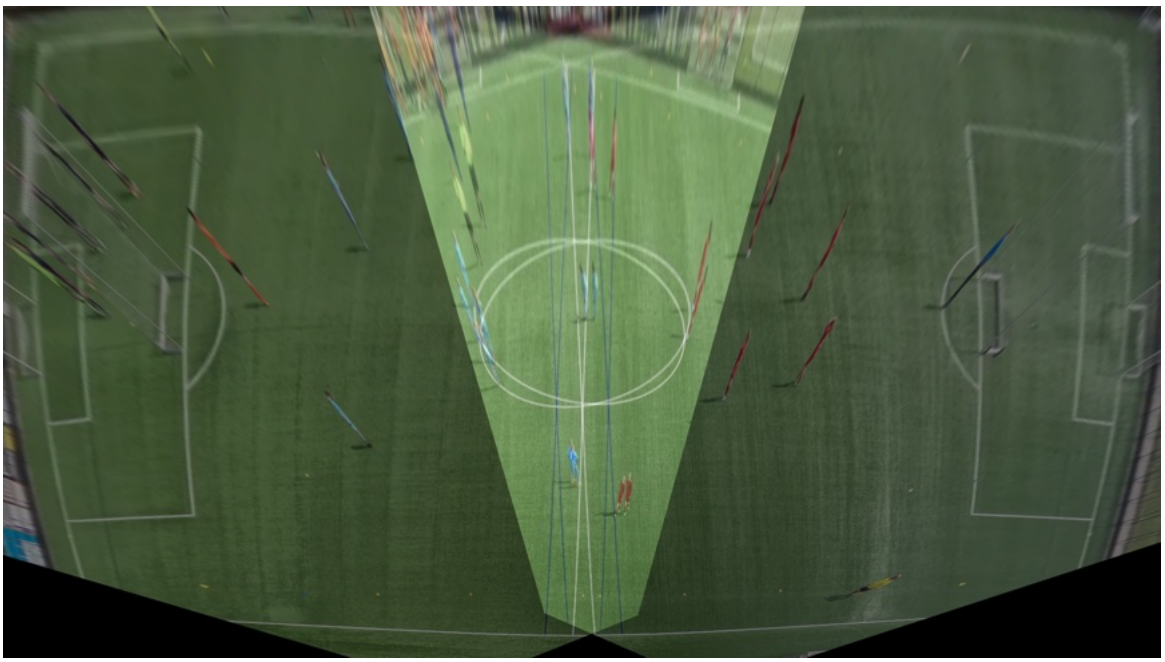

Figure 4.7: Transformed Camera Views

Tracking and Mapping Process

The homography transformation applied to all tracked positions, resulting in a top-down view of the player's movements. This transformation is particularly useful for visualizing the overall player distribution, team formations, and individual player trajectories on the field.

For each player, we process frames from both right and left camera videos. The tracking data is loaded from the database, which contain bounding box coordinates for each player in each frame.

For each frame i:

1. Read the current frame from both right and left camera videos.
2. Check if the player is detected in the right camera frame. If detected, extract the bounding box coordinates $(x1, y1), (x2, y2)$ and compute the center point (x, y).
3. Check if the center point lies within the mask. If not, use the left camera data.
4. Map the point (x, y) to playground coordinates using the respective homography matrix.
5. Draw the transformed point on the playground image as in figure 4.8 and save the results in the database.

This process ensures accurate tracking and visualization of player movements on the playground, providing valuable insights for further analysis.

Figure 4.8: Sample Transformed Player Position

4.6 Statistics Generation and Visualization

The ultimate goal of this project is to generate meaningful statistics for players and teams based on the tracked data collected throughout a football match. After accurately tracking the players and the ball and transforming the data to map it onto a 2D plane, we leverage these tracks to compute various statistics. These statistics include possession, passes, player speeds, distances covered, and more. This chapter will delve into the methodologies and technologies used to calculate these statistics, how the tracking data is retrieved from the database, and how we visualize these statistics using a svelte web application.

4.6.1 Data Retrieval API

To efficiently retrieve tracking data from the ClickHouse database, we developed a custom API using C++. The choice of C++ for the API was driven by its performance benefits, as real-time data retrieval and processing are crucial for generating live statistics during or immediately after a match.

Database Structure

The tracking data is stored in a time-series database, where each entry corresponds to the position of a player or the ball at a given timestamp. The database schema includes the following fields:

- **match_id**: Unique identifier of the match.
- **frame**: The exact frame number of the record.
- **player_id**: Unique identifier for each player.
- **team_id**: Unique identifier for each team.
- **x_coordinate**: Raw X-coordinate of the player or ball from the video view before transformation.
- **y_coordinate**: Raw Y-coordinate of the player or ball from the video view before transformation.
- **x_transformed**: X-coordinate of the player or ball on the 2D plane after transformation.
- **y_transformed**: Y-coordinate of the player or ball on the 2D plane after transformation.

API Design

The API exposes endpoints for querying the tracking data based on various parameters such as time range, player IDs, and team IDs. It is designed to handle high-concurrency requests and optimize data retrieval times. Below is a simplified example of an API request and response:

```
GET /api/tracking?match_id=1&player_id=1

Response:
{
"data": [
{
"frame": "0",
"player_id": 1,
"team_id": 1,
"x_coordinate": 150,
"y_coordinate": 75,
"x_transformed": 210,
"y_transformed": 110
},
...
]
}
```

4.6.2 Statistical Calculations

Once the tracking data is retrieved, various statistical calculations are performed. These calculations are divided into team statistics and player statistics, where every player has his own individual statistics that measures his performance across the match in different times or his total performance of the match. In addition, the team statistics measures how the team performed collectively throughout the match.

Team Statistics:

Team statistics provide insights into the overall performance and strategy of the team. The following are some of the key team statistics calculated.

- **Possession** is calculated by determining the total time each team had control of the ball. This is done by tracking the ball's coordinates and identifying which team's player is closest to the ball at each timestamp. The possession percentage is then calculated as follows:

$$\text{Possession}_{\text{team}} = \frac{\text{frame count of ball possessed}}{\text{total frame count}} \times 100 \tag{4.3}$$

- **Passes** are identified by tracking the ball's movement between players. A pass is recorded when the ball moves from the vicinity of one player to another within a certain time threshold. The number of passes can be aggregated for the entire team or individual players.

Player Statistics:

Player statistics provide detailed insights into individual player performance. The following are some of the key player statistics calculated.

- **Player Speed and Distance Covered** are calculated by dividing the distance covered between two frames by the frame count difference. The total distance covered by a player is the sum of distances covered between all consecutive timestamps. The time can be converted from $\frac{pixels}{frame}$ to $\frac{meters}{second}$ if we know the frame rate of the video (commonly $29.97fps$) and the pitch dimensions in meters compared to the video dimensions in pixels. A dynamic, interactive speed and distance graph for each player can be viewed as shown in Figure 4.9.

$$\text{Speed} = \frac{\text{Distance}_i}{\text{Time}_i} \tag{4.4}$$

$$\text{Total Distance} = \sum_{i=1}^{N} \text{Distance}_i \tag{4.5}$$

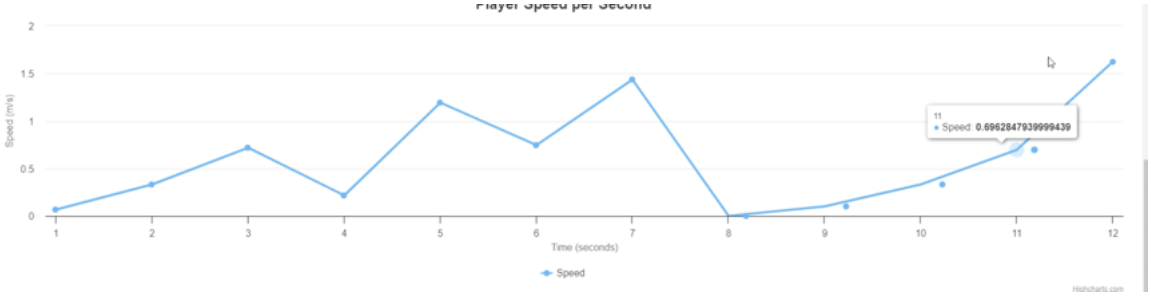

Figure 4.9: Interactive Speed Chart

- **Heatmap** allows user to visualize the zones where the player was active throughout the match using a color scale, example shown in Figure 4.10.

- **Bounding Box Tracking and 2D view** for individual or a set of chosen players, as shown in Figure 4.11.

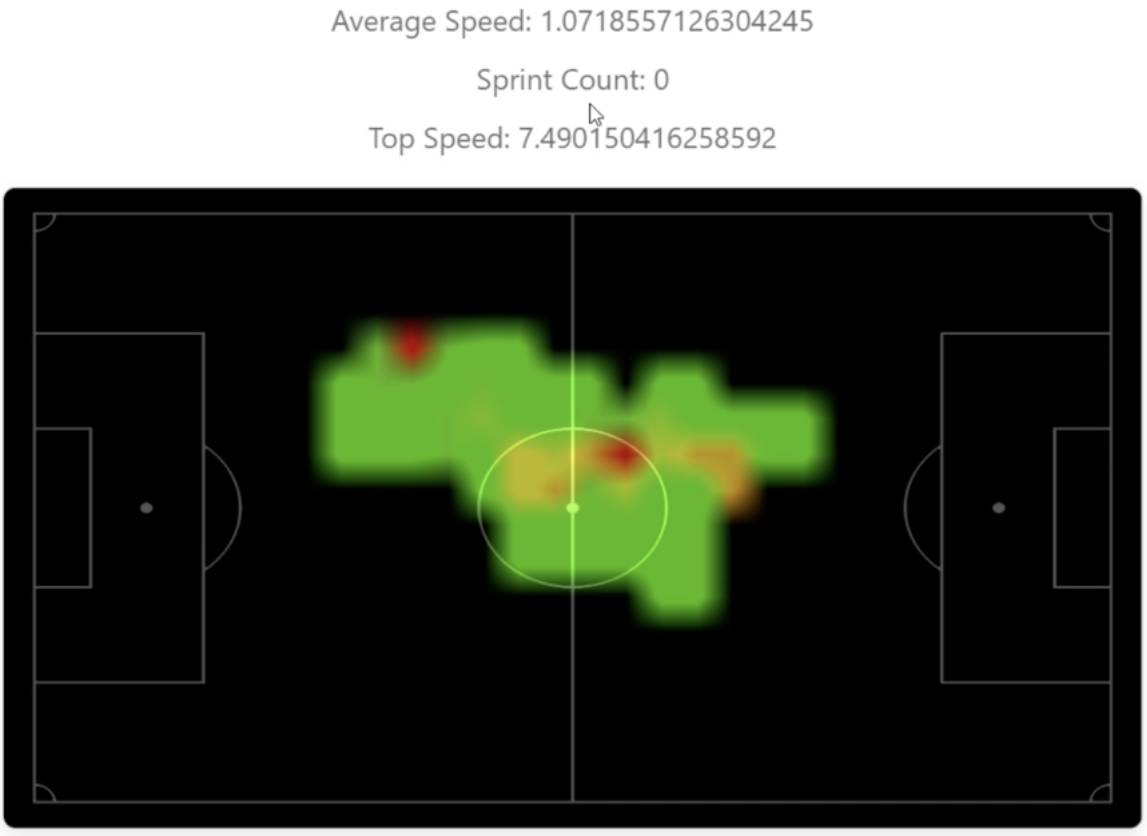

Figure 4.10: Heatmap View

Figure 4.11: Bounding Boxes View

4.6.3 Visualization of Statistics

The visualization of the calculated statistics is a crucial aspect of this project, as it transforms raw data into meaningful insights, enabling stakeholders to gain insights into player and team

performance through intuitive and interactive graphics. To achieve this, we developed a web application using Svelte, a modern JavaScript framework known for its efficient and reactive nature. This section will detail the implementation of the web application, the various interactive visuals it provides, and the user interface design that allows users to select matches and players for detailed analysis.

The web application is designed to be user-friendly, allowing users to easily select matches, view player statistics, and visualize tracking data. Upon entering the application, users are presented with the Match Selection Page. This page allows users to choose a match from a drop-down menu populated with available game data. The selection is crucial as it sets the context for all subsequent visualizations and statistical analyses. Figures 4.12, 4.13, 4.14, 4.15, 4.16 shows snapshots of different pages of the website where the views of the stats are intuitive and interactive.

Figure 4.12: Screenshot from Match Tracked Players Camera View

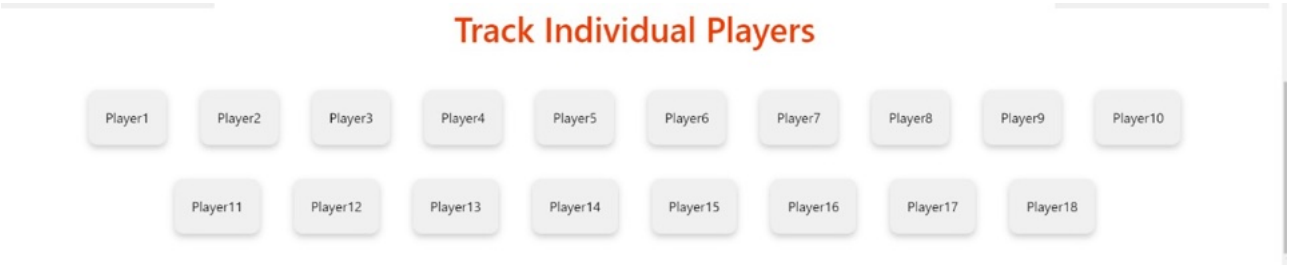

Figure 4.13: Choose Player to View His/Her Data

4.7 Benefits of Integrated Architecture

- Seamless collaboration ensures efficient and accurate player tracking.
- The system balances accuracy with computational costs, switching between multiple modules for nearly similar goals such as the visual tracking, semantic segmentation, similarity,

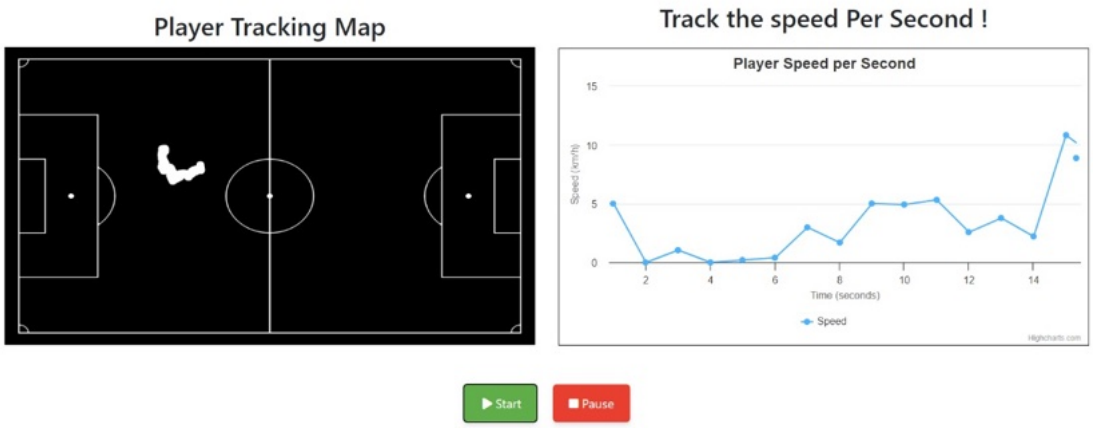

Figure 4.14: 2D Plane Player Tracking View and Speed

Figure 4.15: Player Summary Stats

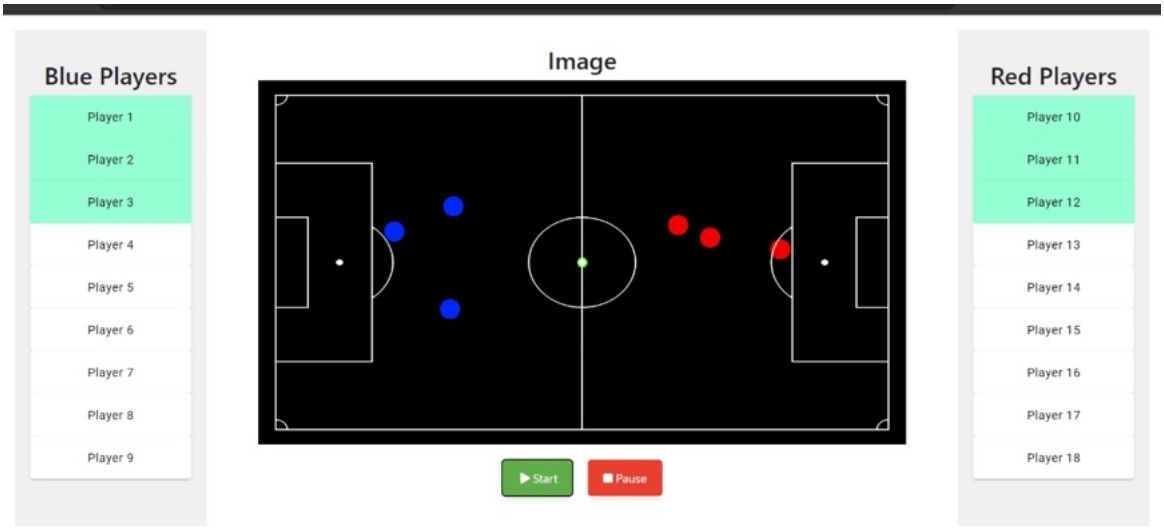

Figure 4.16: Multiple Players 2D View

path prediction and object detection modules. Where the object detection and semantic segmentation provide the highest fidelity when comes to the mask identification of the players but they come with major computational demand over that needed by the much efficient visual tracking module at the cost of medium mask accuracy.

- Path prediction optimizes resource allocation, striking a balance between accuracy and computational efficiency.
- Individual modules address specific tasks, promoting a modular and maintainable system.
- The system dynamically adapts to evolving game situations through continuous learning and data integration.

4.8 Limitations: Acknowledging the Boundaries

Even the most skilled players encounter challenges on the field. In this spirit, we acknowledge the limitations of our current player tracking system, identifying areas where further development could enhance its performance.

By acknowledging these limitations, we pave the way for future research and development, striving to create player tracking systems that can perform even more impressively on the challenging pitch of football analysis.

4.8.1 Challenging Conditions

- **Occlusions:** While the system employs strategies to handle overlapping players, performance can degrade in scenarios with severe or prolonged occlusions.
- **Lighting Variations:** Abrupt changes in lighting conditions, such as shadows or strong sunlight, can impact object detection and tracking accuracy.
- **Camera Views:** The two fixed cameras provide limited perspectives, potentially hindering tracking accuracy in certain areas of the pitch. Even with the undistortion and homography transformation approximations we did, these transformations are not 100% accurate.

4.8.2 Computational Efficiency

- **Processing Time:** While not designed for real-time performance, the system's processing time could be further optimized for faster analysis, especially when handling high-quality videos.
- **Resource Allocation:** The path prediction module strives for cost-efficiency, but further improvements in model selection and resource management could enhance overall performance.

4.8.3 Data and Model Limitations

- **Training Data:** The quality and diversity of training data directly impact model performance. Expanding the dataset with varied scenarios and player appearances could enhance robustness.
- **Model Generalization:** The system's effectiveness may vary across different camera setups, lighting conditions, and player behaviors. Exploring techniques to improve model generalization is crucial for broader applicability.

Chapter 5

Software Requirements Specification (SRS)

The Software Requirements Specification (SRS) outlines the functional and non-functional requirements of the proposed deep learning and computer vision system, emphasizing efficiency in tracking all players and the ball to generate detailed and in-depth statistics. The system aims to provide an intuitive and graphically rich user interface for presenting player statistics dynamically.

5.1 Scope

The scope of the system encompasses the development of a sophisticated efficient computer vision application that utilizes deep learning models for player tracking and in-depth analysis in football matches. The system aims to address the limitations of manual tracking methods and provide accurate positional data for every player, at every moment, across multiple camera feeds. Key functionalities include:

1. **Efficient Player Tracking:** The system must efficiently track each player on his own in addition to the ball throughout a football match.

2. **Occlusion Resolution Strategy:** The system should attempt to undergo occlusion resolution when players overlap and ask for human intervention under certain certainty threshold.

3. **In-Depth Player Statistics:** The system will generate comprehensive player statistics, capturing various aspects and statistics to represent and analyse each player performance in numbers.

4. **In-Depth Team Statistics:** The system will generate comprehensive team statistics, capturing various aspects and statistics to represent and analyse collective team performance in numbers.

5. **Dynamic User Interface:** The user interface will be intuitive and dynamic, providing an intuitive and graphically rich environment for users to interact with deep learning-based player tracking data.

6. **Human-Agent Interaction:** The system may require human-agent interaction in rare cases to update player identification when certainty falls below acceptable measures.

The system's development will adhere to specified budget and time constraints, focusing on efficiency and detailed accurate statistical analysis.

5.2 Functional Requirements

Player Detection and Identification

1. The system shall utilize a collection of state-of-the-art deep learning models for accurate player detection, and identification on the football field at all times.

2. It shall employ a neural network model to distinguish between each player on the pitch.

3. The deep learning models shall be trained to handle variations in player appearance, including different team jerseys and lighting conditions.

Player Tracking

1. The system shall implement deep learning-based tracking algorithms for continuous monitoring of each player throughout the match.

2. It shall leverage convolutional (CNNs) and recurrent neural networks (RNNs) or similar architectures to handle occlusion scenarios, providing robust tracking even in obstructed views.

3. The tracking models shall be adaptable to sudden changes in player speed and direction.

Integration with Deep Learning Models

1. The system shall seamlessly integrate deep learning models with other computer vision techniques for comprehensive player tracking.

2. It shall synchronize information from multiple camera feeds and combine features extracted by deep learning models for a unified view of player movements across the pitch.

User Interface

1. The system shall provide an intuitive user interface for coaches, analysts, and users to interact with deep learning-based player tracking data.
2. Users shall have the ability to customize views, select specific players, and replay specific game moments enriched with insights from deep learning analysis.

5.3 Use Cases

Use Case 1: Real-time Player Tracking

Description: The system shall efficiently and continuously track all players and the ball during a football match, providing accurate positional data.

Actors: Deep Learning Model, Camera Feeds

Flow:

1. The system receives video feeds from multiple cameras.
2. The deep learning model processes the video frames, detecting and identifying players.
3. The tracking algorithm updates the positional data of each player and the ball.
4. In rare cases where certainty falls below acceptable measures, the system alerts a human agent for player identification updates.
5. The system provides the updated player positions to the user interface.

Exception: The system shall handle unexpected interruptions in video feeds gracefully, resuming tracking upon feed restoration.

Use Case 2: Detailed Player and Team Statistics

Description: Users (coaches, analysts, and general users) can interact with the system to access player and team tracking data and in-depth statistics.

Actors: Statistical Analysis Module, User Interface, User.

Flow:

1. The system captures player movements and interactions with the ball.
2. The statistical analysis module processes the captured data to generate detailed player and team statistics.
3. The user interface dynamically presents the generated statistics to users.
4. The user accesses the system through the user interface.

5. The user customizes views, selects specific players, and explores historical data.

6. The system retrieves and displays the requested player tracking information.

Exception: If there are connectivity issues, the system shall provide a user-friendly error message to notify the user and attempt reconnection.

5.4 Non-Functional Requirements

Non-functional requirements define the attributes that characterize the system's operation and performance. These aspects are crucial for ensuring the system's effectiveness, reliability, and user satisfaction.

Performance

Efficiency

The system must efficiently process and analyze player and ball tracking data, providing near-real-time results during and after a football match.

Scalability

The system should be scalable to accommodate an increasing number of players and diverse game scenarios without significant degradation in performance.

Reliability

Availability

The system should strive to maintain high availability, minimizing downtime and disruptions during football matches.

Error Handling

The system must implement robust error-handling mechanisms to gracefully manage unexpected errors and ensure data accuracy.

Usability

User Interface

The user interface must be intuitive, responsive, and graphically rich, ensuring a positive and engaging user experience.

Accessibility

The system should adhere to accessibility standards, ensuring that users with diverse needs can interact with the interface effectively.

Security

Data Confidentiality

Player and team data collected and processed by the system must be kept confidential and protected against unauthorized access.

Integrity

The system must maintain the integrity of tracking data, preventing unauthorized modifications or tampering.

Compatibility

Hardware Compatibility

The system should be compatible with standard hardware configurations, ensuring broad accessibility.

Software Compatibility

The system must be compatible with common operating systems and software environments, facilitating easy integration.

Performance Metrics

Tracking Accuracy

The accuracy of player and ball tracking must meet or exceed predefined benchmarks, ensuring the reliability of generated statistics.

Response Time

The system's response time for user interactions and data processing should adhere to acceptable standards, providing a seamless experience.

5.5 Software and Frameworks

The modular architecture ensures that each component can be individually upgraded or replaced without affecting the entire system. This flexibility allows for the incorporation of advanced technologies and methodologies as they emerge in the field of football analytics.

Our system finds its home on Linux-based operating systems, currently waltzing to the rhythm of Python, PyTorch, and OpenCV. However, a grand migration awaits, where C++ joins the band with LibTorch and OpenCV, enabling parallel processing through multithreading and multiprocessing. This orchestrated hardware-software harmony ensures efficient performance, essential for keeping up with the fast-paced ballet of football. Key software components and frameworks include:

- **Operating System:** The system is developed and tested on Linux-based operating systems to ensure compatibility and stability.
- **Deep Learning Frameworks:** PyTorch is the primary framework for implementing and training deep learning models due to its flexibility and robust tools.
- **Computer Vision Libraries:** OpenCV is utilized for image processing tasks, offering essential functionalities for video analysis.

Development Tools

Development is facilitated by the following tools:

- **Integrated Development Environment (IDE):** Visual Studio Code is used as the primary IDE for coding and debugging.
- **Version Control:** Git is employed for version control, supporting collaborative development and codebase management.

These technologies form the foundation of the football analytics system in the build, emphasizing high-performance computing and efficient processing of video inputs.

5.6 Constraints

Constraints play a crucial role in shaping the boundaries and capabilities of the system. Understanding and addressing these constraints is vital for the successful development and deployment of the computer vision and deep learning-based football player tracking system.

Technological Constraints

PyTorch Framework Compatibility

The system's development is currently constrained by the compatibility of the PyTorch deep learning framework. The selected framework must effectively support the development and integration of advanced models for player and ball tracking. A planned migration to LibTorch in C++ during the second semester introduces an additional constraint for seamless transition and compatibility.

Computational Power

The efficiency of the system depends on the available computational resources. To ensure optimal performance, the system must be designed to leverage the computational power, GPU capabilities, and memory of the underlying hardware.

Human-Agent Interaction

In rare instances where system certainty is below acceptable measures, human-agent interaction may be required for player identification updates. Balancing autonomy with human intervention is a key operational constraint.

Regulatory Constraints

Data Privacy Compliance

The system must strictly adhere to data privacy regulations, ensuring the secure and compliant handling of player and team data. Compliance with regional and international data protection laws is mandatory.

Ethical Use of Data

Development and operation must align with ethical standards, emphasizing the responsible and respectful use of data. Ethical considerations involve protecting the privacy and rights of individuals involved in football matches.

Chapter 6

Results and Evaluation

The development of an innovative sports analytics system necessitates a thorough examination of its constituent modules before their integration into a seamless framework. In this chapter, we present the results of individual prototypes for each module within our proposed system in addition to fully integrated system. The primary objective of these prototypes is to assess the performance, accuracy, and limitations of each module in isolation in addition the full system.

Our approach involves breaking down the system into five distinct modules, each catering to a specific aspect of sports analytics: Player Tracking, Object Detection, Path Prediction, Semantic Segmentation, and Similarity Analysis. By isolating these modules, we aim to gain insights into their independent functionalities, strengths, and areas requiring refinement.

Evaluating each module independently allows us to delve into specific challenges, optimizations, and intricacies associated with player tracking and analysis. The results obtained from these not only serve as benchmarks for individual module performance but also inform the integration process, guiding us towards a cohesive and robust sports analytics system.

This chapter unfolds with detailed accounts of the outcomes, offering a comprehensive understanding of the strengths and limitations inherent to each module. As we navigate through the results, we lay the groundwork for the subsequent chapter, where we explore the integration of these modules to create a holistic sports analytics solution.

6.1 Object Detection Model Refinement

The initial prototype of our object detection model revealed suboptimal performance, largely attributed to inaccuracies in image annotations. Recognizing the pivotal role of precise annotations in shaping the model's understanding, we undertook a thorough reassessment of our annotation strategy to enhance the interpretative capacity of the system.

6.1.1 Enhancements in Annotation Approach

The primary catalyst for significant improvements in model performance stemmed from strategic adjustments in our annotation approach. Key enhancements include:

Comprehensive Annotation of Balls

In the initial phases, our annotation strategy was limited to the match ball, potentially neglecting other instances of balls within the images. Understanding the importance of capturing every ball in the field of view, we refined our annotation protocol to comprehensively label all instances. This expansion ensured a more nuanced understanding of ball-related entities.

Inclusive Annotation of Human Entities

To account for the diverse entities present on the field, including players, referees, and any other personnel, we expanded our annotation criteria. Initially, our focus might have been solely on players, leading to oversight in crucial elements. The revised approach ensures that any human presence within the field is accurately annotated.

These adjustments were pivotal in refining the dataset quality, enabling the model to learn from a more comprehensive set of annotations. As a result, the prototype's performance experienced a significant boost, demonstrating improved accuracy and reliability in object detection.

Iterative Refinement Process

The iterative process of refining annotations underscores the significance of meticulous data preparation in the success of the object detection model. Through continuous refinement and attention to detail, we successfully mitigated the limitations observed in the initial prototype, laying the groundwork for more promising results in subsequent evaluations.

6.1.2 Results Samples

Despite these improvements, challenges persisted, as illustrated in the figure below. Issues such as incorrect ball handling and misidentification of players remained, prompting further investigation and refinement in subsequent phases.

Failed Results: As shown below in the figure, the ball is not correctly handled and not all players detected are actually players.

Figure 6.1: Failed Sample of Object Detection

Improved Results: As shown below in the figure, the ball in addition to the players were correctly identified with high confidence rates.

Figure 6.2: Success Sample of Object Detection

6.2 Visual Object Tracker

The Visual Tracker module, anchored by an Autoencoder, previously discussed, underwent rigorous prototyping and evaluation to gauge its performance in capturing player movements and generating precise masks. The prototype is designed to demonstrate its effectiveness in tracking individual players throughout a football match and the results from the prototype phase illuminate both successes and areas necessitating further refinement.

6.2.1 Tracking Approach

The tracking process is a collaborative procedure between different computer vision (connected component analysis, geometric transformations) and deep learning (the autoencoder model) techniques to generate the desired result.

The procedure is initiated by providing the model with a cropped image of the player along with its corresponding mask. Subsequently, the model begins masking the player in the designated region of the full match frame. The module is responsible for updating the cropped image presented to the model in each subsequent frame, ensuring the continuous refinement of the tracking process.

The tracking process involves the following key steps:

1. **Image Cropping:** Each frame is initially cropped to focus on the specified window containing the player of interest.

2. **Binary Masking:** The FCN model processes the cropped image window to generate a binary mask representing the segmented player.

3. **Connected Components Analysis:** A connected components analysis is applied to identify the player's segmented region within the binary mask.

4. **Bounding Box Calculation:** The bounding box of the player's segmented region is computed, allowing for precise localization.

5. **Adaptive Window Update:** The tracking window is dynamically updated based on the player's movements in addition to the predicted path, ensuring accurate and adaptive tracking.

6.2.2 Results Samples

The Visual Tracker prototype was evaluated on a sequence of frames, and a sample of results are illustrated in the figures. Each sample consists of 2 figures where the first shows the window the autoencoder worked on followed by the connected components analysis used to draw the bbox. The second picture of each sample features the full frame with 2 bbox drawn, the red illustrating the window and the blue one is the final tracked bbox of the player.

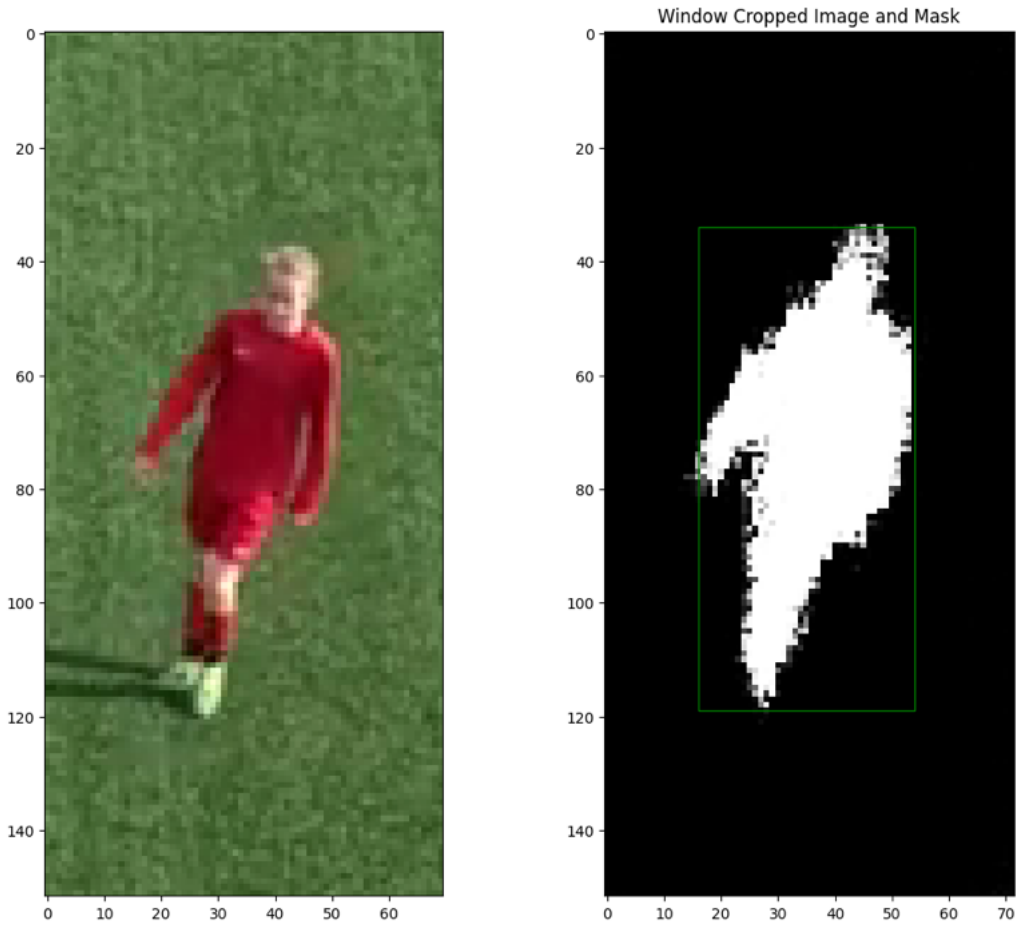

Figure 6.3: Sample 1 of Visual Player Tracking

Figure 6.4: Sample 1 of Visual Player Tracking, Full Frame

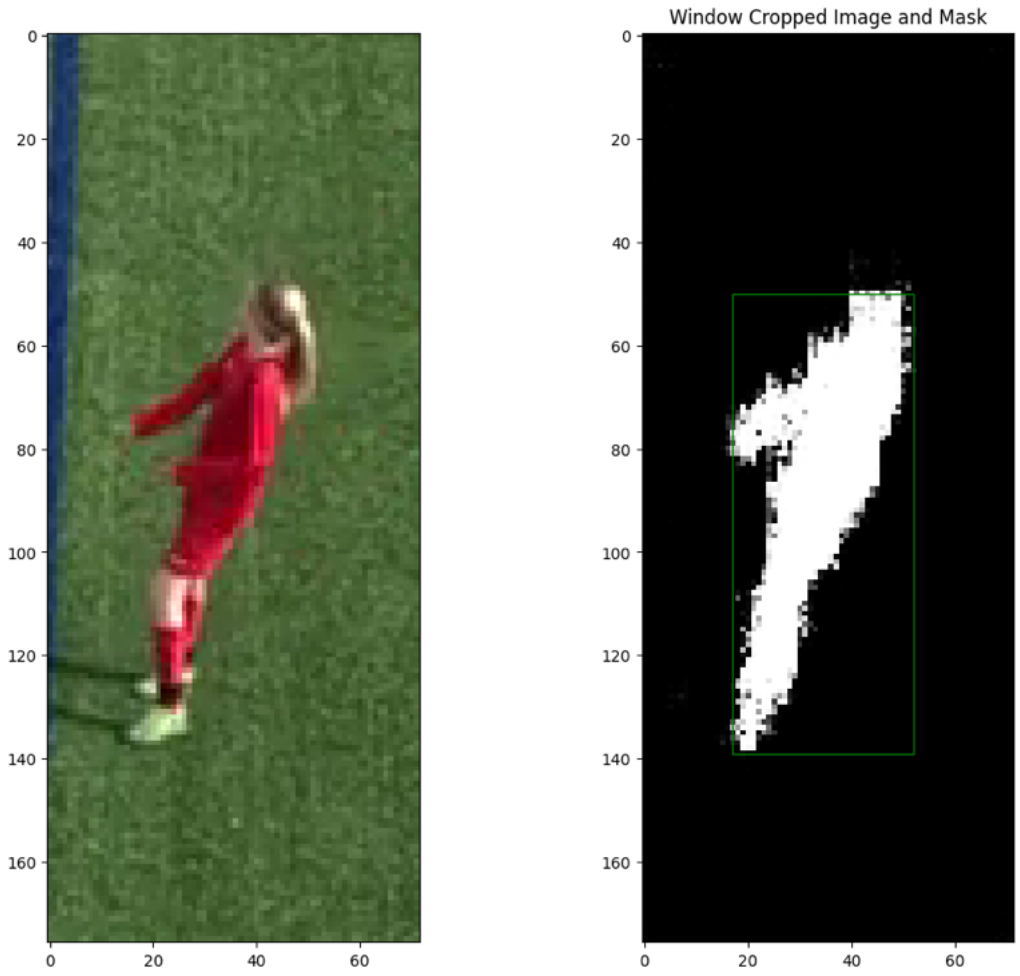

Figure 6.5: Sample 2 of Visual Player Tracking

Figure 6.6: Sample 2 of Visual Player Tracking, Full Frame

6.2.3 Evaluation Metrics

The visual object tracker is a key component of our system, responsible for accurately identifying and tracking players throughout the duration of the match. To evaluate its performance, we used two primary metrics: the average Intersection over Union (IOU) of bounding boxes against the ground truth, and the average number of frames before tracking failure. This section presents the evaluation methodology, results, and visual comparisons.

Intersection over Union (IOU)

IOU is a measure of the overlap between the predicted bounding box and the ground truth bounding box. It is defined as:

$$\text{IOU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

For a predicted bounding box B_p and a ground truth bounding box B_{gt}, the IOU is calculated as:

$$\text{IOU} = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}}$$

An IOU of 1 indicates a perfect overlap, while an IOU of 0 indicates no overlap. The average IOU provides a measure of how accurately the tracker is following the players.

Average Number of Frames Before Failure

This metric measures the robustness of the tracker by calculating the average number of frames the tracker successfully follows a player before a failure occurs. A failure is defined as the point where the tracker loses the player or the bounding box deviates significantly from the ground truth.

6.2.4 Evaluation Results

The visual object tracker was evaluated on the tracks of 18 players, and the results are summarized in Table 6.1.

Metric	Value
Average IOU	0.51
Average Number of Frames Before Failure	1486

Table 6.1: Evaluation Results of the Visual Object Tracker

6.2.5 Discussion

The visual object tracker demonstrates moderate accuracy with an average IOU of 0.51. This indicates that, on average, the tracker maintains a reasonable overlap with the ground truth bounding boxes. The robustness of the tracker is evidenced by the average number of frames before failure, which stands at 1486. This suggests that the tracker can follow players for a significant duration before losing track.

The visual comparisons further illustrate the performance of the tracker. In some cases, the tracker closely follows the ground truth bounding boxes, while in others, deviations are observed. These results highlight areas for improvement, such as enhancing the tracker's ability to handle occlusions and abrupt movements.

Overall, the evaluation of the visual object tracker provides valuable insights into its performance, guiding future improvements and optimizations to enhance accuracy and robustness.

6.3 Evaluating the Path Prediction Module

The path prediction module is a crucial component of our system, as it forecasts the future positions of players based on their past trajectories. To evaluate its performance, we tested the module against 500 testing samples using several metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Cosine Similarity Across Frames. This section presents the evaluation methodology, results, and visual comparisons of the predicted paths against the actual paths.

6.3.1 Evaluation Metrics

To assess the accuracy and reliability of the path prediction module, we employed the following metrics:

Mean Squared Error (MSE)

MSE measures the average of the squares of the errors, i.e., the average squared difference between the predicted and actual values. It is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i are the actual values and $\hat{y}_i$ are the predicted values.

Mean Absolute Error (MAE)

MAE measures the average of the absolute errors, i.e., the average absolute difference between the predicted and actual values. It is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Absolute Percentage Error (MAPE)

MAPE measures the average absolute percentage error between the predicted and actual values, providing a normalized measure of prediction accuracy. It is defined as:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

Cosine Similarity Across Frames

Cosine similarity measures the cosine of the angle between two vectors, in this case, the actual and predicted position vectors across frames. It is defined as:

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} y_i \cdot \hat{y}_i}{\sqrt{\sum_{i=1}^{n} y_i^2} \cdot \sqrt{\sum_{i=1}^{n} \hat{y}_i^2}}$$

This metric helps evaluate how close the angle of the predicted positions is to the actual positions as we progress further into the 64 predicted frames.

6.3.2 Evaluation Results

The path prediction module was evaluated on 500 testing samples, and the results are summarized in Table 6.2.

Metric	Value
MSE	0.00023
MAE	0.00813
MAPE	3.5934%
Cosine Similarity	See Figure 6.7

Table 6.2: Evaluation Results of the Path Prediction Module

6.3.3 Cosine Similarity Across Frames

To further analyze the prediction performance, we plotted the cosine similarity of predicted positions against the actual positions over 64 frames. Figure 6.7 shows how the cosine similarity changes as we predict further into the future.

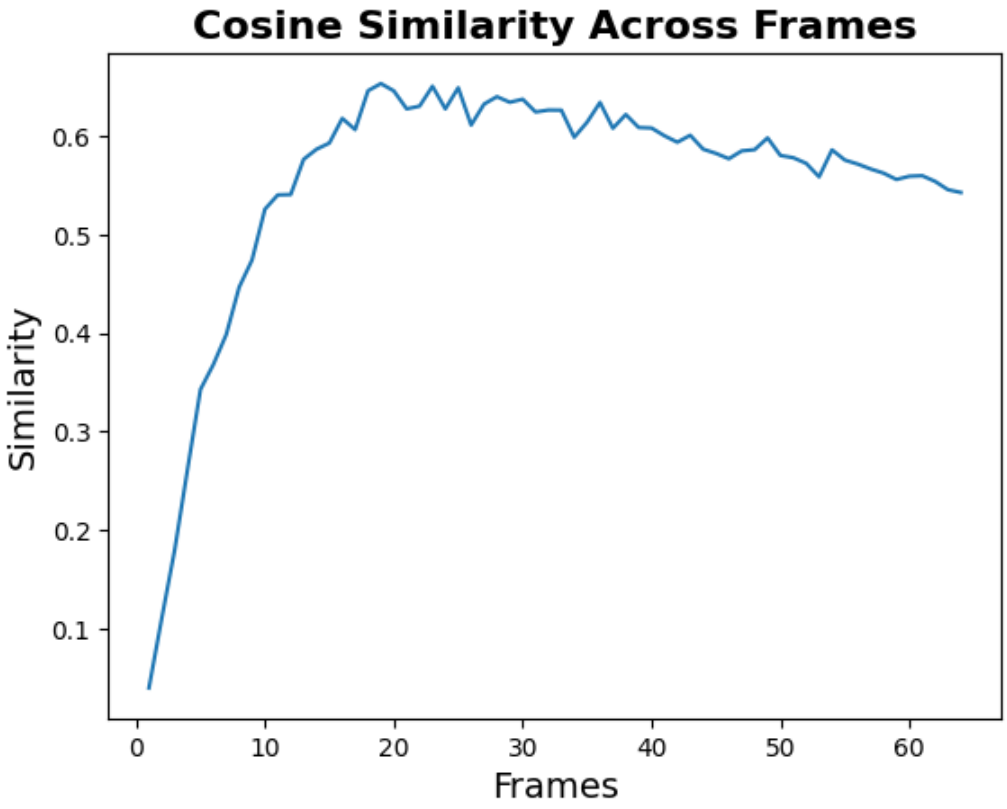

Figure 6.7: Cosine Similarity Across Frames: Evaluates the angular closeness of predicted positions to actual positions over 64 frames.

6.3.4 Visual Comparisons

To provide a visual understanding of the path prediction module's performance, we attached some visual results comparing the input paths (in orange), predicted paths (green), and actual paths (blue). Figures 6.8, 6.9, and 6.10 illustrate three different examples of path predictions.

6.3.5 Discussion

The path prediction module demonstrates high accuracy with an MSE of 0.00023 and an MAE of 0.0081. The low MAPE value of 3.59% indicates that the predictions are generally very close to the actual values. The cosine similarity analysis shows that the predicted positions maintain a high angular closeness to the actual positions over the 64 frames, indicating consistent and reliable predictions.

These results validate the effectiveness of the path prediction module and highlight its potential for use in real-time applications where accurate and timely predictions of player positions are crucial. The visual comparisons further reinforce the quantitative metrics, providing clear evidence of the module's ability to track and predict player movements accurately.

Figure 6.8: Path Comparison 1: Input path, predicted path, and actual path.

Figure 6.9: Path Comparison 2: Input path, predicted path, and actual path.

Figure 6.10: Path Comparison 3: Input path, predicted path, and actual path.

6.4 Evaluating the Similarity Module

The evaluation of the similarity model was conducted using a comprehensive set of metrics to ensure its robustness and reliability. The key metrics used for the evaluation include precision, recall, f1-score, and support for each class, along with accuracy, macro average, and weighted average. Additionally, a confusion matrix was generated to provide a visual representation of the model's performance across different classes.

6.4.1 Precision, Recall, and F1-Score

Table 6.3 presents the precision, recall, f1-score, and support for each class in the similarity model. The precision metric measures the model's ability to correctly identify positive instances, recall measures the ability to find all positive instances, and f1-score is the harmonic mean of precision and recall. Support indicates the number of true instances for each class.

Class	Precision	Recall	F1-Score	Support
0	0.79	0.82	0.81	6939
1	0.63	0.50	0.56	7661
2	0.81	0.99	0.89	10832
3	0.49	0.48	0.49	5322
4	0.73	0.97	0.83	2194
5	0.76	0.56	0.65	4368
6	0.60	0.98	0.75	1859
7	0.90	0.87	0.88	2734
8	0.97	0.94	0.95	7152
9	0.93	0.77	0.84	5984
10	0.91	0.86	0.88	4217
Accuracy		0.78 (59262)		
Macro Avg	0.78	0.79	0.78	59262
Weighted Avg	0.79	0.78	0.78	59262

Table 6.3: Precision, Recall, F1-Score, and Support for Each Class

6.4.2 Confusion Matrix

The normalized confusion matrix, depicted in Figure 6.11, provides a detailed breakdown of the model's performance by showing the number of correct and incorrect predictions for each class. The diagonal elements represent the number of correct predictions, while the off-diagonal elements indicate misclassifications.

6.4.3 Overall Performance

The similarity model achieved an overall accuracy of 78%, with a macro average precision, recall, and f1-score of 0.78, 0.79, and 0.78, respectively. The weighted average values for precision, recall, and f1-score were also consistent at approximately 0.78. These results demonstrate the model's strong performance across various classes, particularly for classes 2, 4, and 8, which exhibited high precision, recall, and f1-score values.

In conclusion, the similarity model has demonstrated strong performance through rigorous evaluation metrics. The detailed precision, recall, and f1-score metrics, along with the confusion matrix, provide a comprehensive understanding of the model's capabilities and areas for improvement. The visual representations further enhance the interpretability of the results, showcasing the model's effectiveness in real-world scenarios.

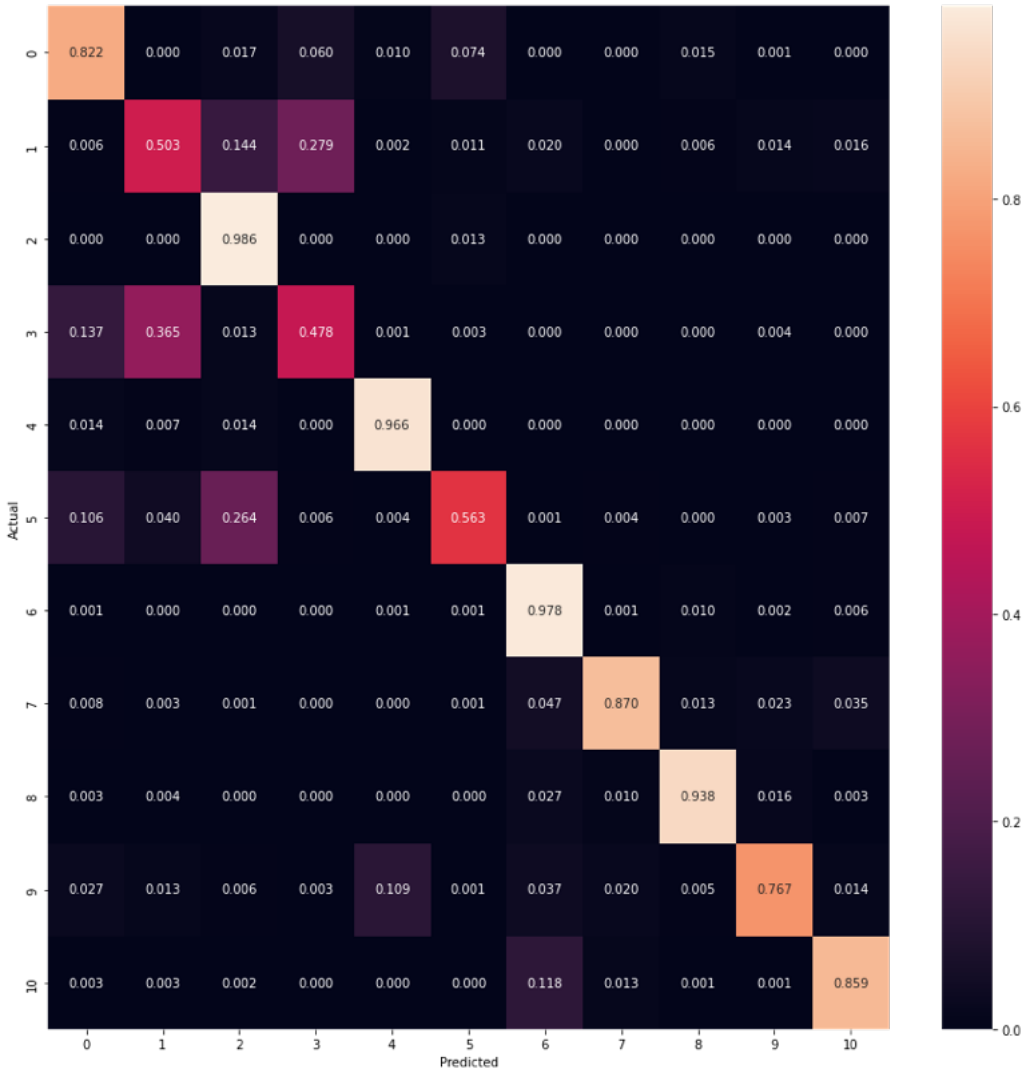

Figure 6.11: Normalized Confusion Matrix of the Similarity Model

Conclusion

In conclusion, the football analytics system presented here represents a groundbreaking approach to player tracking and analysis in the dynamic realm of football matches. The system's intricate architecture, characterized by a modular design, seamlessly integrates cutting-edge technologies to automate and enhance traditional manual tracking methods. The amalgamation of deep learning advancements with computer vision techniques propels the system beyond the limitations of labor-intensive processes, limited granularity, and occlusion issues.

Our primary goals of developing a modular system, providing accurate player statistics, creating a dynamic user interface, ensuring efficiency in tracking, and facilitating human-agent interaction have guided the design and development of each module. The integration of deep learning advancements and computer vision techniques has allowed us to overcome the limitations of manual tracking, offering a smart toolset for coaches, analysts, and football enthusiasts to gain deeper insights into the beautiful game.

The modular architectural design ensures flexibility, scalability, and efficient communication between components. Each module, whether it be the Object Detection spotlight operator, Visual Tracker persistent conductor, Path Prediction cautious prophet, Similarity dynamic re-identification framework, or Semantic Segmentation, plays a crucial role in the symphony of player tracking.

Given the computational complexity of real-time tracking and prediction, we implemented a distributed system architecture. This allowed us to parallelize the visual tracking module for each player, optimizing performance and ensuring timely processing of data. The stateless nature of the similarity and path prediction modules facilitated their efficient integration into the distributed framework. The system was designed with modularity and scalability in mind. Each component, from data ingestion and processing to tracking and prediction, was developed as an independent module, enabling seamless integration and easy maintenance. The use of APIs for data retrieval and processing further enhanced the system's flexibility and extensibility.

A critical aspect of the project was the development of an interactive web application to present the results. Built using Svelte, the web application provides a user-friendly interface for exploring the generated statistics and visualizations. Users can select specific matches and players to view their tracked positions and related statistics, allowing for detailed analysis of individual player performance and team dynamics. The web application offers a variety of visual representations, including heatmaps, diagrams, and interactive charts. These visuals

provide insights into player movements, ball possession, passes, player speeds, and distances covered, making the data easily interpretable. Both team and player statistics are displayed in a visually appealing manner, allowing users to explore comprehensive metrics such as possession percentages, pass completion rates, and speed profiles, among others.

Ultimately, this project serves as a testament to the collaborative power of interdisciplinary research and development. By leveraging the strengths of computer vision, deep learning, distributed computing, and web technologies, we have created a comprehensive system that transforms raw tracking data into actionable insights, paving the way for advanced sports analytics and beyond.

Bibliography

[1] Alex A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[2] Alexandre Alahi, Kshitij Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971. IEEE, 2016.

[3] David S. Bolme, John R. Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2544–2550, 2010.

[4] Liang-Chieh Chen, Yuheng Hu, and Geng Zhang. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision (ECCV)*, 2018.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] X. Chen, M. Treiber, V. Kanagaraj, and H. Li. Social force models for pedestrian traffic–state of the art. *Transportation Reviews*, 38(5):625–653, 2018.

[7] Valsamis Douskos, Ilias Kalisperakis, and George Karras. Automatic calibration of digital cameras using planar chess-board patterns. *Procedia Computer Science*, 3:1100–1106, 2007.

[8] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE, 2013.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

[10] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995.

[11] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765, 2016.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[13] H. Ishfaq, A. Hoogi, and D. L. Rubin. TVAE: Triplet-based variational autoencoder using metric learning. 2018.

[14] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[15] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[16] A. N. Kolmogorov and V. I. Arnold. On the realization of continuous functions of several variables by superposition of continuous functions of one variable. *Proceedings of the Steklov Institute of Mathematics*, 48:3–17, 1957.

[17] Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision*, pages 47–54, 2017.

[18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[19] Kevin Noto, Takuya Funatomi, and Michihiko Minoh. Soccernet: A scalable dataset for action spotting in soccer videos. *arXiv preprint arXiv:1902.01169*, 2019.

[20] Frank Pasquale. Toward a fourth law of robotics: Preserving attribution, responsibility, and explainability in an algorithmic society. *Ohio St. LJ*, 78:1243, 2017.

[21] Joseph Redmon and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[23] Daniel Ridel, Nikhil Deo, Dennis Wolf, and Mohan Trivedi. Scene compliant trajectory forecast with agent-centric spatio-temporal grids. *IEEE Robotics and Automation Letters*, 5(2):2816–2823, 2020.

[24] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1988.

[25] S. Samanta, S. O'Hagan, N. Swainston, T. J. Roberts, and D. B. Kell. VAE-SIM: a novel molecular similarity measure based on a variational autoencoder. *Molecules*, 25(15):3446, 2020.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[28] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[29] J. Xiang, Z. Huang, X. Jiang, and J. Hou. Similarity learning with deep crf for person re-identification. *Pattern Recognition*, 135:109151, 2023.

[30] Jianglong Ye, Yuntao Chen, Naiyan Wang, and Xiaolong Wang. Online adaptation for implicit object tracking and shape reconstruction in the wild. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[31] Shuai Yi, Hongsheng Li, and Xiaogang Wang. Pedestrian behavior understanding and prediction with deep neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision–ECCV 2016*, pages 263–279, Cham, 2016. Springer International Publishing.

[32] Chenjuan Yu, Xiaojie Ma, Jie Ren, Hengshuang Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *European Conference on Computer Vision*, pages 507–523. Springer, 2020.

[33] Yunming He Peng Chen Jing Tian Mingxiu Tang Lei Wang Ze Liu, Yutong Lin. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[34] et al. Zhang, Xiao. An enhanced swin transformer for soccer player re-identification. volume 14, page 51767. Nature Research, 2024.

[35] Wenjie Zhang, Hengrong Zhang, and Yang Yang. Learning multi-object tracking and segmentation from event cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9640–9649, 2019.