École Nationale Supérieure d'Informatique et d'Analyse des Systèmes - Rabat

Filière : Business Intelligence & Analytics

Basketball Free Throw Shooting Prediction using Pose Detection and GNNs

---

Élèves :
FENKOUCH Ayman
MOUDRIK Yahya

Encadrant :
Pr. Lamia Benhiba

Jury :
Pr. Lamia Benhiba
Pr. Si LHoussein Aouragh

Année Universitaire 2024-2025

Acknowledgements

First and foremost, we would like to express our sincere gratitude to our supervisor, **Pr. Lamia Benhiba**, for her continuous guidance, valuable insights, and unwavering support throughout this project. Her expertise in artificial intelligence and graph-based models greatly contributed to shaping the direction of our work.

We are also thankful to the jury members, **Pr. Lamia Benhiba** and **Pr. Si Lhoussein Aouragh**, for their time, feedback, and constructive evaluation of our work.

This project would not have been possible without the resources and infrastructure provided by the **École Nationale Supérieure d'Informatique et d'Analyse des Systèmes (ENSIAS)**. We also extend our gratitude to the **Business Intelligence and Analytics** program for equipping us with the tools necessary to pursue this line of research.

Finally, we thank our families, classmates, and friends for their encouragement and moral support throughout our academic journey.

Abstract

This project explores the use of deep learning and graph-based representations for predictive analysis in basketball, with a specific focus on free-throw efficiency. We propose a complete pipeline that leverages computer vision and graph neural networks (GNNs) to analyze player posture and predict the outcome of free-throw attempts. Videos of basketball players are processed using MediaPipe Pose to extract body landmarks and compute key joint angles. These features are then used to construct pose graphs, which are stored in Neo4j, while metadata such as video labels (hit/miss) are saved in MongoDB. The extracted graph sequences are used to train and evaluate several model architectures, including Graph Neural Networks (GNNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). The objective is to determine the most effective model for predicting shot outcomes and to assess the influence of player posture on shot success. Experimental results provide insights into the potential of graph-based posture analysis for performance evaluation in sports. This work contributes to the growing field of AI-assisted sports analytics and offers a foundation for real-time coaching tools and biomechanical performance assessments.

Keywords: Basketball Analytics; Graph Neural Networks; Pose Estimation; Shot Prediction; Deep Learning; Sports AI; MediaPipe; Neo4j.

Résumé

Ce projet explore l'utilisation de l'apprentissage profond et des représentations basées sur des graphes pour l'analyse prédictive dans le basketball, avec un accent particulier sur l'efficacité des lancers francs. Nous proposons un pipeline complet exploitant la vision par ordinateur et les réseaux de neurones graphiques (GNN) afin d'analyser la posture des joueurs et de prédire l'issue des tentatives de lancer franc. Les vidéos de joueurs sont traitées à l'aide de MediaPipe Pose pour extraire les repères corporels et calculer les principaux angles articulaires. Ces caractéristiques sont ensuite utilisées pour construire des graphes de posture, stockés dans Neo4j, tandis que les métadonnées telles que les étiquettes des vidéos (réussite/échec) sont sauvegardées dans MongoDB. Les séquences de graphes extraits servent à l'entraînement et à l'évaluation de plusieurs architectures de modèles, notamment les réseaux de neurones graphiques (GNN), les réseaux convolutifs (CNN) et les réseaux récurrents (RNN). L'objectif est d'identifier le modèle le plus performant pour la prédiction de l'issue des tirs et d'évaluer l'influence de la posture sur la réussite. Les résultats expérimentaux fournissent des perspectives sur le potentiel de l'analyse de posture basée sur les graphes pour l'évaluation de la performance sportive. Ce travail contribue à l'essor de l'analyse sportive assistée par l'IA et jette les bases d'outils d'entraînement en temps réel et d'évaluations biomécaniques.

Mots-clés : Analyse du basketball ; Réseaux de neurones graphiques ; Estimation de pose ; Prédiction de tir ; Apprentissage profond ; IA sportive ; Angles articulaires ; MediaPipe ; Neo4j.

Contents

List of Figures

List of Tables

List of Abreviations

- **GNN** : Graph Neural Network
- **GCN** : Graph Convolutional Network
- CNN : Convolutional Neural Network
- RNN : Reccurent Neural Network
- YOLO : You Only Look Once
- ST-GCN : Spatial-Temporal Graph Convolutional Networks
- LSTM : Long Short-Term Memory

Introduction

Basketball is a highly technical sport where subtle differences in movement, posture, and coordination can have a significant impact on performance. One of the most studied actions in basketball is the free throw, a seemingly simple shot taken under standard, uncontested conditions. Despite its apparent simplicity, the free throw involves complex biomechanics and nuanced posture control, making it a key area of interest for performance optimization and sports science research.

With the rapid advancement of artificial intelligence and computer vision, new opportunities have emerged to analyze athletes' movements using visual data. Pose estimation techniques—such as those provided by tools like MediaPipe, allow for the automatic extraction of body landmarks from video footage, offering objective and scalable ways to study posture. At the same time, graph-based methods and Graph Neural Networks (GNNs) provide a powerful framework for modeling spatial and biomechanical relationships between different parts of the human body.

In this project, we aim to leverage these technologies to model player posture during free throw attempts and to predict shot outcomes (success or failure). We developed an end-to-end pipeline that processes video footage, extracts body skeletons using MediaPipe Pose, calculates relevant joint angles, builds posture graphs, and stores these representations in a graph database (Neo4j). Metadata such as video name, timestamps, and labels (hit or miss) are managed in a document-based database (MongoDB).

The extracted graph-based data is then used to train and evaluate several classification models, including GNNs, Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs), to identify the most suitable approach for this analysis. In addition to measuring predictive performance, our study also investigates whether player posture has a measurable effect on shooting efficiency, contributing to a better understanding of biomechanical factors in basketball performance.

This research stands at the intersection of sports, artificial intelligence, and biomechanics, and aims to contribute to the development of intelligent performance analysis tools and real-time coaching systems.

The remainder of this document is organised into three core chapters followed by a general conclusion. **Chapter 1** surveys the state of the art: it reviews modern human-pose-estimation techniques, vision-based sports analysis and shot-outcome prediction, deep-learning models for motion recognition, recent advances in graph neural networks, their practical applications, and the open challenges that motivate our work. **Chapter 2** details the methodology, covering the overall research process, data collection and preparation, experimental environment, the full set of model architectures, and the evaluation metrics employed. **Chapter 3** presents the experimental results: it describes the technical pipeline, reports quantitative performance for every model, offers a global comparison, and discusses implications, limitations, and future directions. Finally, the **Conclusion** summarises the principal findings and outlines prospective extensions of this research.

Chapter 1

Related Works

Introduction

In this chapter, we explore the progression of research in pose estimation, computer vision-based sports analysis, and the application of deep learning models—especially graph-based architectures—for human motion understanding and athletic performance evaluation. Our focus is on literature related to basketball free throw analysis, movement recognition, and predictive modeling using pose data. The reviewed works span from foundational pose estimation systems to cutting-edge neural network models and practical applications in sports contexts.

1.1 Human Pose Estimation Techniques

Human pose estimation has seen significant advances with deep learning. Modern approaches leverage convolutional neural networks to detect key joint positions in images or video. OpenPose, introduced by Cao et al. [1], was a milestone as the first open-source real-time system for multi-person 2D pose detection. OpenPose uses a bottom-up strategy with Part Affinity Fields (PAFs) to associate detected body parts to individuals, achieving high accuracy in multi-person scenes. It can output up to 25 body keypoints (plus hands, feet, face) in 2D. MediaPipe Pose (MPP), developed by Google in 2019 [5], is another widely used framework. MediaPipe employs the BlazePose model to extract 33 body landmarks per frame and is optimized for real-time performance on mobile or CPU devices. Unlike OpenPose, MediaPipe can also infer 3D pose (depth) from a single camera, providing spatially detailed pose data. Other frameworks exist as well – for example, PoseNet focuses on lightweight, fast estimation (at the cost of fewer keypoints), and AlphaPose (Regional Multi-Person Pose) emphasizes accuracy with a more computationally intensive pipeline. Comparative studies have found OpenPose and MediaPipe to offer an excellent balance of accuracy and speed, making them preferable for real-time applications over alternatives like PoseNet or AlphaPose. PoseNet is extremely fast but lacks the rich joint detail, while AlphaPose achieves high accuracy but is less efficient for real-time use. Consequently, OpenPose and MediaPipe are frequently chosen in sports and movement analysis due to their reliability, coverage of key joints, and scalability.

1.2 Vision-Based Sports Analysis and Shot Outcome Prediction

Computer vision techniques have been applied increasingly in sports to analyze athlete motion, recognize actions, and even predict performance outcomes. In basketball, where shooting technique is critical, pose estimation enables quantitative analysis of a player's form. Prior sports science research has identified biomechanical indicators of a good shot – for instance, the "three 90 degrees" shooting form (having the wrist-forearm, forearm-upper arm, and upper arm-torso angles all at 90° during the shot preparation) is considered more stable and likely to succeed. However, identifying such pose subtleties by eye is difficult for players and coaches. Vision-based systems address this by extracting a shooter's joint angles and posture for each attempt.

Recent studies have specifically attempted shot outcome prediction from pose data. Li and Hua (2024) [2] developed a model to predict the success rate of stationary basketball shots (e.g. free throws) by combining improved object detection and pose estimation algorithms. They optimized a YOLOv5 detector (to track the ball or player) and enhanced an OpenPose-based pose estimator for accuracy. Their results showed high predictive performance: the model achieved an ROC-AUC of 0.974 and over 95% accuracy in classifying made vs. missed shots. This demonstrates that pose features can effectively indicate shot success. The authors note such prediction systems can provide useful insights for tactical analysis and player performance evaluation during games.

Beyond outcome prediction, pose-based analysis has been used to give feedback on technique. For example, Chen and Lu (2022) [3] aligned 3D pose sequences of a "student" shooter with a "teacher" (expert) shooter to pinpoint differences and offer frame-by-frame recommendations for improving the student's form. In their approach, dynamic time warping was applied to time-align the pose trajectories, enabling a direct comparison of key angles throughout the shot motion. Such applications illustrate the potential of pose estimation in sports training – by quantifying an athlete's movement, systems can detect deviations from ideal form and suggest corrections.

Pose estimation and action recognition have also been utilized in other sports contexts. There are works on classifying sports actions (e.g. identifying a tennis swing vs. a basketball free throw) from pose sequences, and even detecting subtle events. For instance, some projects combine player pose with ball tracking to detect when a shot is taken and whether it scores [6][12]. Overall, vision-based analysis in sports has enabled automatic posture recognition, skill assessment, and outcome prediction that were traditionally done via coaching observation or sensor-based motion capture.

1.3 Deep Learning Models for Human Action and Motion Recognition

Early approaches to action recognition from human movement data often employed sequential models. Recurrent neural networks, especially LSTM (Long Short-Term Mem-

ory) networks, have been widely used to model time-series of human joint coordinates [9]. For example, the introduction of large skeleton motion datasets (like NTU RGB+D) spurred the development of deep LSTM models that could learn temporal dynamics of joint movements for action classification. These RNN-based methods treat the pose sequence as a series of joint position vectors over time, and the LSTM's gating mechanism can capture motion patterns like the rhythm of a jump or the coordination of limbs in a shooting action. However, a limitation of basic RNN/LSTM approaches is that they do not inherently exploit the spatial relationships between body parts – each joint trajectory is modeled largely independently, which can ignore the skeletal structure. They also can be difficult to interpret in terms of which joint movements are most important to the action outcome.

In parallel, convolutional neural networks (CNNs) have been applied to pose-based tasks in various ways. CNNs excel at extracting hierarchical features from images, which made them a natural fit for early pose estimation models (e.g. the CNN backbone in OpenPose). Additionally, researchers have represented skeleton motion data in image-like formats (for instance, "skeletal images" where joint positions or heatmaps over time are encoded as channels) so that 2D or 3D CNNs can process them. CNN architectures are computationally efficient due to weight sharing and have been a preferred choice for vision tasks such as image classification, object detection, and even pose-based activity recognition. For example, Lin, Zhang, and Zhou (2023) generated 2D skeleton stick-figure images from exercise videos and successfully used pre-trained CNNs (like ResNet, VGG) to classify different rehabilitation exercises by posture [4]. Similarly, some systems feed per-frame pose coordinates into a CNN or a temporal convolutional network to learn motion features. While CNN-based approaches can capture patterns, they may require careful design (e.g. ordering of joints in an "image" tensor) to respect the body's topology.

Overall, CNNs and RNNs formed the first wave of deep learning for action recognition and movement classification. They proved effective for tasks like gesture recognition and simple action classification. Yet, as the field progressed, researchers sought architectures that could more explicitly leverage the human body's skeletal graph structure. This gave rise to Graph Neural Network models for human motion.

1.4 Graph Neural Networks for Pose-Based Action Recognition

Graph Neural Networks (GNNs) have emerged as a powerful approach to model human motion by treating the body pose as a graph of connected joints. In a skeleton graph, each node corresponds to a human joint (e.g. shoulder, elbow, knee), and edges represent natural connections or interactions (e.g. bones connecting joints, or even logical connections like left-right symmetry). This representation preserves the biomechanical structure of the body, allowing learning algorithms to propagate information along the limbs and torso in a human-informed way. Yan et al. (2018)[8] introduced the Spatial-Temporal Graph Convolutional Network (ST-GCN), a seminal model that applies graph convolutions over the skeleton's spatial structure at each time step, as well as standard convolutions over time. The ST-GCN automatically learns both spatial patterns (which joints move to-

gether, how) and temporal patterns (movement dynamics) from data, instead of relying on manually crafted features or traversal order[8]. This led to significantly improved performance and generalization on action recognition benchmarks, outperforming the earlier CNN/LSTM-based methods.

Following ST-GCN, many extensions were proposed: for instance, Two-Stream Adaptive GCNs that learn the graph connectivity or apply attention to important joints [15], and Graph Convolutional LSTMs that combine recurrent units with graph convolutions[9]. These efforts all indicate the advantage of marrying the skeleton's graph structure with deep models.

In sports and movement analysis, GNN-based techniques have shown promise in both classification and regression tasks. Liu, R., Hu, Y., & Zhang, T. (2023) integrated a graph neural network with a high-resolution pose estimation model (HRNet) to recognize athletic actions from pose data [11]. The graph-based model effectively captured limb movement patterns, improving accuracy and efficiency in identifying athletes' motions during games. By modeling the skeletal data as a fixed topology graph and extracting spatio-temporal features via graph convolutions, their approach could accurately distinguish different sports movements and provide an effective reference for analyzing athletic technique.

GNNs have also been used beyond action classification, for example, to predict continuous performance metrics. Yang et al. (2025) proposed a hybrid GNN with a gated recurrent unit (GRU) to predict the speed of a baseball pitch from the pitcher's motion data [7]. In their study, nodes represented key joints involved in pitching, and the GNN-GRU model learned how the coordination of these joints influences the throw velocity. This approach significantly outperformed a baseline LSTM network in prediction accuracy (achieving lower error and higher R^2). Moreover, because the model was graph-based, the authors could apply layer-wise relevance propagation to interpret which joints contributed most to the speed outcome – a valuable feature for providing biomechanical insights (e.g. confirming that arm and torso motions were most influential for pitch speed). One finding was that while the GNN-GRU was more accurate, it was also more sensitive to variability in the input data, suggesting that complex graph models might require careful generalization strategies on limited datasets. This underscores a general point: GNNs bring power and interpretability, but one must manage their complexity to avoid overfitting when data are scarce.

1.5 Applications in Athletic Performance Tracking and Biomechanical Feedback

The convergence of pose estimation and advanced neural networks has opened up new possibilities for athletic training and performance analysis. A number of AI-driven coaching systems have been developed in both research and commercial domains. These systems leverage pose data to give feedback on an athlete's form, much like a human coach or a motion capture lab would, but with the convenience of a camera-based setup. For instance, in the fitness domain, pose estimation is used in mobile apps and virtual trainers to guide users during exercises (yoga, weightlifting, physical therapy, etc.). An AI fitness

coach can automatically detect a user's body keypoints and evaluate their movement – identifying if a push-up's angle is wrong or if a squat's depth is insufficient – thereby providing corrective feedback in real time. This approach eliminates the need for wearable sensors or in-person supervision, making training more accessible.

In basketball, a standout example is the HomeCourt app, which uses computer vision on a smartphone to analyze shooting practice [12]. HomeCourt's advanced shot analysis can track each shot a player takes on camera and give instantaneous feedback on multiple performance metrics. Specifically, it measures a shooter's release time (how quickly the ball is released), release angle of the shot, the vertical jump height, the ball speed, and even the player's leg angle at launch. It can also automatically categorize the type of shot (free throw, jump shot off the dribble, etc.) and count makes or misses. Such real-world applications demonstrate the level of detailed biomechanical feedback achievable with pose estimation: players can review whether their release angle was optimal or if their legs were properly bent, and adjust their technique accordingly. According to coaches, having these quantitative insights – which are hard to discern with the naked eye – enables athletes to train smarter and develop better shooting habits faster.

Beyond individual training, pose-based analysis tools are being explored for team sports strategy and injury prevention. By tracking players' movements in games or practices, coaches could evaluate fatigue (from deteriorating form), identify risky biomechanics that might lead to injury, or study the form of elite players as a reference for others. In research settings, vision-based pose tracking has been validated against marker-based motion capture for sports movements to assess its accuracy [13]. While not yet as precise as laboratory systems, the advantage of video pose estimation is that it can be used in real time, in the actual sporting environment, without attaching sensors to athletes. This makes it feasible to capture genuine performance data during competition or natural training. As a result, there is growing interest in using these techniques for athletic performance tracking, providing coaches and sport scientists with a continuous stream of movement data to analyze player performance and health.

1.6 Challenges in Pose Estimation and Outcome Prediction

Despite the progress, several challenges persist in pose-based sports analytics:

- **Pose Estimation Accuracy and Occlusions:** Achieving high precision in keypoint detection during athletic motions can be difficult. Sports movements are often fast and involve complex body contortions, which can lead to partial occlusion of joints. For example, a basketball player's shooting arm might obscure parts of their torso or the other arm from certain camera angles. Occlusions are a well-known issue, limbs may become hidden behind the athlete's body or another player, causing pose estimators to drop or misplace those keypoints. Additionally, extreme body poses (like an airborne layup with legs bent) may fall outside the distribution of poses the model was trained on, lowering accuracy. This is compounded by motion blur in fast actions and varied lighting or camera viewpoints in sports

arenas. Research continues on improving robustness, with some methods introducing occlusion-aware modules or multi-view camera systems to better handle these cases. The reliability of the pose data is crucial because any errors directly impact the quality of outcome predictions or feedback derived from them.

- **Data Scarcity and Annotation:** Quality training data for sports-specific pose analysis is limited. Large public pose datasets (COCO, MPII) mostly consist of everyday or generic human poses, not athletes executing specific techniques. For a task like free throw outcome prediction, one would ideally need a substantial labeled dataset of shooting sequences with ground truth labels (make or miss). However, such datasets are scarce or non-existent in public domains. Chen and Lu (2022) [3] noted the lack of any labeled basketball shooting pose dataset, which forced them to collect their own clips from YouTube for a small-scale study. Annotating sports videos with ground truth (like success/failure or scores) is time-consuming and often requires expert knowledge (e.g., identifying what constitutes "proper form"). This data scarcity hinders the training of robust models and often necessitates techniques like transfer learning or data augmentation. It also makes it hard to cover the wide variability in how different athletes perform the same action.

- **Graph Construction from Human Pose:** When using graph-based models (GNNs) on pose data, one challenge is deciding how to construct and parameterize the skeleton graph. The human body has a natural kinematic graph (based on bones connecting joints), which most methods use as a baseline. However, there may be additional relationships (like left-right limb coordination, or joint interactions that are not directly connected by anatomy) that could be encoded. Some works keep a fixed manual graph structure, while others have proposed learning the graph topology adaptively from data. Each approach comes with trade-offs: a fixed graph is straightforward and grounded in anatomy, but a learned graph might capture action-specific connections (at the risk of overfitting). Defining the graph also involves setting edge weights or distances (some models treat all connected joints equally; others weight edges by physical distance or mobility). Moreover, representing the temporal aspect as part of the graph (as in ST-GCN's spatial-temporal graph) [8] can be complex. Careful design or learning of the graph is necessary to ensure important joint dependencies for the task (e.g., synchronization of shoulder, elbow, and wrist for shooting) are represented. The success of skeleton-based GNNs in action recognition shows that a well-constructed graph can significantly enhance feature learning, but finding the optimal graph structure for a new task is often non-trivial.

- **Model Generalization:** Generalizing models to different players, contexts, or camera setups is another key challenge. A model trained on one group of athletes or a specific environment might not perform as well on others. Differences in body shapes, skill levels, and individual technique can introduce distribution shifts. For instance, an algorithm learning to predict free throw success from one team's practice videos might latch onto idiosyncratic features (like a particular player's shooting quirk) that don't transfer to other players. Moreover, if the background or camera

angle changes (say, moving from a side view to a front view of the shooter), 2D pose coordinates can change in scale or orientation, affecting the model. Recent research in baseball pitch prediction found that while a GNN-based model was more accurate than an LSTM on the test set, it was also more sensitive to variability in the input data [7] – suggesting the GNN had learned very specific correlations that might not hold universally. This highlights the need for strategies like data augmentation, cross-subject training, or domain adaptation to improve generalization. Ensuring that a pose-based model captures fundamental motion features (and not just player-specific traits or recording conditions) is crucial for it to be broadly useful. Future work is focusing on making these models more robust – for example, by training on diverse datasets (when available), normalizing poses to a common view or scale, and using techniques such as regularization or meta-learning to avoid overfitting to narrow patterns.

Conclusion

In summary, the literature shows a clear evolution in basketball free throw (and general sports) outcome prediction: from improving the raw pose detection (using frameworks like MediaPipe/OpenPose) to deploying advanced models (CNNs, RNNs, and now GNNs) that can analyze those poses for actionable insights. While significant progress has been made in recognizing and evaluating athletic movements through computer vision, ongoing research continues to tackle the challenges of accuracy, data availability, graph modeling, and generalization to fully realize reliable free throw prediction systems. The related work in pose estimation, sports vision, and graph-based action recognition lays a strong foundation for developing a system that predicts free throw success from pose sequences while providing interpretable feedback on shooting form. All these insights will inform the design choices in our approach, from the selection of pose estimator to the architecture of the GNN and the strategies to train it effectively on limited basketball data.

Chapter 2

Methodology

Introduction

This chapter outlines the comprehensive methodology adopted to investigate the efficacy of Graph Neural Networks (GNNs) in analyzing basketball free throw performance using pose and motion data. The research is grounded in a comparative framework, benchmarking GNN models against conventional architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). We detail each step of the experimental pipeline, from data collection and preprocessing to model design, training, and evaluation. The objective is to establish a rigorous, reproducible foundation for exploring how spatial-temporal patterns in human motion can be leveraged for performance classification using advanced deep learning techniques.

2.1 Research Process

The objective of this research is to explore the effectiveness of Graph Neural Network (GNN) architectures for analyzing basketball free throw performance using posture and motion data. Traditional approaches such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are used as baselines to compare against advanced GNN models including Temporal Graph Convolutional Networks (TGCN) and GNNs augmented with attention mechanisms.

The research was conducted in the following stages:

1. **Problem Definition:** Formulate the task as a binary classification problem: determining whether a free throw results in a `hit` or a `miss`.
2. **Data Acquisition and Annotation:** Extract relevant video clips

containing free throw attempts and annotate each sample with a binary label.

3. **Pose Estimation and Feature Engineering:** Use MediaPipe to extract pose landmarks and compute angular features from the skeleton.
4. **Graph Construction and Data Formatting:** Construct temporal graphs for GNN input and sequences for RNNs. Format data to be suitable for deep learning frameworks.
5. **Model Design and Implementation:** Implement multiple models including baseline CNN/RNN and advanced GNN architectures with temporal and spatial attention.
6. **Training and Evaluation:** Train all models using consistent hyperparameters, and evaluate their performance using accuracy, recall, and F1-score.
7. **Comparison and Analysis:** Compare model performance and interpret results to assess the benefits of using GNNs for motion-based sports analytics.

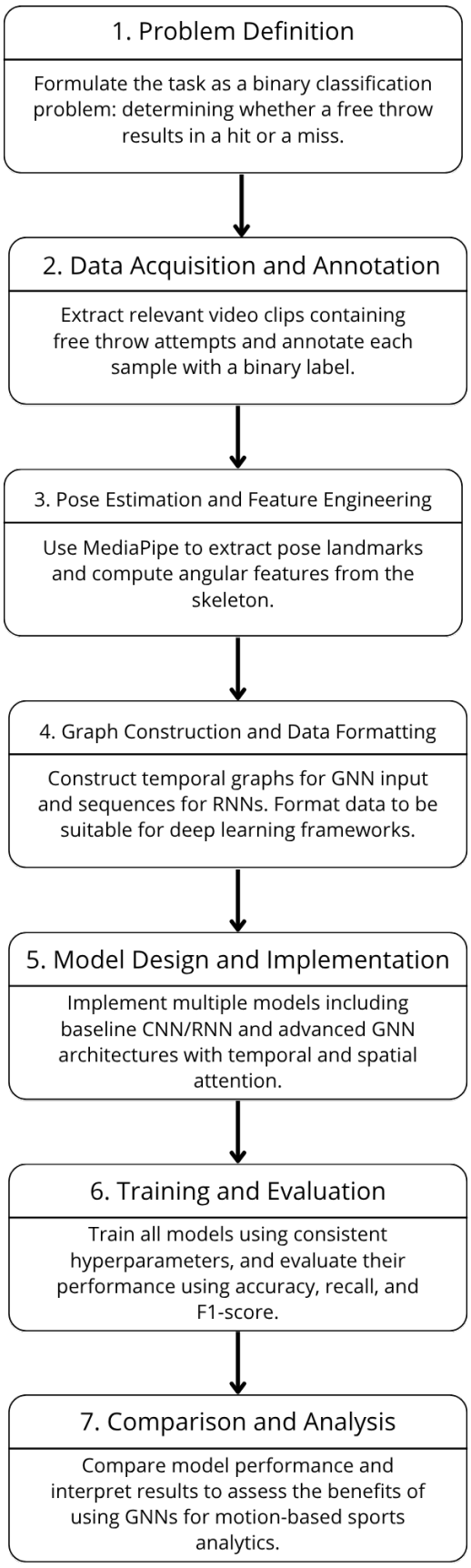

Figure 2.1: Figure illustrating the research process

2.2 Data

2.2.1 Data Collection

The dataset used in this project was constructed from a curated subset of the publicly available BASKET dataset [16], which contains general basketball gameplay footage. Since the dataset does not include labeled free throw attempts, a manual filtering process was applied to extract only those video segments where a player performs a free throw.

Figure 2.2: Exemple of Extracted Frame

The steps involved in collecting and labeling the dataset were:

- **Video Segmentation:** Full gameplay videos were divided into shorter clips focusing solely on free throw sequences using a video editing tool.
- **Labeling:** Each clip was manually labeled based on the outcome of the free throw, with `1` representing a successful shot (`hit`) and `0` representing a failed attempt (`miss`).
- **Final Dataset Composition:** A total of 90 labeled clips were collected and split between the two classes to ensure a dataset for training and evaluation.

- **Format:** The clips were saved in MP4 format and subsequently processed into individual image frames to enable model input.

To maintain consistency across the dataset, all clips were normalized in terms of resolution and frame count per sequence. This preprocessing ensured that models receive uniform input, reducing variability that could negatively affect learning performance.

2.2.2 Data Preparation

The dataset consisted of basketball free throw video clips clearly labeled as successful (`hit`) or unsuccessful (`miss`) based on their respective folder names. Two distinct preprocessing pipelines were employed depending on the targeted models.

CNN and Fine-tuned ResNet Models

For the CNN-based models, raw video clips stored in a shared Google Drive folder were preprocessed using OpenCV within Google Colab, following these steps:

- **Frame Extraction:** The first thirty frames from each video were extracted and saved to disk.
- **Color Space and Resolution:** Extracted frames were converted from BGR to RGB and resized to 224×224 pixels.
- **Normalization:** Frames were transformed into tensors and normalized using ImageNet's standard channel statistics.
- **Dataset Splitting:** A stratified split was conducted, assigning 70% of data for training and 30% for testing. No separate validation set was used as each model was trained for a fixed number of epochs.

GNN and RNN-based Models

For models capturing temporal and graph-based structures, additional preprocessing steps were required:

- **Pose Estimation:** Each extracted frame underwent pose estimation using MediaPipe Pose to obtain 33 skeletal joint coordinates per frame.
- **Angle Extraction:** Angles between relevant joint triplets (knees, hips, shoulders, elbows) were computed, providing temporal features for subsequent models.
- **Graph Construction:** Frames were transformed into graph representations:
  - Nodes represent joints annotated with position and angle features.
  - Edges represent anatomical connections and temporal links between consecutive frames.
- **YOLOv8 Fine-Tuning:** A YOLOv8 object detection model was fine-tuned on 2,086 annotated frames to isolate the main player. Training involved data augmentation techniques (horizontal flipping, rotation, brightness variations). Despite improved player detection, YOLOv8 exhibited difficulties accurately tracking the basketball, limiting its downstream effectiveness.
- **Frame Sampling:** A consistent number of frames per clip was sampled for uniformity across the dataset.
- **Normalization:** Extracted angles and joint coordinates were normalized to ensure stable and effective training.

Employing these dual preprocessing pipelines enabled optimized data input tailored to each specific model's architectural strengths.

2.3 Experimental Design

2.3.1 Experiment Environment

Two distinct environments were used depending on the model architecture:

Local Environment (GNN and RNN Models). The GNN and RNN-based models were trained on a local machine, which allowed full control over preprocessing, model customization, and graph-based experimentation without limitations from cloud resource quotas.

Google Colab (ResNet-50 and Simple CNN Models). To benefit from GPU acceleration, CNN-based models were trained on Google Colab using the free GPU infrastructure provided. The configuration was:

- **Platform:** Google Colab (cloud notebook environment)
- **GPU:** NVIDIA Tesla T4
- **RAM:** 12.67 GB
- **Software Stack:**
  - Python 3.10 (pre-installed)
  - PyTorch and torchvision
  - scikit-learn (for evaluation)
  - Google Drive integration for storing models and accessing data

Using this hybrid setup allowed us to efficiently utilize resources: cloud GPU for fast training on image-based models and local hardware for flexible experimentation with graph-structured data.

2.3.2 Models

This section outlines the architectures employed to classify basketball free throw attempts as either a `hit` or `miss`. The models range from traditional

Convolutional and Recurrent Neural Networks to advanced Graph Neural Networks incorporating spatial and temporal attention mechanisms.

CNN

Fine-tuned ResNet-50

Overview: ResNet-50 is a deep convolutional neural network composed of 50 layers and originally trained on ImageNet. In our work, we used transfer learning to adapt the model to the binary classification of basketball free throws (hit vs. miss).

Architecture: The architecture is based on residual learning. ResNet introduces identity shortcut connections that allow gradients to flow directly through earlier layers, solving the vanishing gradient problem in very deep networks.

- **Convolutional Stem:** The input image ($3 \times 224 \times 224$) is passed through a 7×7 convolution with stride 2, followed by batch normalization, a ReLU activation, and a 3×3 max pooling.
- **Residual Blocks:** The network contains four stages of residual blocks, where each block consists of:
  - 1×1 convolution (dimensionality reduction),
  - 3×3 convolution (processing),
  - 1×1 convolution (restoration),
  - Batch normalization and ReLU activations,
  - Identity shortcut connection (or projection if dimensions differ).
- **Global Average Pooling:** Reduces the spatial dimensions of the final feature maps to a single vector per channel.
- **Classifier Head:** The final fully connected (FC) layer was replaced with a new `Linear(in_features, 2)` layer to predict one of the two classes.

Training Details:

- All layers were kept trainable (full fine-tuning).
- Optimizer: Adam.
- Learning Rate: 0.001.
- Epochs: 10, Batch Size: 16.

Advantages: Leveraging pretrained features allows faster convergence and better generalization on small datasets.

Limitations: Heavy architecture ($\sim$25M parameters), slower training and inference compared to lightweight models.

Custom Simple CNN

Overview: To establish a lightweight baseline, we also designed a custom Convolutional Neural Network with fewer parameters.

Architecture:

- **Conv Block 1:**
  - 3×3 convolution with 32 filters.
  - ReLU activation.
  - 2×2 max pooling.
- **Conv Block 2:**
  - 3×3 convolution with 64 filters.
  - ReLU activation.
  - 2×2 max pooling.
- **Fully Connected Block:**
  - Flatten layer.
  - Fully connected (FC) layer with hidden units (unspecified but typically 128 or 256).
  - Dropout (rate = 0.5).
  - Output FC layer with 2 units (for binary classification).

Training Details: Same as for ResNet-50 (10 epochs, Adam, learning rate 0.001, batch size 16).

Advantages: Low number of parameters ($\sim$1M), faster inference, good performance despite simplicity.

Limitations: Lower representational capacity than ResNet; relies more on data quality.

RNN

Architecture: The BasketballRNN model utilizes an LSTM to capture temporal dependencies in the pose sequence data.

- **LSTM Layer:** Configurable input size, hidden size, and number of layers; dropout applied if more than one layer
- **Dropout Layer:** Dropout rate of 0.2
- **Fully Connected Layer:** Maps LSTM output to 2 classes

TGCN

Architecture: The GCN_LSTM model combines spatial feature extraction via a Graph Convolutional Network (GCN) with temporal modeling using an LSTM.

- **GCNConv Layer:** Applies graph convolution to each frame
- **Global Mean Pooling:** Aggregates node features
- **LSTM Layer:** Processes sequence of pooled features
- **Fully Connected Layer:** Outputs class probabilities

GNN Spatial Attention

Architecture: The GCN_LSTM_SpatialAttention model replaces the GCN with a Graph Attention Network (GAT) to focus on important nodes within each frame.

- **GATConv Layer:** Applies attention mechanism over graph nodes
- **Dropout Layer:** Dropout rate of 0.1

- **Global Mean Pooling:** Aggregates attended node features
- **LSTM Layer:** Captures temporal dynamics
- **Fully Connected Layer:** Outputs class probabilities

GNN Temporal Attention

Architecture: The GCN_LSTM_TemporalAttention model incorporates a temporal attention mechanism after the LSTM to emphasize significant time steps.

- **GCNConv Layer:** Extracts spatial features per frame
- **Dropout Layer:** Dropout rate of 0.1
- **Global Mean Pooling:** Aggregates node features
- **LSTM Layer:** Models temporal sequence
- **Temporal Attention:** Weighs LSTM outputs across time
- **Fully Connected Layer:** Outputs class probabilities

GNN Full Attention

Architecture: The GCN_LSTM_FullAttention model integrates both spatial and temporal attention mechanisms, along with attention pooling and layer normalization.

- **GATConv Layer:** Applies spatial attention
- **Layer Normalization:** Normalizes GAT outputs
- **Dropout Layer:** Dropout rate of 0.1
- **Attention Pooling:** Learns to weight node importance

- **LSTM Layer:** Processes temporal sequence
- **Layer Normalization:** Normalizes LSTM outputs
- **Temporal Attention:** Highlights important time steps
- **Fully Connected Layer:** Outputs class probabilities

2.4 Evaluation Metrics

To evaluate the performance of the models, we use the following standard classification metrics:

2.4.1 Accuracy

Accuracy measures the overall correctness of the model:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

2.4.2 Precision

Precision measures the proportion of correctly predicted positive observations to the total predicted positives:

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

2.4.3 Recall

Recall (also called sensitivity) measures the proportion of correctly predicted positive observations to all actual positives:

$$\text{Recall} = \frac{TP}{TP + FN}$$

2.4.4 F1-Score

The F1-Score is the harmonic mean of precision and recall, providing a balance between the two:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics allow us to evaluate not only how often our models are correct, but also how well they handle false positives and false negatives, which is crucial in sports analytics applications where errors can be costly.

Conclusion

In summary, this chapter has presented the end-to-end methodology implemented for evaluating basketball free throw attempts through a machine learning lens. The research integrates data acquisition from real-world videos, pose-based feature extraction, and graph construction, culminating in the application of both traditional and graph-based deep learning models. By establishing a clear and consistent experimental protocol—including model configurations, training environments, and evaluation metrics—this methodology sets the groundwork for meaningful comparisons and reliable analysis of the role of GNNs in sports performance classification. The results and interpretations derived from these models will be discussed in the subsequent chapters.

Chapter 3

Results

Introduction

This chapter presents the technical setup and experimental outcomes of our basketball free throw classification task. The study aims to evaluate the effectiveness of various deep learning models—including CNNs, RNNs, and Graph Neural Networks (GNNs)—in classifying free throw attempts as successful or failed based on pose data extracted from video. The pipeline integrates pose estimation, graph construction, graph storage in Neo4j, metadata handling in MongoDB, and deep learning training using PyTorch Geometric. We detail the implementation of baseline models and multiple GNN variants, highlighting how spatial and temporal dynamics of human motion impact performance. Evaluation metrics such as Accuracy, Recall, and F1-Score guide the comparative analysis.

3.1 Technical Architecture

3.1.1 Overview

The technical pipeline leverages pose estimation, graph construction, graph databases, document stores, and deep learning frameworks. The overall workflow is composed of the following key stages:

1. Pose estimation and frame sampling from video.
2. Graph construction from pose landmarks.
3. Feature extraction and transformation into GNN-ready format.
4. Graph storage in a **Neo4j** graph database.
5. Metadata and label storage in **MongoDB**.
6. Preparation for training with **PyTorch Geometric**.

3.1.2 Pose Extraction and Graph Construction

From each input video, frames are sampled at fixed intervals (`FRAME_SKIP`) to reduce redundancy. Each sampled frame is passed through a pose detection model (e.g., `MediaPipe`

`Pose`) to extract 3D landmarks (x, y, z, visibility). These landmarks are used to construct an undirected graph $G = (V, E)$ as follows:

- Each node $v_i \in V$ represents a body joint with features: position (x, y, z) and visibility.
- Edges $e_{ij} \in E$ are created according to predefined anatomical connections (e.g., shoulder-to-elbow), and are weighted using the Euclidean distance between joint coordinates.
- A subset of joints is used to compute angles (e.g., elbow, knee), which are stored as additional node features.

Figure 3.1: Mediapipe Posture Extraction

Nodes are remapped to have consecutive indices for compatibility with PyTorch Geometric. Two graphs are retained per timestep: the *original* with MediaPipe indices, and the *remapped* graph.

3.1.3 Graph Data Preparation for GNNs

To train Graph Neural Networks (GNNs), each graph is transformed into a format compatible with `PyTorch Geometric`:

- **Node features matrix** $X \in R^{N \times 4}$ where N is the number of nodes and each feature vector contains $[x, y, z, visibility]$.
- **Edge index** matrix `edge_index` $\in Z^{2 \times |E|}$ encodes the connections between nodes.

- **Edge attributes** $\in R^{|E|\times 1}$ correspond to distances between connected nodes.
- **Angle features** are stored in a separate matrix and linked to specific joints.
- Each frame's graph is associated with a label (1 = hit, 0 = miss).

3.1.4 Graph Storage in Neo4j

To support visualization and time-based analysis, each timestep's graph is stored in a **Neo4j** graph database:

- Each node is stored as a `PoseNode` with attributes: video ID, time index, node index, angle value, and timestamp.
- Edges are stored as `CONNECTED_TO` relationships with weights.
- A unique `video_id` (UUID) links all frames from the same video.

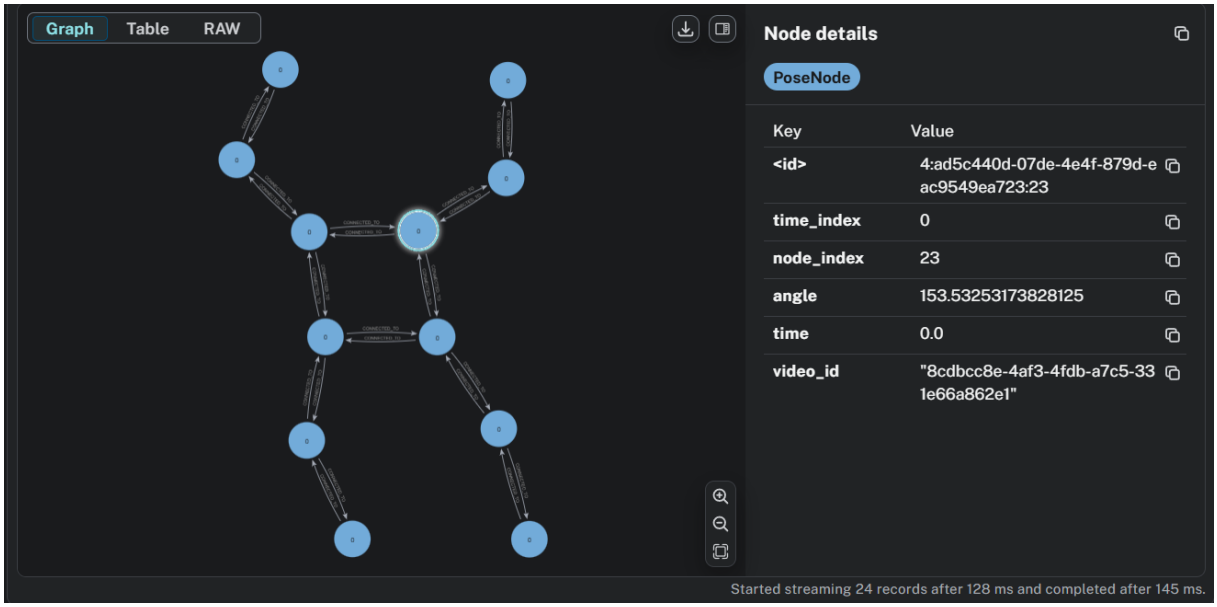

Figure 3.2: Neo4j Graph Example at a given Timestep

The graph schema supports efficient queries and traversal across time, enabling exploratory graph analytics and dynamic feature analysis.

3.1.5 Metadata Storage in MongoDB

MongoDB is used to store high-level metadata:

- `video_id`: the unique identifier of the video.

- `video_name`: the name of the processed file.
- `label`: the ground truth (1 = hit, 0 = miss).

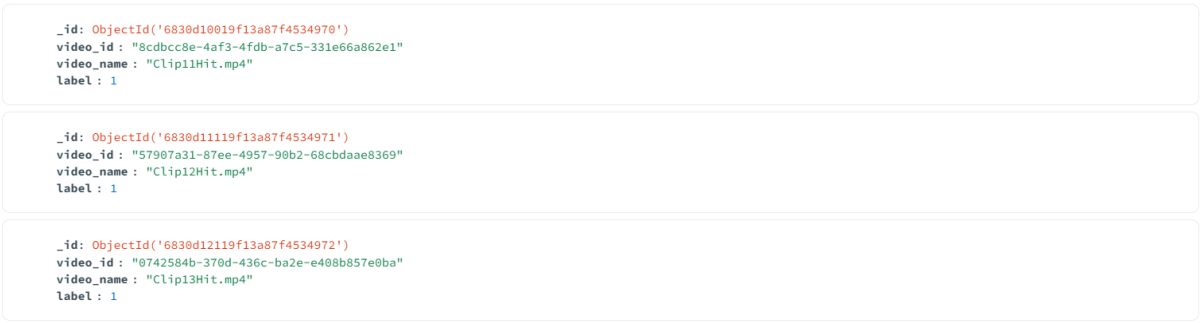

Figure 3.3: Videos Metadat in MongoDB

This setup decouples structured graph data from unstructured metadata, allowing flexible access and indexing.

3.1.6 Frameworks and Libraries

The technical pipeline integrates several tools and libraries:

- **MediaPipe**: for real-time 3D human pose estimation.
- **NetworkX**: for graph construction and manipulation.
- **PyTorch** and **PyTorch Geometric**: for deep learning and graph neural network training.
- **Neo4j**: for time-aware graph storage and querying.
- **MongoDB**: for storing labels and metadata.
- **OpenCV**: for frame extraction from video files.

3.2 Baseline Models

3.2.1 Convolutional Neural Network Models

Simple CNN

The custom lightweight CNN achieved remarkably strong performance given its simplicity:

- **Accuracy:** 0.9866

- **Recall:** 0.9690
- **F1-Score:** 0.9798

This model was built from scratch and consists of two convolutional blocks. Each block includes a 3×3 convolution layer followed by a ReLU activation and a 2×2 max pooling operation. The first block uses 32 filters, and the second uses 64. After the convolutional stages, the resulting feature maps are flattened and passed through a fully connected hidden layer. A dropout layer with a rate of 0.4 is applied before the final linear output layer, which has two units corresponding to the binary classification task.

Training was performed over 10 epochs using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The model was trained entirely in Google Colab using the provided GPU, and the best-performing model was saved to Google Drive for evaluation. Despite having approximately one million parameters, this model generalized exceptionally well, achieving high recall on the test set, meaning it correctly identified the vast majority of successful free throw attempts. Its high F1-score further confirms that this model strikes an excellent balance between precision and recall, making it a reliable baseline for the task.

The best performance was achieved using the following hyperparameters (obtained via grid search): `learning rate = 0.001`, `batch size = 32`, `num_filters = 64`, `dropout = 0.4`, `num_epochs = 10`.

Fine-tuned ResNet-50

The ResNet-50 model also delivered high performance, with slightly lower recall but comparable overall accuracy and F1-score:

- **Accuracy:** 0.9835
- **Recall:** 0.9619
- **F1-Score:** 0.9750

This model was initialized with pretrained weights from ImageNet and fine-tuned for the free throw classification task. Specifically, the final fully connected layer of ResNet-50 was replaced with a new linear layer containing two output units. All layers of the network were left trainable, allowing the model to adapt its deep features to the basketball domain.

Training followed the same procedure as for the Simple CNN — ten epochs, batch size of 16, and the Adam optimizer with a learning rate of 0.0001. Like the CNN, this model was trained using Google Colab and benefited from GPU acceleration. The model's deep residual architecture, consisting of convolutional layers and identity shortcuts, allowed it to learn complex visual representations of the free throw motion across frames.

Although it slightly underperformed compared to the Simple CNN in terms of recall, ResNet-50 demonstrated consistent robustness and precision. Its high F1-score reflects

that the model correctly predicted both hits and misses with balanced performance, despite the deeper architecture and higher number of parameters.

The best performance was achieved using the following hyperparameters (obtained via grid search): `learning rate = 0.0001`, `batch size = 16`, `num_epochs = 10`.

3.2.2 RNN

The RNN model, trained on sequences of joint angles over time, yielded significantly worse results:

- **Accuracy:** 0.7778
- **Recall:** 0.50
- **F1-Score:** 0.4375

Despite its temporal modeling capabilities, the RNN failed to capture the necessary spatiotemporal patterns for classification, particularly struggling with recall. This underperformance likely results from the model's inability to encode spatial context, a limitation further explored in the discussion.

3.3 Graph Neural Network (GNN) Models

3.3.1 TGCN (Temporal Graph Convolutional Network)

- **Accuracy:** 0.7778
- **Recall:** 1.00
- **F1-Score:** 0.8750

TGCN integrates temporal graph convolutions over the sequence of body joint graphs. While the accuracy is modest, its perfect recall and strong F1-score suggest it is effective at identifying successful shots, albeit at the cost of increased false positives.

3.3.2 Spatial Attention GNN

- **Accuracy:** 0.7407
- **Recall:** 1.00
- **F1-Score:** 0.8511

This model applies attention mechanisms across joints (nodes), allowing it to weigh more important joints (e.g., elbows or wrists) higher. It matches TGCN in recall and slightly underperforms in F1-score, suggesting similar behavior but potentially overfitting or noise sensitivity in spatial attention weights.

3.3.3 Temporal Attention GNN

- **Accuracy:** 0.8148
- **Recall:** 1.00
- **F1-Score:** 0.8980

This model outperformed other GNN variants. By focusing attention on specific frames, it isolates the key motion moments in the free throw, such as jump and release. The performance suggests that temporal saliency plays a more critical role than node-level spatial saliency.

3.3.4 Full Attention GNN (Spatial + Temporal)

- **Accuracy:** 0.7407
- **Recall:** 1.00
- **F1-Score:** 0.8511

Surprisingly, combining both spatial and temporal attention did not improve performance. The added complexity may have caused overfitting, especially given the limited training data, or introduced redundancy between already well-captured cues.

3.4 Global Comparison Summary

Model	Accuracy	Recall	F1-Score
Simple CNN (2D)	**98.66%**	**96.90%**	**97.98%**
ResNet-50 (Fine-tuned)	98.35%	96.19%	97.50%
RNN (Angles only)	77.78%	50.00%	43.75%
TGCN (Graph + Time)	77.78%	100.00%	87.50%
GNN with Spatial Attention	74.07%	100.00%	85.11%
GNN with Temporal Attention	81.48%	100.00%	89.80%
GNN with Full Attention	74.07%	100.00%	85.11%

Table 3.1: Comparison of all models on the basketball free throw classification task.

As summarized in Table 3.1, the two convolutional models: Simple CNN and fine-tuned ResNet-50, achieved the highest overall performance, both in terms of accuracy and F1-score. The Simple CNN obtained perfect recall and the best F1-score among all models. On the other hand, the RNN model showed the lowest performance across all metrics. Among graph-based models, the GNN with temporal attention achieved the strongest results, outperforming other GNN variants and the RNN. All GNN variants shared perfect recall but varied in precision, as reflected in their F1-scores. While GNNs and RNNs offer more structured inputs, their performance was generally lower than the CNN models trained on raw frame data in this setup.

3.5 Discussion

3.5.1 YOLOv8-Based Preprocessing

As part of the preprocessing pipeline, a YOLOv8 model was fine-tuned on 2,086 annotated frames to isolate the main player in each video. The approach was based on detecting the basketball first, then identifying the player closest to the ball as the shooter.

However, this pipeline faced several issues. The model frequently detected incorrect players or unrelated objects on the court, especially in crowded scenes or when the ball was partially occluded. These false detections introduced noise into the posture extraction process, often leading to pose estimates of non-shooting players or irrelevant backgrounds. As a result, the YOLOv8-based player detection was excluded from the final pipeline.

Given the central role of accurate player identification for downstream pose analysis, future work should explore more robust player tracking methods. This could involve training on larger annotated datasets, leveraging temporal consistency across frames, or using keypoint-based trackers to better associate poses with the correct subject.

3.5.2 CNNs vs. GNNs: Raw Frames vs. Structured Pose

CNN-based models — both the custom Simple CNN and the fine-tuned ResNet-50 — clearly outperformed all other architectures in terms of accuracy, recall, and F1-score. This strong performance was achieved despite the models operating directly on raw video frames, without any explicit pose or joint-level information.

This outcome highlights the ability of convolutional networks to implicitly learn discriminative motion and posture features from visual cues. In this specific task — with controlled camera angles, lighting, and consistent framing — raw pixel-based features appear sufficient for capturing the visual patterns that distinguish successful and failed free throws.

Another potential explanation for the superior performance of CNNs lies in their ability to process the entire visual scene, not just the posture of the shooter. Unlike GNNs and RNNs, which rely on pose estimation or joint angles, CNNs analyze raw video frames and can implicitly capture contextual cues such as the expressions or reactions of surrounding

players, ball trajectory, and even the overall scene composition. These subtle visual patterns — while not explicitly modeled — may correlate with shot success and influence the network's decision-making. This highlights the advantage of using raw visual input in scenarios where relevant information may extend beyond the primary subject.

Interestingly, the Simple CNN outperformed the deeper ResNet-50 model despite its simpler architecture. This may be attributed to the relatively small and homogeneous dataset, which favored a lower-capacity model that was less prone to overfitting. The task of distinguishing successful from failed free throws may not require the full representational power of a deep residual network. Furthermore, the Simple CNN was trained from scratch on data with consistent visual framing, allowing it to focus on the most relevant visual cues without being affected by pretrained weights from unrelated domains. These factors likely contributed to its superior generalization performance.

Conversely, GNNs, despite leveraging rich pose graphs and joint connectivity, were outperformed by CNNs. While graph-based models offer theoretical advantages in interpretability and spatial reasoning, they seem to require more data to learn meaningful patterns robustly.

3.5.3 GNNs vs. RNNs: The Value of Spatial Context

The RNN model, trained on sequences of joint angles alone, significantly underperformed compared to all other models. Its inability to model the spatial relationships between joints limits its capacity to learn meaningful shooting patterns. In contrast, GNNs explicitly represent the body as a graph, capturing both pose structure and temporal progression.

The performance gap between RNNs and GNNs confirms that incorporating spatial context — particularly the anatomical relationships between joints — is critical when analyzing athletic movement. While RNNs are effective at sequence modeling, they lack the structural inductive bias that GNNs naturally provide.

3.5.4 Intra-GNN Comparison: Temporal Attention as Key Driver

Among all graph-based models, the Temporal Attention GNN achieved the best performance. By dynamically focusing on the most important time steps in a shooting sequence (e.g., arm extension or ball release), the model could prioritize the most informative frames.

On the other hand, adding both spatial and temporal attention (Full Attention GNN) did not yield further gains and, in fact, resulted in lower performance compared to using temporal attention alone. This suggests that the added complexity may have introduced redundancy or overfitting due to the limited size of the dataset.

- The Temporal Attention GNN benefited from sequence-level discrimination.

- The Spatial Attention GNN was less effective in isolation, likely due to all joints receiving similar focus during repetitive movements.
- The Full Attention GNN may have been too expressive for the amount of available data, failing to generalize.

3.5.5 Model Explainability vs. Performance Trade-off

One of the main advantages of GNNs is their explainability. Attention weights (spatial or temporal) allow researchers and coaches to understand which joints or frames the model relied on for its predictions. This can provide actionable feedback in a sports training context, even if overall accuracy is slightly lower than CNNs.

In contrast, CNNs, despite their superior performance, remain harder to interpret. They act as black boxes with limited transparency regarding which parts of the image contributed most to the classification decision. Therefore, depending on the application (e.g., athlete development vs. automated analytics), one model family may be favored over the other.

3.5.6 Limitations and Future Work

While the results are promising, several limitations need to be acknowledged:

- **Limited Dataset Size:** The number of labeled free throw clips is small, which may have restricted the performance of deeper or more complex models like GNNs with full attention.
- **Pose Estimation Robustness:** Although MediaPipe performed well overall, accuracy declined under occlusions, fast movements, or non-standard poses.
- **Player Detection Failures:** One of the major challenges encountered was detecting the correct player in each frame. The YOLOv8-based method often detected irrelevant players or objects, especially in scenes with multiple athletes on court. This directly impacted pose extraction quality and model accuracy.
- **Single-Camera Input:** All models rely on monocular 2D video input. Depth ambiguity and perspective distortion can affect both pose estimation and visual feature extraction.

Figure 3.4: Example of Player Detection Failure

To address these challenges, future work may consider:

- Collecting a larger, more diverse dataset of free throws under different camera angles and lighting conditions
- Integrating multi-view or 3D pose estimation to enrich pose representations
- Developing hybrid models that combine CNN visual features with GNN structural reasoning
- Exploring explainability tools such as Grad-CAM for CNNs or visualization of temporal attention weights for GNNs
- Improving the player detection step by training models specifically to track the shooter across frames, potentially using ball trajectory analysis, motion patterns, or sequence-based association

3.5.7 Summary of Findings

The key insights derived from this study are summarized below:

- **Simple CNN outperformed all models**, achieving the highest accuracy and recall, despite having the simplest architecture and lowest number of parameters.
- **Fine-tuned ResNet-50 achieved strong performance** and demonstrated robustness in classification, validating the benefit of transfer learning on a small dataset.

- **Graph-based models (GNNs) performed better than RNNs**, especially when temporal attention was used to focus on key frames such as the shot release. However, GNNs were still outperformed by CNNs overall.
- **RNNs underperformed significantly**, indicating that relying solely on temporal joint angles without spatial context is insufficient for shot outcome prediction.
- **Temporal attention in GNNs proved more effective than spatial attention**, emphasizing the importance of frame-level focus in sports motion analysis.
- **Combining both spatial and temporal attention did not improve performance**, likely due to model overfitting and increased complexity relative to dataset size.
- **Player detection was a major bottleneck**. The YOLOv8-based shooter identification pipeline was unreliable, often misidentifying players or background elements, which negatively impacted the quality of pose data.
- **MediaPipe was reliable for pose extraction**, but its effectiveness was limited by upstream detection errors and single-camera input constraints.
- **CNNs offered superior accuracy**, while **GNNs offered interpretability**, presenting a trade-off between raw performance and explainable insights that must be considered based on the intended application.

Conclusion

The results demonstrate that both CNN-based image models and GNN-based pose graph models can be effective in free throw classification, but they differ significantly in behavior and application scope. The custom Simple CNN outperformed all models in overall performance, achieving the highest recall and F1-score, showing the power of spatial visual patterns. On the other hand, GNN-based models—especially the Temporal Attention GNN—offered perfect recall and showed promise in capturing temporal joint dynamics, though they slightly trailed in precision.

The RNN baseline confirmed the importance of spatial structure, as it underperformed when using only sequential joint angle data. Among GNN variants, attention-based models (temporal and spatial) outperformed simpler architectures by prioritizing key timeframes and joints, suggesting that integrating motion saliency significantly enhances performance.

Overall, the pipeline validates the feasibility of combining pose graphs and graph neural networks for sports analysis. It also establishes a foundation for future improvements, such as multi-shot sequence modeling, player-specific learning, and real-time feedback systems.

Conclusion

This project explored the intersection of computer vision, biomechanics, and machine learning in the context of basketball free throw analysis. The primary objective was to investigate how different model architectures, including convolutional neural networks, recurrent networks, and graph-based models, could be employed to predict the outcome of free throw attempts based solely on video data. The broader ambition was to understand the relationship between player posture and shot success, while also assessing the feasibility of using pose estimation pipelines for interpretable sports analytics.

A complete data pipeline was developed, starting from the collection and manual labeling of a curated set of basketball clips. Preprocessing strategies were implemented to extract relevant information from the raw video, including frame sampling, pose estimation using MediaPipe, joint angle computation, and graph construction. Although efforts were made to integrate YOLOv8 for improved player detection, the model struggled to consistently identify the correct shooter. This issue, often caused by the presence of multiple players or objects in the frame, significantly impacted pose quality and highlighted a key challenge in automated sports analysis.

Multiple models were trained and evaluated, including a simple custom CNN, a fine-tuned ResNet-50, an RNN, and several variants of Graph Neural Networks. Each model brought a different approach to interpreting the data. CNNs were applied to raw video frames, RNNs to sequences of joint angles, and GNNs to structured skeletal graphs. The evaluation metrics, including accuracy, recall, and F1-score, provided a balanced view of the models' classification capabilities.

Among all tested models, the Simple CNN achieved the highest performance, even surpassing the deeper ResNet-50 architecture. Its simplicity, combined with well-prepared input data, allowed it to learn effective representations

of shooting patterns without the need for explicit pose features. Graph-based models showed competitive results, particularly the temporal attention GNN, which effectively captured key motion sequences. However, they were generally outperformed by CNNs in terms of accuracy and overall reliability. The RNN model performed the worst, confirming that temporal data alone is insufficient without spatial context in tasks involving body movement.

In addition to numerical evaluation, this project provided key conceptual insights. CNNs demonstrated superior performance in controlled environments, but they operate as black-box models with limited interpretability. In contrast, GNNs, while slightly less accurate, offer clearer insight into how and why decisions are made by highlighting the importance of specific joints or time frames. This trade-off between performance and explainability is critical when considering real-world applications in sports coaching and athlete evaluation.

Several limitations were identified during the project. The dataset was relatively small and homogeneous, which may have limited the performance of deeper and more complex models. The pose estimation process, although effective in most cases, was sometimes compromised by occlusions, motion blur, or poor shooter identification. Moreover, all models relied on single-camera 2D input, which lacks depth information and can introduce distortions depending on the camera angle.

To address these issues, future work may involve expanding the dataset to include more players and diverse environments, using multi-view or depth-based pose estimation to improve joint accuracy, and developing hybrid architectures that combine the strengths of CNNs and GNNs. Additionally, improving the robustness of shooter detection would enhance pose reliability and downstream model performance.

In conclusion, this project successfully demonstrated that machine learning models can predict the outcome of basketball free throws using visual and structural information extracted from video data. It showed that simple

models can achieve high accuracy with minimal preprocessing, while more advanced models like GNNs offer interpretability and structure-aware learning. These findings represent a step forward in the development of automated sports analysis tools that are both effective and explainable, and open new possibilities for integrating artificial intelligence into performance monitoring and coaching in athletics.

References

1. Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2018). OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *arXiv preprint arXiv:1812.08008*. https://arxiv.org/abs/1812.08008

2. Li, Z., & Hua, Y. (2024). Basketball Fixed-point Shot Hit Prediction Based on Human Pose Estimation Algorithm. *Informatica*. https://www.informatica.si/index.php/informatica/article/view/5781

3. Chen, K., & Lu, H. (2022). Comparing student and expert pose trajectories in free-throw sequences. *CS231A Stanford Project*. https://web.stanford.edu/class/cs231a/prev_projects_2022/lCS231a_Project_Final_Report.pdf

4. Lin, X., Zhang, J., & Zhou, Y. (2023). A Comparison of Deep Learning Techniques for Pose Recognition in Up-and-Go Pole Walking Exercises Using Skeleton Images and Feature Data. *Electronics*, 14(6), 1075. https://www.mdpi.com/2079-9292/14/6/1075

5. Zheng, M., Zhao, B., & Jin, Y. (2023). Human Pose Estimation Using MediaPipe Pose and Optimization Method Based on a Humanoid Model. *Applied Sciences*, 13(4), 2700. https://www.mdpi.com/2076-3417/13/4/2700

6. Shah, A. (2023). Real-Time AI Basketball Shot Detection with YOLOv8 and OpenCV. GitHub. https://github.com/avishah3/AI-Basketball-Shot-Detection-Tracker

7. Yang, W., Zhou, T., & Zhang, X. (2025). Leveraging Graph Neural Networks and Gated Recurrent Units for Accurate and Transparent Prediction of Baseball Pitching Speed. *Scientific Reports*. https://www.nature.com/articles/s41598-025-88284-x

8. Yan, S., Xiong, Y., & Lin, D. (2018). Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. *arXiv preprint arXiv:1801.07455*. https://arxiv.org/abs/1801.07455

9. Si, C., Chen, W., Wang, W., Wang, L., & Tan, T. (2019). An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition. *CVPR*. https://openaccess.thecvf.com/content_CVPR_2019/papers/Si_An_Attention_Enhanced_Graph_Convolutional_LSTM_Network_for_Skeleton-Based_Action_CVPR_2019_paper.pdf

10. Zheng, X., Liu, H., & Wang, C. (2024). Skeleton Action Recognition via Graph Convolutional Network with Self-Attention. *Electronic Research Archive*. https://www.aimspress.com/article/doi/10.3934/era.2024129

11. Liu, R., Hu, Y., & Zhang, T. (2023). Motion Capture for Sporting Events Based on Graph Convolutional Neural Networks and Single Target Pose Estimation Algorithms. *Applied Sciences*, 13(13), 7611. https://www.mdpi.com/2076-3417/13/13/7611

12. HomeCourt. (2023). Shot Analysis FAQ. https://www.homecourt.ai/faq/shotanalysis

13. Pei, Y., Liu, Q., & Lin, J. (2024). The Potential of Human Pose Estimation for Motion Capture in Sports: A Validation Study. *Sports Engineering*. https://link.springer.com/article/10.1007/s12283-024-00460-w

14. Singh, R., & Kaur, H. (2023). Enhancing Sports Pose Estimation with Unsupervised Domain Adaptation. *ANVI*. https://internationalpubls.com/index.php/anvi/article/view/3516

15. Shi, L., Zhang, Y., Cheng, J., & Lu, H. (2019). Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition. *CVPR*. https://openaccess.thecvf.com/content_CVPR_2019/papers/Shi_TwoStream_Adaptive_Graph_Convolutional_Networks_for_Skeleton-Based_Action_Recognition_CVPR_2019_paper.pdf

16. Yulu Pan, Ce Zhang, Gedas Bertasius (2025). BASKET: A Large-Scale Video Dataset for Fine-Grained Skill Estimation https://arxiv.org/abs/2503.20781v1