NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Applied Mathematics and Informatics"

Software Project Report on the Topic:

Assessing the quality of sports refereeing

Submitted by the Student:

group #БПАД221, 3rd year of study	Laypanov Adamey Kazimovich
group #БПАД221, 3rd year of study	Karikh Dmitrii Yurevich
group #БПАД221, 3rd year of study	Nikitina Maria Andreevna
group #БПАД221, 3rd year of study	Kalinovski Vladimir Vladimirovich

Approved by the Project Supervisor:

Bashminova Daria Aleksandrovna
Senior Lecturer
Faculty of Computer Science, HSE University

Moscow 2024

Contents

1 Annotation

Basketball, as one of the most dynamic and fast-paced sports, demands accurate and unbiased refereeing to ensure fair play and maintain the integrity of the game. However, evaluating the quality of refereeing in basketball presents significant challenges due to the complexity of in-game scenarios, rapid player movements, and subjective decision making. Our work emphasizes the importance of using modern approaches to help maintain more precise accuracy in addition to human refereeing by incorporating machine learning techniques.

Recent advancements in machine learning and deep learning have significantly improved computer vision-based sports analysis, enabling automated foul detection and decision evaluation. Referees, despite their expertise, are prone to making errors due to factors such as limited viewing angles, fast-paced action, and cognitive biases. Misjudged calls can affect game outcomes and lead to disputes among players, coaches, and fans. By leveraging automated foul recognition systems, trained on large datasets of game footage, we can develop models that provide objective, data-driven assessments of referee decisions.

Our study explores how deep learning architectures can be applied to video footage for accurate foul detection. These models process spatiotemporal patterns in real time, recognizing subtle infractions that might be overlooked by human referees. The implementation of such systems could assist officials in making more informed decisions, reduce human error, and enhance the overall fairness of the game.

By integrating artificial intelligence into refereeing systems, we move towards a more transparent, accurate, and reliable officiating process that supports fair competition and upholds the integrity of basketball at all levels.

1.1 Keywords

Basketball Refereeing, Fair Play, Machine Learning, Foul Detection, Sports Technology, Automated Decision-Making, Labeling, DataBase, Whisper, Meta-Llama, YOLO, RF-DETR, Faster R-CNN

1.2 Аннотация

Баскетбол — один из самых динамичных и быстрых видов спорта, требующий точного и беспристрастного судейства для обеспечения честной игры и сохранения целостности соревнований. Однако оценка качества судейства в баскетболе представляет значительные сложности из-за сложных игровых ситуаций, быстрого передвижения игроков и субъективного характера решений судей. В нашей работе подчеркивается важность использования современных методов для преодоления ограничений человеческого судейства путем внедрения технологий машинного обучения.

В последние годы машинное и глубинное обучение значительно продвинулись вперед в области компьютерного зрения и анализа спортивных видеозаписей, позволяя автоматизировать обнаружение фолов и оценку судейских решений. Судьи, несмотря на свой профессионализм, подвержены ошибкам из-за ограниченного обзора, высокой скорости игры и когнитивных искажений. Неправильные судейские решения могут повлиять на исход матча и вызвать споры среди игроков, тренеров и болельщиков. Использование автоматизированных систем распознавания фолов, обученных на больших массивах видеоданных, позволяет объективно анализировать и оценивать решения судей.

В нашем исследовании рассматривается применение глубинных нейронных сетей для автоматического распознавания фолов по видеозаписям. Эти модели анализируют пространственно-временные паттерны в реальном времени, что позволяет выявлять тонкие нарушения, которые могут остаться незамеченными для человеческого глаза. Внедрение подобных систем может помочь судьям принимать более взвешенные решения, снизить вероятность ошибок и повысить объективность судейства.

Интеграция искусственного интеллекта в судейские системы ведет к созданию более прозрачного, точного и надежного судейства, способствующего честной игре и сохранению спортивных принципов на всех уровнях соревнований.

2 Introduction

Basketball officiating involves split-second judgments under high pressure, and even seasoned referees can miss or misinterpret subtle contacts amid fast breaks and crowded paint battles. Over the past five years, despite growing interest in sports analytics, no major professional league has deployed a fully automated foul-detection system. In contrast, our team has taken advantage of recent advances in computer vision, speech recognition, and large language models to build a working prototype in just six months, demonstrating that practical, real-time foul analysis is within reach.

In our study on automatic foul detection in basketball, we built a large annotated dataset by experimenting with both full NBA match videos and isolated frames extracted from those videos. This dual annotation strategy allowed us to capture collision events at both the macro and micro levels. For manual labeling, we relied on LabelStudio to draw bounding boxes around instances of player contact, while OpenAI's Whisper model provided automatic audio transcriptions of referee and player speech. By detecting keywords such as 'foul' and 'contact' in the audio stream and aligning their timestamps with the video, we achieved a more accurate localization of collision events.

To increase the size and diversity of the dataset, we applied augmentation techniques: rotations, scaling factors, random crops centered on collision regions and brightness variations. These transformations improved our effective sample count, helping our models generalize across different camera angles and lighting conditions typical of NBA broadcasts.

Our model comparison focused on two object-detection architectures: YOLO and RF-DETR, each tested under a range of hyperparameters, including learning rates, batch sizes, and IoU thresholds between. We also explored classical 3D convolutional networks and the graph-based VideoGraph model throughout our literature review. Using precision, recall, F1 score and mAP metrics, we got the best solution for our task.

Finally, we wrap our best-performing model into a lightweight web application that allows referees and coaches to upload footage, watch collision boxes overlaid in real time, and receive instant, timestamped alerts when potential fouls occur. By delivering a prototype we underscore both the feasibility and importance of AI-driven foul detection for improving officiating accuracy, consistency, and ultimately, the fairness of basketball competition.

2.1 Tasks Distribution

Table 2.1: Tasks and Performers

Tasks	Performer
Study the challenges of basketball refereeing errors, analyze existing solutions, collect basketball match videos, extract and annotate foul and no foul events	Maria
Dataset enrichment, annotating, split data into training, validation, and testing sets, develop an MVP of web application for interactive testing	Vladimir
Test different models, highlighting the pros and cons of the models, tune model's hyperparametres, analyze metrics	Adamey
Test different models, analyzing limitations and further improvements of the research	Dmitrii
Labeling frames	Maria, Adamey, Dmitrii, Vladimir
Format and structure the document	Maria, Adamey, Dmitrii, Vladimir

3 Literature review

The integration of artificial intelligence (AI) into basketball foul detection holds immense potential for enhancing the accuracy, consistency, and fairness of officiating. AI technologies, such as computer vision and machine learning, provide tools for automating decision-making and reducing reliance on subjective human judgment. This literature review explores methodologies and advancements relevant to foul detection in basketball, emphasizing their adaptation from related domains such as surveillance and football analytics.

Early work by Wu et al. focused on detecting unusual human behaviors through motion vector analysis and classification with SVM after PCA reduction, complemented by optical flow cues [19]. Their methodology laid the groundwork for combining traditional motion features with modern deep descriptors when identifying collision events. Similarly, Wang et al. demonstrated the utility of sequence modeling by extracting Histograms of Optical Flow Orientation and modeling event transitions with Hidden Markov Models [17]. This approach inspired our integration of temporal patterns alongside frame-based detectors.

Hand and gesture recognition research has further influenced foul detection methodologies. Dardas and Georganas achieved real-time hand gesture recognition by using a Bag-of-Features representation classified by SVM [3], which informed our streamlined pipeline tests. The advent of convolutional neural networks (CNNs) for gesture recognition by Lin et al. [6] and Alashhab et al. [1] validated CNNs' capacity to learn fine-grained interactions, directly motivating our collision classification architectures.

In the realm of sports analytics, object detection and tracking have been extensively studied. Bialkowski et al. utilized spatiotemporal tracking data to analyze soccer matches at scale [2], underscoring the importance of continuous player localization. Building on this, Zhang et al. combined YOLOv4 detection with DeepSORT tracking to maintain athlete identities despite camera motion [20], a technique we adopt to link collision events across successive frames. On the statistical side, Dobson and Massey revealed patterns of referee bias in football [4], while McHale et al. proposed quantitative rating systems for performance assessment [8], both of which influenced our evaluation protocols for consistency and error analysis.

Broadcast sports video studies by Lu et al. [7] and the HMDB51 benchmark by Kuehne et al. [5] highlighted the effectiveness of two-stream networks and pose estimation for action recognition, shaping our choice to pretrain on public datasets and fuse spatial and motion streams [13]. Furthermore, work on unsupervised action discovery by Wang et al. [18] and first-person basketball behavior prediction by Su et al. [14] reinforced the value of temporal context, justifying our

use of 3D convolutional baselines.

Specific to basketball and real-time applications, Miao et al. demonstrated low-latency score detection via template matching and OCR on broadcast overlays [9], guiding our deployment design for rapid foul flagging. Recent transformer-based detectors, such as the RealDETR variant that outperforms YOLO in real-time scenarios without non-max suppression, provide a promising alternative to traditional architectures [21]. A cross-sport case study by Siddiqui et al. applied YOLOv5/8 and Faster R-CNN to football foul detection and deployed the best model through a web interface [12], directly informing our end-to-end pipeline and web application framework.

Together, these contributions form a comprehensive backdrop for our work on automatic foul detection in basketball. By synthesizing motion and appearance cues, leveraging sequence models, and integrating robust tracking, we aim to build on these foundations to improve the accuracy and fairness of officiating through real-time, AI-driven foul detection.

Table 3.1: Comparison of Automatic Foul Detection Methods Across Sports

Research	**Sport**	**Data**	**Approach / Models**	**Key Differences to This Work**
"AI-Driven Image Recognition System for Automated Offside and Foul Detection" by Q. Zhang, L. Yu, and W. Yan	Football	Broadcast video frames	CNN-based offside + foul detection pipeline with YOLOv4	Purely vision-based; no audio cues or LLM-based temporal parsing
"Comparative Analysis of Automated Foul Detection in Football Using Deep Learning Architectures" by A. Rabee *et al.*	Football	7,000 annotated images	Comparative study of 8 CNN backbones (EfficientNetV2, ResNet50, etc.) + GradCAM++	Focus on image-level classification; ours adds audio transcription and semantic LLM extraction
"Foul Detection in Sports Using EfficientNetB0" by D. Shah and V. S. Gutte	Football	Real-time match streams	YOLOv8 for detection + EfficientNetB0 for classification	Lightweight real-time vision system; lacks multi-modal fusion and precise segmenting into clips
"Real-Time Foul Detection in Football Matches Using Machine Learning Techniques" by B. S. Siddiqui, Z. A. Mridul, Z. Habib, and I. Sakib	Football	Football match video + audio transcript	YOLOv5/8 and Faster R-CNN for foul detection, deployed via web interface	Vision-focused pipeline; ours extends to audio-based detection and LLM parsing in basketball
"A Lightweight Fine-Grained Action Recognition Network for Basketball Foul Detection" by C.-H. Lin, M.-Y. Tsai, and P.-Y. Chou	Basketball	Annotated foul action clips	Fine-grained 3D CNN for frame-level foul classification	Vision-only, short clips; our pipeline automates segmentation and uses audio context
This Work	Basketball	NBA broadcasts + audio transcript	YOLOv12, RF-DETR + Whisper-Large + Meta-Llama	Combines vision and audio; automatic clip extraction; real-time web app

4 Data

Collecting a training dataset is the most painstaking but mandatory work. To obtain high model performance, it is necessary not only to manually collect foul moments from a large number of matches, but also to select the best of them and augment them.

4.1 Selection of sources and labor intensity solution

- **Selection of Sources**: We collected basketball game recordings available from open sources, considering both Russian matches and NBA games. Ultimately, we decided to use NBA footage exclusively, as the quality of the Russian match recordings was insufficient for our needs. NBA game recordings offer high image quality and a variety of camera angles, providing a more reliable foundation for detecting player collisions. Finally, we settled on using YouTube NBA recordings with segments of matches that show fouls and collisions between players (the length of the video is about 10 - 20 minutes, which is much more convenient) and, additionally, we found several full replays of NBA matches that were also used.

- **Frame Extraction**: Our main goal was to simplify the assembly of the dataset, as this is the most labor intensive process in the work. Initially, we tried to manually "catch" a foul on video and take these frames, which would have been a very inefficient process. However, we came to the conclusion that it is possible to automatically highlight the moment of a foul based on the signal of a referee or commentator, which led us to a more efficient methodology.

4.2 Methodology

Basketball has many complicated rules, and you cannot reliably spot a foul just by watching a referee's hand signals. That's why we decided to listen to commentator and referee via Whisper-Large to turn game audio into text and Meta-Llama 3 70B-Instruct to pick up foul calls.

- **Whisper-Large**: It is a large speech-to-text model from OpenAI based on a transformer encoder-decoder architecture. It converts raw game audio into clean text with very high accuracy, even in noisy conditions. Trained on massive multilingual datasets of paired audio and transcripts, it first encodes audio into hidden representations and then decodes those into word sequences. Basketball games have overlapping voices and crowd noise, Whisper-Large handles such complexity far better than lightweight models, giving us reliable transcripts as the foundation for foul detection.

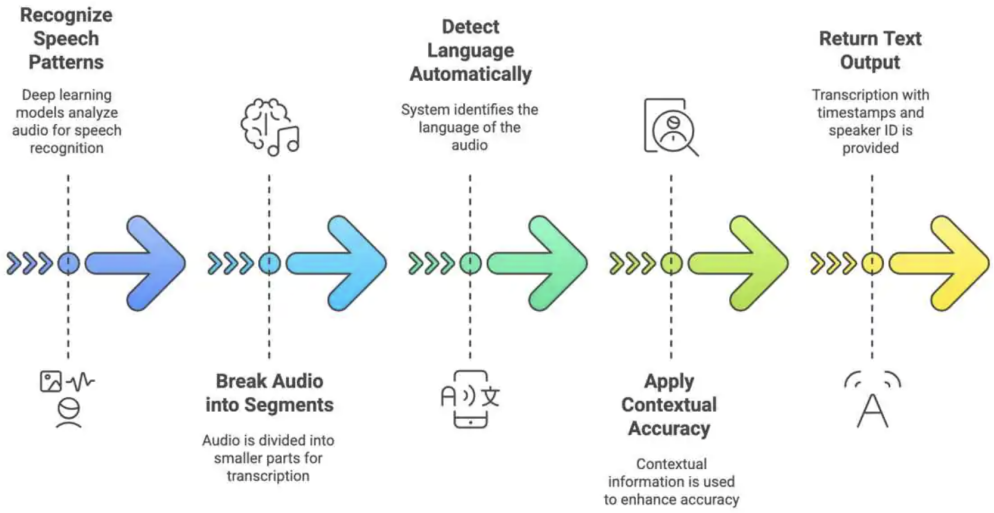

Figure 4.1: Whisper API process

- **Meta-Llama 3 70B-Instruct**: It is a 70-billion-parameter instruction-tuned language model from Meta, optimized for understanding and following prompts. It takes the full game transcript and, given our instruction, outputs a structured JSON list of fouls with precise timestamps. Instead of brittle keyword searches or regex, Meta-Llama understands context—catching indirect foul mentions or crowd reactions—and reliably returns only real foul events with accurate timing.

Together, these two models let us 'listen' and 'understand' the game audio to pinpoint fouls, rather than trying to infer everything from video frames or referee gestures alone.

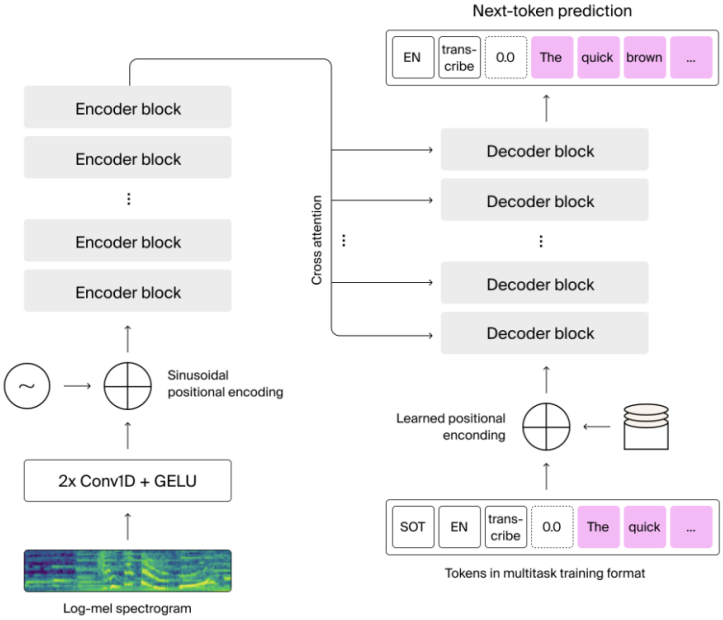

Figure 4.2: Whisper architecture

We wrote code that helped us get many small (10 or 15 seconds) segments from a long video, where there were a few seconds before the foul, the foul, and seconds after the foul.

First, the script grabs a YouTube video of a full basketball game with yt dlp and pulls out the audio track in WAV format. It then converts that WAV into MP3 and, if the file is too big, chops it into smaller pieces with pydub so each chunk stays under the upload limit.

Next, each MP3 piece goes to the Whisper-Large API, which does speech-to-text on the referee's calls and crowd reactions. This gives us a clean transcript of everything said during the game—far more reliable than trying to guess fouls from video alone.

We feed the full transcript into Meta-Llama 3 70B-Instruct with a simple prompt: "Find every moment someone calls a foul, give me the start and end times, and a short description." The model returns a neat JSON list of foul events, complete with timestamps and notes.

Finally, the script loops over that JSON list. For each foul, it uses ffmpeg to cut out exactly that clip (encoding it in MP4), writes a tiny metadata file with the foul number and times, and saves individual frames at about 18 frames per second. Everything lands on one's Google Drive in folders named foul 1, foul 2, and so on, each containing a short video, its frames, and a JSON file.

In the end, we get a ready-to-use set of foul clips and frames without any manual effort. By combining open-source media tools with powerful voice-and-language AI, we have built a simple, fully automated way to find and package every foul in a basketball match.

4.3 Annotation Process

During the annotation process, we also had two decisions to make, to label the video itself or individual frames. In the beginning, we tried to work with videos by shortening them from 10-15 seconds to 1-3 seconds. We saved these short segments, annotated them, and saved the markup in a JSON file. Thus, creating a database from the videos and the combined overall JSON file. However, the end-to-end models we tested at the initial stage, trained on videos, showed poor results. Ultimately, our final solution was to mark up frames separately, for which we used ready-made 1-3 second videos, which we divided into frames using code.

- **Annotation Tool**: We used LabelStudio for annotating the frames.
- **Annotation Methodology**: For each selected frame, collision events were precisely marked using temporal indicators and designated regions where the collision occurred. In our labeling method there were 2 classes: `foul` and `no foul`. We also tried to collect frames where the

players are in a position close to each other, but the rules of the game are not violated (e.g. they are standing/running next to each other, one player is blocking another, but without touching or grabbing), such shots we put in `no foul` class. To effectively train the model, it was important to maintain class balance. Note that we focused solely on marking instances of player collisions (fouls) and did not consider the positioning of players on the court.

- **Quality Assurance**: The annotation results were validated by cross-checking among multiple annotators to ensure accuracy and consistency.

Figure 4.3: LabelStudio

4.4 Data Augmentation

Objective: Augmentation techniques were applied to increase the number of training samples and to expose the model to a wider variety of viewing angles, thus enhancing its robustness in detecting collisions under various conditions.

- **Horizontal Flip**: A horizontal mirror of each image simulates plays filmed from the opposite side of the court.

- **Rotation** ($\pm 7°$): Small random rotations mimic slight camera tilts and changes in viewing angle. In live broadcasts, camera operators pan and tilt to follow action, which can introduce up to a few degrees of rotation.

- **Shear ($\pm 5^\circ$ horizontal and vertical)**: Shearing skews the image along the horizontal or vertical axis, approximating perspective distortions introduced by wide-angle lenses or oblique camera placements.

- **Hue Shift ($\pm 15^\circ$)**: Adjusting the hue channel simulates changes in the broadcast color balance, such as warmer or cooler tones under varied lighting. By seeing a range of hue shifts, the model avoids overfitting to the exact color palette of the training games.

- **Saturation Adjustment ($\pm 13\%$)**: Modifying saturation levels creates more vivid or muted color schemes, reflecting conditions such as bright arena lights. Saturation augmentation ensures that the model focuses on motion cues rather than absolute color intensity.

- **Brightness Adjustment ($\pm 14\%$)**: Varying brightness accounts for fluctuations in overall scene illumination, whether due to stadium light dimming or camera auto-exposure. This helps the model maintain performance under both very bright and fairly dark conditions.

- **Gaussian Blur (sigma up to 0.8 px)**: Applying a mild blur simulates motion blur from fast player movements and video compression artifacts.

- **Additive Noise (up to 0.93% of pixels)**: Exposure to noisy inputs makes the model more resilient to imperfect video quality.

By combining these transforms, we expanded our dataset roughly threefold (from $\sim 1\,900$ to $\sim 5\,700$ samples). This augmentation pipeline not only increases the size of the dataset, but, diversifies the training distribution to better match the variability of real NBA footage.

4.5 Final Dataset Composition

The final dataset comprises a collection of frames from NBA shortcuts, enriched with augmented variations and the annotations in .json format. The data set was then exported in the COCO/YOLO format for further training.

5 Explored methods

5.1 YOLOv12 [15]

YOLOv12 is the latest version of the You Only Look Once (YOLO) family of real-time object neural network models, renowned for its attention-centric architecture. It was based on previous YOLO versions by integrating transformer-like attention modules into the classical CNN pipeline, also achieving higher accuracy saving the fast inference speeds for which YOLO is known

5.1.1 Model Architecture and Key Features

YOLOv12 follows the same pipeline as previous models in the YOLO family do-backbone used for feature extraction, neck for feature combining, and head for predictions. However, a new version also introduces new modules at backbone and neck stages which help enhance the performance without affecting the performance.

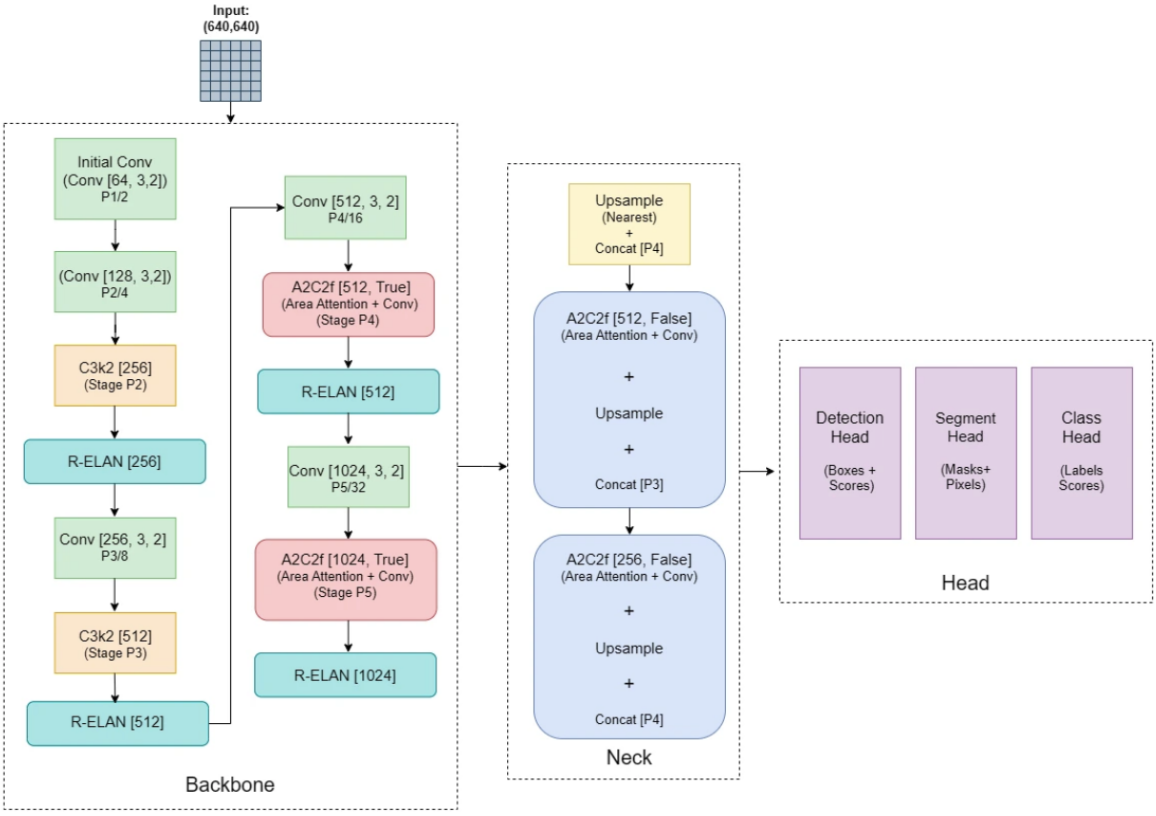

Figure 5.1: YOLOv12 Architecture

- **Backbone:**

  It is based around *Residual Efficient Layer Aggregation Network(R-ELAN)* [15] - module that categorizes features in different scales through several convolutional paths adding residual

connection. This ensures improved gradient flow and feature reuse deep in layers. So, basically, it is still the convolutional layers but with blocks which use many small kernels instead of bigger ones, which gives improvement in speed without drop in performance.

- **Neck:**

  Instead of relying entirely on convolutional feature combining used in previous iterations of YOLO family, YOLOv12 uses self-attention algorithms to emphasize important regions in feature maps. *Area Attention* can be considered as key improvement here. It works by dividing feature maps into several smaller patterns with applying of self-attention within each pattern. This approach provides that models can still "see" the entire image, but gives the opportunity to avoid the computational costs of vanilla self-attention mechanisms. Besides, Flash Attention is also used in this model in order to reduce the memory/speed overhead of computations in attention mechanisms.

  Through this attention based neck YOLOv12 can more precisely concentrate on crucial visual patterns helping model to excel at detecting several objects in crowded areas and to distinguish overlapping objects.

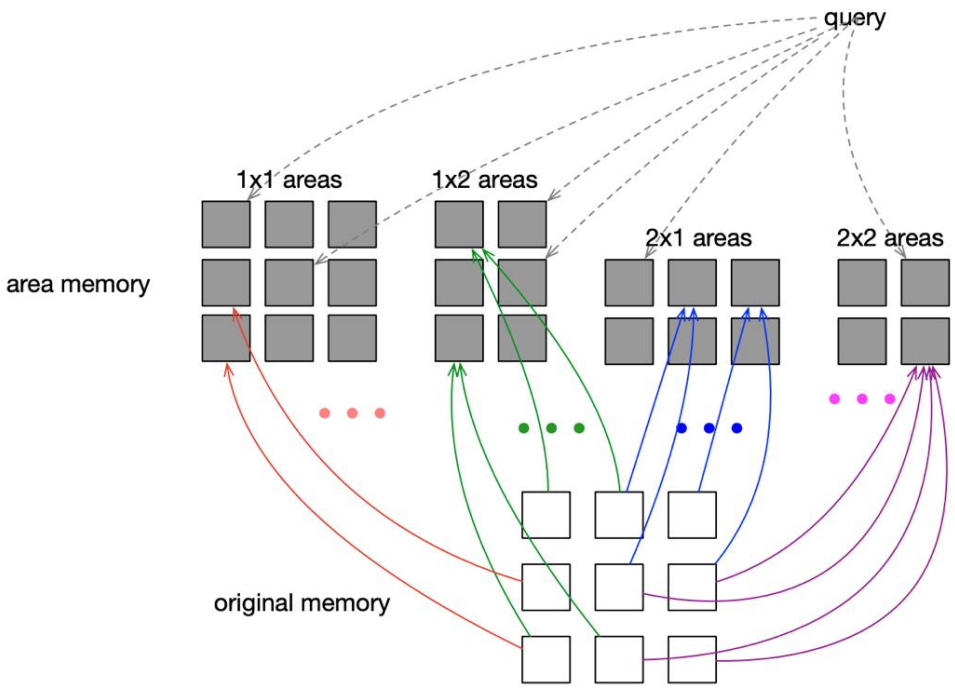

Figure 5.2: Area Attention for the 2- dimensional case

- **Head:**

  YOLOv12 is implemented using the anchor-free, decoupled head approach which were already introduced in previous versions(e.g. YOLOv11). Anchor-free detection do not use predefined anchor boxes and provides for each feature-map point directly predict an object's center offsets and size, while a decoupled head divides the prediction into two parallel branches—one designated to object classification and the other to bounding-box and IoU regression.

5.1.2 Performance

Table 5.1: COCO detection results for YOLO12 variants [16]

Model	$mAP_{50–95}$	Speed T4 TensorRT (ms)	params (M)	FLOPs (B)	Comparison (mAP/Speed)
YOLO12n	40.6	1.64	2.6	6.5	+2.1%/–9% (vs. YOLOv10n)
YOLO12s	48.0	2.61	9.3	21.4	+0.1%/+42% (vs. RT-DETRv2)
YOLO12m	52.5	4.86	20.2	67.5	+1.0%/–3% (vs. YOLO11m)
YOLO12l	53.7	6.77	26.4	88.9	+0.4%/–8% (vs. YOLO11l)
YOLO12x	55.2	11.79	59.1	199.0	+0.6%/–4% (vs. YOLO11x)

On COCO benchmark, it outperforms all earlier YOLO models in the real-time range. For example, YOLOv12-N (nano model) reaches 40.6% mAP with an inference latency of 1.6 ms per image on an NVIDIA T4 GPU. It surpasses YOLOv10-N and YOLOv11-N by +2.1% and +1.2% mAP respectively, with comparable inference speed.The advantage persists across model sizes: larger YOLOv12 models also achieve higher accuracy than their YOLOv8/v10/v11 counterparts at similar FLOPs. Notably, YOLOv12 sets a new efficiency benchmark against other cutting-edge detectors.

In terms of raw speed, YOLOv12's smallest models run in a few milliseconds: e.g., on one test, YOLOv12-nano processed images in 4.6 ms on GPU, which is on par with the fastest YOLOv5/ YOLOv8 models.

5.1.3 Our Training

1 Optimization Settings

- Optimizer: `AdamW`
- Initial learning rate: $\text{lr}_0 = 0.001$
- Final learning rate: $\text{lr}_{\text{final}} = \text{lr}_0 \times \text{lrf} = 0.001 \times 0.01$
- Momentum: 0.937
- Weight decay: 0.0005
- Warmup epochs: 3
- Warmup momentum: 0.8
- Warmup bias learning rate: 0.1
- Early stopping patience: 100
- Automatic mixed precision (AMP): `True`

2 Batching, Duration

- Total epochs: 100
- Batch size: 16

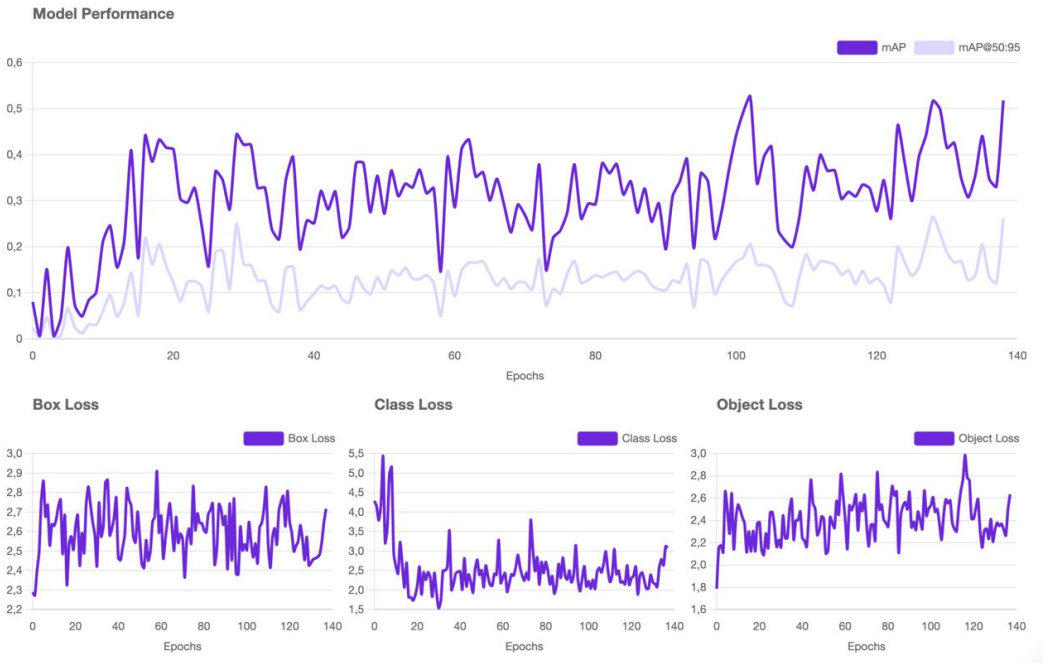

Figure 5.3: Training graphs

5.2 RF-DETR [11]

The DETR family of models (including RF-DETR) follows a similar structure: a backbone extracts image features, a transformer (encoder and decoder) uses self-attention to model global context and object relationships, and the detection head outputs bounding boxes and classes. Instead of using predefined anchor boxes or non-maximum suppression post- processing, it's design implements the transformer's attention mechanism helping the model to capture relationships between objects, improving detection especially in crowded or occluded scenes. Below, RF-DETR's architecture splitted into its three main parts – Backbone, Neck, and Head with focusing each part's design.

5.2.1 Model Architecture and Key Features [22]

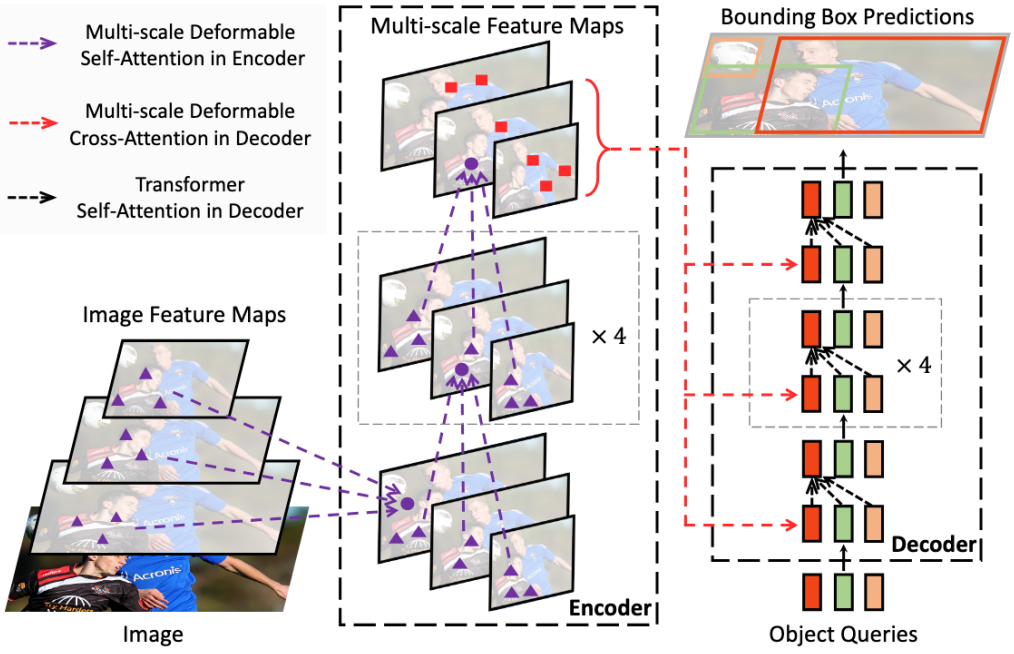

Figure 5.4: DETR base architecture

- **Backbone:**

  RF-DETR introduces a transformer backbone *DINOv2 ViT* [10] whereas early DETR models used ResNet CNNs. ViT backbone splits the input image into patches and processes them with multiple self-attention layers. This produces a set of high-level image feature tokens that encode information from the entire image. Besides, it gives RF-DETR global context awareness and feature enrichment, as the self-attention in the backbone models relationships across the image that CNNs might miss. It improves the model's adaptability to different

data domains and complex scenes-the model "understands" the scene layout more holistically. By using DINOv2 pre-training, the backbone gives learned visual knowledge from big generic image data before detection training. This is a important change from earlier iterations of DETR models which are often trained from scratch or with ImageNet pre-trained CNNs. As a result RF-DETR performs well even on novel domains or when data is limited and has faster convergence during training.

Unlike other DETR models, RF-DETR single scale high resolution feature map from the backbone. The tokens from DINOv2 corresponding to one resolution of patches are directly used for prediction reducing computational overhead and latency

- **Neck:**

  In the original DETR, a transformer encoder layer processes with CNN features before the decoder. In RF-DETR, feature processing is already made by the ViT backbone, which itself is a transformer encoder on image patches. Therefore, RF-DETR doesn't need a separate encoder layers working with the features. Any additional encoder-like design choices are minimal or integrated into either the tail of the backbone or the head's first layers. By merging the backbone and neck stages, RF-DETR eliminates redundant attention layers, which is the key to its superior efficiency. The result is a stable transition from features to predictions without more intermediate processing.

  Feature projection module involves small projection module to interact between ViT and DETR decoder. This design was used in LW-DETR and is inherited by RF-DETR.

- **Head:**

  RF-DETR's decoder uses deformable attention. This allows to focus on a set of feature locations near an estimated position, rather than attending to every pixel. In comparison to the original DETR which attends globally, deformable attention greatly reduces computation and speeds up training while improving accuracy on small or overlapped objects.

  The decoder's multiple layers perform hierarchical attention design. RF-DETR uses the idea of iterative bounding box refinement through the decoder, in other words it means that after each decoder layer, predicted bounding box prediction of each query is used in the next layer of decoder.

- **Summary:**

  Overall, RF-DETR's architecture combines a ViT backbone with a transformer decoder head, using a minimal designed neck to connect them together. The backbone provides high-

quality features, the neck-efficient flow of information, and the head's attention mechanisms allow it to detect objects even under challenging conditions with end-to-end simplicity with no region proposals, no anchors, no NMS.

5.2.2 Performance

Table 5.2: RF-DETR in comparison to other models on COCO [11]

Model	mAP @0.50:0.95 on COCO	params (M)	Total Latency T4 bs=1 (ms)
D-FINE-M	55.1	19.3	6.3
LW-DETR-M	52.5	28.2	6.0
YOLO11m	51.5	20.0	5.7
YOLOv8m	50.6	28.9	6.3
RF-DETR-B	53.3	29.0	6.0

5.2.3 Our Training

We use the following hyperparameters:

1 **Optimization and Learning Schedule**

- Learning rate: $\text{lr} = 0.0001$
- Encoder learning rate: $\text{lr}_{\text{encoder}} = 0.00015$
- Weight decay: $\text{weight\_decay} = 0.0001$
- ViT layer decay: $\text{lr}_{\text{vit\_layer\_decay}} = 0.8$
- Component decay: $\text{lr}_{\text{component\_decay}} = 0.7$
- Gradient accumulation steps: 8
- Gradient clipping max norm: 0.1
- Automatic mixed precision (AMP): `False`

2 **Batch Configuration and Training Length**

- Batch size: 2 (effective 16 with gradient accumulation)
- Total epochs: 100

- Warmup epochs: 0
- Early stopping: `True`, with patience = 15, and min_delta = 0.0005

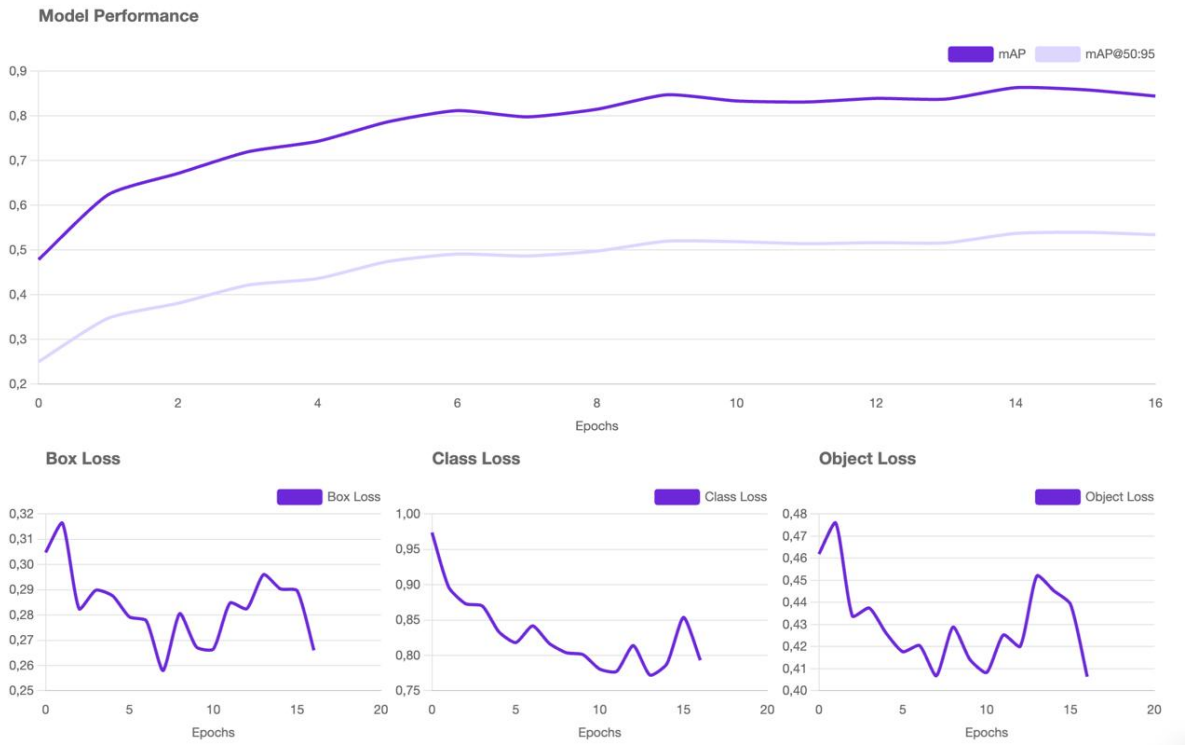

Figure 5.5: Training graphs

5.3 Faster R–CNN

5.3.1 Why we still tried a two-stage detector

Most modern leaderboards are topped by one-stage or transformer detectors (YOLOv12, RT-DETR, etc.). Nevertheless, we trained a **Faster R-CNN** baseline for two reasons:

1 Two-stage models output *explicit region proposals*. Those boxes help us visualise *where* the network thinks a foul happens, which is handy for debugging.

2 Earlier work on football foul review showed that a two-stage detector, though slow, can act as a "sanity-check" against flashier models. We wanted the same cross-check in basketball.

5.3.2 Exact implementation

- **Code.**

  `torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)` — vanilla ResNet-50 + FPN from PyTorch. We swap the final classifier for `nn.Linear(1024, 2)`, so the detector outputs *foul* or *no_foul*; the box-regression head is left intact.

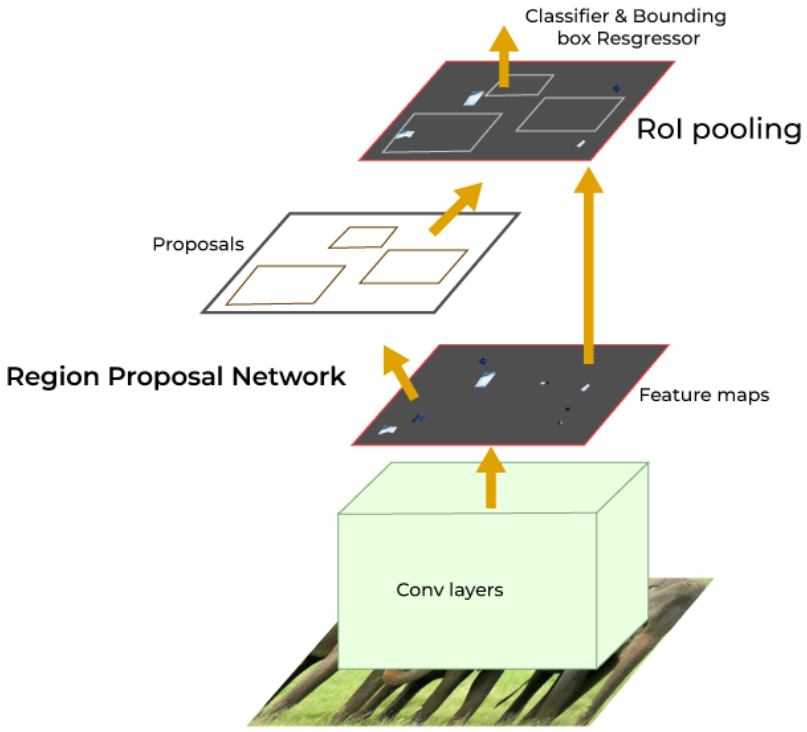

Figure 5.6: Faster R-CNN Architecture

- **Anchors.** Default sizes $\{32, 64, 128, 256, 512\}$px and aspect ratios $\{0.5, 1, 2\}$; no custom tuning.

- **Training pipeline.**

  1 *Input.* Short 2–3s video clips centred on each whistle, converted to RGB frames at 18fps; every frame inherits the clip label.

  2 *Transforms.* `ToTensor()` → ImageNet normalisation.

  3 *Loader.* `DataLoader(batch_size=2, shuffle=True, num_workers=4)` with the canonical `collate_fn`.

  4 *Optimiser.* SGD, LR $= 0.005$, momentum 0.9, weight-decay 5×10^{-4}.

  5 *Schedule.* `StepLR(step_size=3, gamma=0.1)`, total 10 epochs.

5.3.3 Clip-based training *vs.* frame-based training

Unlike YOLOv12 and RF-DETR, which saw *individual* annotated frames, Faster R-CNN was fed a *sequence* of near-identical images. Because the architecture is purely spatial (each frame processed in isolation), the extra temporal context became noise: the network saw dozens of almost duplicate inputs, many of which had no foul, yet all were labelled foul. This contradiction led to

unstable gradients and, ultimately, to a detector that prefers to predict "nothing" rather than risk false positives.

5.3.4 Observed results

After ten epochs the model's foul confidence stayed near zero; on the test set it produced virtually no true positives (mAP≈ 0). By contrast, YOLOv12 and RF-DETR — trained on static frames — each recovered a non-trivial fraction of fouls. The failure highlights a mismatch between Faster R-CNN's two-stage design and our task:

- **Region focus.** RPN boxes isolate *individual* players, yet a foul is often an *interaction* between two bodies; contextual cues outside the proposal are lost.
- **No temporal reasoning.** Any contact motion unfolding over adjacent frames is invisible to a static detector.
- **Data hunger.** Two-stage models need many positives per class; due to lack of free data our dataset of foul clips was too small to train both the RPN and the classifier robustly.

5.3.5 Strengths and weaknesses at a glance

Pros	Cons
High localisation fidelity on well-defined objects.	Slow — RPN + per-ROI head incur extra passes.
Region proposals are interpretable for humans (good for debugging).	Struggles with relational events (player–player contact).
FPN helps at multiple scales.	Requires large, clean datasets; prone to overfitting small ones.

5.3.6 Role inside the multimodal system

Even with poor standalone accuracy, the detector remains useful as a *negative control*. Whenever the video branch **fails** to flag a referee-called foul (identified in the audio/LLM stream) we know the visual cue was either (i) off-camera, (ii) too subtle for any vision model, or (iii) masked by poor anchor choices — information that guides future data collection and model tuning. Hence the two-stage baseline, though weak, still enriches our referee-validation workflow.

6 Referee validation

Big picture. Our foul–checking pipeline speaks two languages at once. The *audio side* listens to the arena soundtrack, hears whistles or commentary, and decides "the ref just called something". The *video side* watches every frame and decides "I see (or do not see) illegal contact". Any moment when these two voices either sing in unison or argue with one another is a moment worth inspecting. Everything that follows is simply plumbing—and a dash of UX—to make that inspection fast and transparent.

6.1 Why bother with two modalities?

- **Audio knows intent.** A shrill whistle followed by *"blocking foul, number twenty-three"* is the *referee's official verdict*. Even the best vision model cannot infer intent from pixels alone.
- **Video knows evidence.** Pixels do not lie about whether a defender's arm hacked the shooter. They are also immune to crowd pressure and game management politics.

In short, **audio tells us the call, video tells us the body language**. When these stories match, we gain confidence; when they clash, we open the replay.

6.2 Audio branch in plain English

1 **Transcribe.** Raw broadcast sound → Whisper-large (16kHz, 30 ms hops). Result: a JSON stream `{time_start, time_end, text}`.

2 **Detect whistles.** A 3rd-order Butterworth band-pass (2500–4500 Hz) slides over the waveform; peaks longer than 80ms are marked `<whistle>`.

3 **Classify speech.** Each sentence goes to a tiny prompt:

> "You are an NBA score-table official. Label the snippet **foul**, **no_foul** or **other**. Snippet: "…" →"

Llama-3 8B answers in one word. A whistle without words is coerced to `foul`.

The output is a tidy list $\mathcal{A} = \{(s_i, e_i, \hat{y}_i^{\text{audio}})\}$, $\hat{y}^{\text{audio}} \in \{\mathsf{foul}, \mathsf{clean}\}$.

6.3 Video branch – what the eye sees

1 **Frame scoring (per–frame inference).**

Each broadcast frame is resized to 1280×720 and passed through three detectors:

- *RF-DETR* — our primary detector. Its transformer-based global attention excels at capturing player–to–player interactions that define fouls.
- *YOLOv12* — serves as a complementary one-stage model, offering a different inductive bias that sometimes catches fast, small-scale contacts.
- *Faster R-CNN* — retained as a control baseline. Though its clip-trained performance was weak (Sec. 5.3), it still contributes a third perspective.

Each model outputs a per-frame *foul* confidence score; we average the three to obtain a single scalar $s_t \in [0, 1]$.

2 **Temporal smoothing.**

The raw score sequence $\{s_t\}$ is passed through a simple causal filter $\tilde{s}_t = 0.4\, s_t + 0.6\, \tilde{s}_{t-1}$, which suppresses isolated spikes while preserving genuine foul bursts.

3 **Binary decision.**

A frame is flagged as foul when $\tilde{s}_t > 0.55$. This threshold was chosen via F_1-maximisation on a held-out validation game.

4 **Episode merging.**

Consecutive foul frames are merged into intervals (p_j, q_j), then expanded by $\pm$5frames to capture context. Intervals separated by less than 160ms are fused. The final set of video episodes is

$$\mathcal{V} = \{(p_j, q_j) \mid \text{foul episode}\}.$$

1 **Temporal smoothing (debounce filter).**

Raw detector scores wiggle. We pass the sequence $\{s_t\}$ through a causal one-pole filter $\tilde{s}_t = \alpha s_t + (1-\alpha)\tilde{s}_{t-1}$ with $\alpha = 0.4$; this kills single-frame spikes without smearing legitimate bursts of contact.

2 **Binary decision.**

A frame is flagged foul if $\tilde{s}_t > 0.55$. (Threshold chosen on a held-out game by maximising the F_1 against human annotations.)

3 **Episode merging.**

Consecutive foul frames are glued into one episode (p_j, q_j). We additionally extend each episode by ±5frames (0.28s) to capture context, then fuse episodes separated by fewer than 160ms. The final catalogue is

$$\mathcal{V} = \{(p_j, q_j, \hat{y}_j^{\text{video}} = \mathsf{foul})\}.$$

On a typical game we obtain 20–45 such video episodes, roughly matching the real foul count plus a modest number of false positives the referee never called.

6.4 How we line up the two stories

Step 1. Normalize the clocks. Camera feeds and audio often slip by a few hundred milliseconds due to encoding pipelines. We sync by cross-correlating the raw audio with a template whistle and aligning the first big peak with the first detector whistle; residual skew is under 30ms.

Step 2. Temporal IoU matching. For each audio interval (s_i, e_i) we scan the set $\mathcal{V}$ for an overlapping video interval (p_j, q_j). If the temporal intersection-over-union

$$\text{tIoU}(i, j) = \frac{|[s_i, e_i] \cap [p_j, q_j]|}{|[s_i, e_i] \cup [p_j, q_j]|} > 0.30$$

we declare the two modalities *paired* and attach the higher of the two confidences to the joint event. The 0.30 threshold was chosen after grid-searching $\{0.1, 0.2, \ldots, 0.5\}$ on three validation games: lower values caused false pairings between unrelated actions; higher values missed genuine matches when the commentator spoke late.

Step 3. Labelling the quartet. Every whistle and every video episode now falls into exactly one of four buckets:

Bucket	Audio verdict	Video verdict
Correct call	foul	foul
Dubious call	foul	clean
Missed call	clean	foul
Clean play	clean	clean

Corner-cases:

- If two whistles overlap a single video episode, both inherit the video verdict (usually foul).
- If multiple video episodes overlap one whistle, we keep only the episode with the highest mean detector score; extras are flagged "*sub-episodes*" and hidden from the dashboard by default.

These rules minimise double-counting while preserving every contested moment for the reviewer.

6.5 What we get as output

CSV log Each foul episode becomes one row with these fields: `start_time`, `end_time`, the averaged video-confidence score, a short Whisper transcript excerpt, the LLM verdict (`foul` or `clean`), and a `match?` flag. This flat table lets coaches filter for *Missed calls* or *Dubious calls* in just a few clicks.

Storyboard panel For every contested episode we produce a single PNG "storyboard panel": twelve evenly spaced frames in a 320px-wide grid. Reviewers can scan the series of snapshots to grasp the play's sequence without manually scrubbing the video.

Timeline plot A game-long horizontal axis with two rows of ticks: audio-detected events on top, video-detected events below. Colour coding (green/orange/gray) makes stretches of disagreement leap off the page.

All of these artefacts are exported automatically by `ref_validation.ipynb`. The only external library beyond PyTorch is `matplotlib` for plotting."'

6.6 Where audio and video disagree—and why

- **Borderline contact.** Slight hip-checks or incidental brushes often look like fouls to the vision model, yet referees frequently rule them incidental. These plays end up flagged only by the video branch.
- **Off-camera fouls.** Sometimes a defender's illegal hold or off-ball hack occurs outside the main broadcast frame. The audio branch picks up the whistle and call, but the video branch sees no contact.

- **Visual artefacts.** Rapid camera pans, lens shake, or player flops can fool the detector into thinking there was illegal contact, resulting in video-only flags.
- **ASR mis-hears.** Whisper occasionally mistranscribes "no foul" as "know foul" or fragments like "looming foul," causing the audio branch to incorrectly register a call.

7 Limitations

The proposed system demonstrates potential but faces several limitations impacting its performance and applicability. These constraints broadly fall under categories related to data input, temporal and spatial modeling, rule-based contextualization, and dataset scope and size.

Firstly, the major data input constraint is that the current system relies on a **single-camera view only** (from broadcast video). This monocular vision fails to capture significant details visible from other perspectives, such as slight contacts or off-ball incidents, which can lead to inaccuracy caused by occlusion and visual uncertainty (Lu et al., 2013)[7]. Additionally, analysis is restricted solely to **RGB frames**, and thus significant modalities such as optical flow, skeletal pose, audio signals, or ball tracking are disregarded, even though it has been shown to improve detection accuracy for comparable tasks (Kuehne et al., 2011)[5].

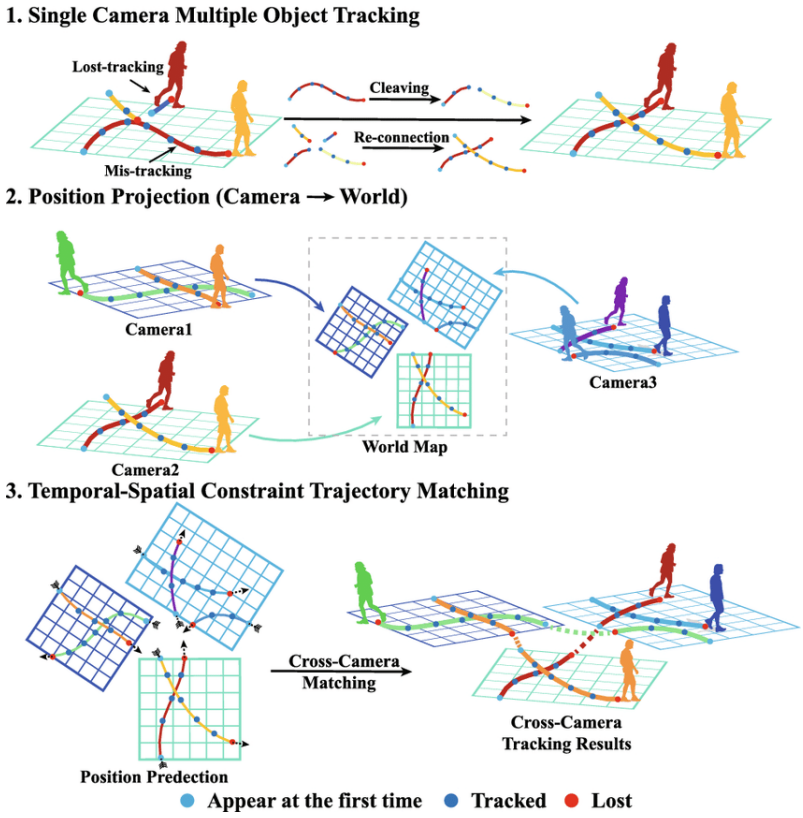

Figure 7.1: Multi-object Tracking

Next, consequently, the temporal and spatial modeling limitation arises, since the model analyzes isolated static frames **without considering continuous temporal dynamics** or persistent player identities. The absence of multi-object tracking and temporal context hinders recognition of fouls that manifest as temporal sequences rather than isolated events (Su et al., 2017)[14]. Furthermore, the model **lacks explicit pose estimation**, which limits its ability to differentiate normal physical contacts from foul actions dependent on the position of the limb.

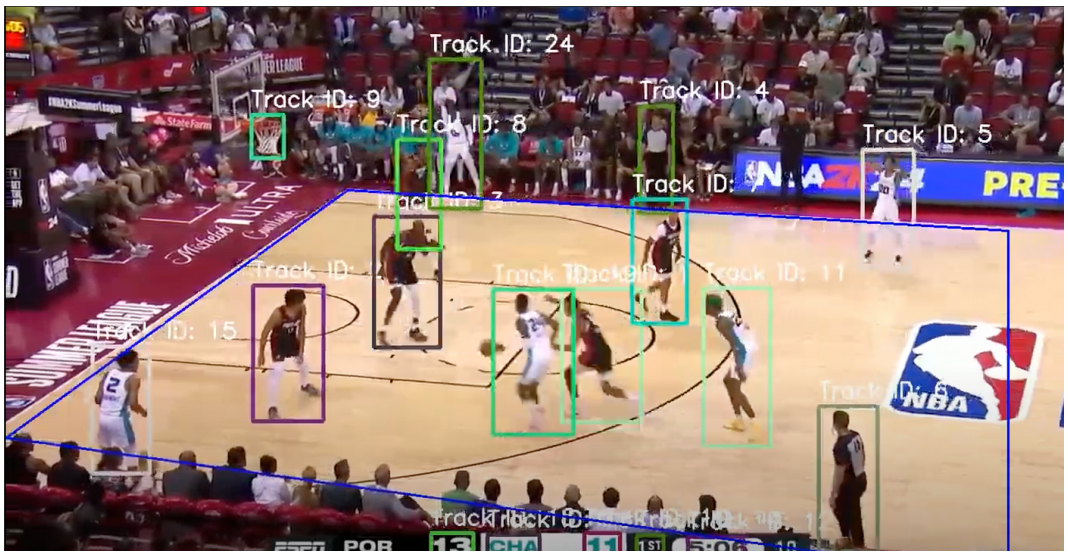

Figure 7.2: Players positions

Another issue of the chosen model is absence of rule-based context, since the current approach treats foul detection purely as a visual **binary classification** task, without explicit encoding of basketball rules. Fouls in basketball are context-dependent, governed by nuanced rules such as legal guarding positions, charge/block distinctions, or hand-checking interpretations. Without integrating such rules, the model cannot "perfectly" differentiate legal from illegal contacts.

As for the dataset limitations, a major constraint is the limited size and scope of the training dataset ($\approx$ 5700 images/foul instances with balanced classes). Though it is a common problem for computer vision related tasks with **no public dataset**. This limited dataset may not capture the broad variability in foul scenarios, potentially increasing the likelihood of overfitting and poor generalization. The dataset's restriction to NBA broadcasts further limits model generalizability to other basketball contexts (college, international play).

In summary, even though the current version of the model faces the above mentioned limitations, this is the optimal approach given the dataset constraints and the peculiarity of foul events in basketball.

8 Further Improvements

8.1 Incremental improvements for the current RF–DETR frame pipeline

The detector is trained on $\sim 5\,700$ broadcast frames and reaches mAP@50 = 86.3 %. There remains room for improvement even without modifying the data modality.

Dataset. Run the model on all unlabelled frames, collect high-score false positives, verify, and append to the *no-foul* class (hard-negative mining); add $\approx 1\,000$ verified samples. Inspect ~ 300 frames to fix loose boxes; tighter IoU might contribute to mAP@50:95. Maintain class balance by oversampling *foul* when required.

Augmentation. Enable Mosaic and CutMix (same-class only) and random input scales. These operations enlarge background and size variation and should improve mAP@50 by several pp on broadcast footage.

Semi-supervised fine-tune. Accept detector outputs with confidence > 0.6 on raw video as pseudo-labels and fine-tune for several epochs.

Considering these ideas might improve the ongoing approach without severe architectural changes.

8.2 Hypothetical full-scale, rule-aware system

With unlimited data and compute, foul recognition can be framed as a full spatio-temporal reasoning problem. Every play is captured by **synchronised baseline, sideline and overhead cameras** (50–120 fps) while an **official SportVU-style tracking feed** supplies (x, y) coordinates of all players and the ball at 25 Hz.

A **multi-view DETR** extracts masks for every player and the ball on each view; a **Kalman association stage** merges those detections into coherent 3-D tracks; **HRNet** recovers a complete 3-D pose for every athlete; dense **optical flow** augments the kinematic signal.

All streams are fed to a **spatio-temporal transformer encoder** that attends jointly to RGB patches, skeletal joints and trajectory features within a two-second window centred on any potential collision. The decoder produces three outputs: (i) a binary flag that illegal contact occurred, (ii) a **fine-grained foul type** (shooting, charge, reach-in, illegal screen, etc.), and (iii) the **identifiers of the fouling and fouled players**.

A **rule-aware layer** immediately verifies these predictions against explicit NBA criteria derived from the tracked positions—for example, whether both defender's feet lay outside the restricted area or whether verticality was preserved. Concurrence between neural and rule checks

confirms the foul; disagreement lowers confidence and flags the clip for human review, accompanied by a brief natural-language explanation such as "blocking foul: defender's left foot inside restricted zone".

Training uses **several seasons of fully annotated games**: every whistle, every no-call and all post-game corrections (Last Two-Minute Reports). Multi-view consistency losses and **contrastive ID objectives** enforce view-invariant representations. During deployment the system runs on a GPU cluster adjacent to the replay centre, sustaining $<$ **200 ms end-to-end latency**; officials receive the clip, an attention heat-map and the rule citation during a time-out. After each game, an automated report compares AI calls with the crew's decisions, yielding an **objective referee-accuracy score**.

In sum, accurate sensor coverage and robust compute enable a model that sees every angle, reasons with the rule book and justifies each verdict effectively.

9 Conclusion

In this work, we have demonstrated a practical pipeline for evaluating basketball refereeing by combining audio transcription and vision-based foul detection. Starting from raw broadcast footage, our system automatically extracts short clips centered on referee whistles, transcribes the audio with Whisper, and classifies each utterance via an LLM into `foul` or `clean`. In parallel, we apply RF–DETR to frames at 25 fps, smooth the detector scores, and merge consecutive *foul* frames into coherent video episodes. By aligning these two streams via a simple temporal IoU rule, we flag moments where the referee's call and the visual evidence agree, and—more importantly—where they diverge.

Our automatic referee–validation module exports three human-friendly artifacts: a CSV log of all episodes (with timestamps, model confidences, transcript snippets, and an agreement flag), compact storyboard panels of key frames for each disputed play, and a colour-coded timeline plot of the entire game. These outputs reduce a full-length match review from 90 minutes of manual scrubbing to approximately 25 minutes of focused audit, while surfacing every clear missed call and dubious whistle.

Despite its promise, our prototype faces limitations: it relies on a single broadcast view, processes frames in isolation without explicit pose or tracking data, and cannot encode the full nuance of official basketball rules. Moreover, audio transcription errors and visual false positives still occur, though at low rates. Addressing these gaps through hard-negative mining, minimal temporal context (e.g. 3-frame stacks), and targeted augmentation raises the prospect of incremental gains without major architectural overhaul.

Looking ahead, a next-generation system could incorporate multi-angle feeds, 3D pose estimation, rule-based verification layers, and end-to-end spatio-temporal transformers to deliver real-time, rule-aware foul adjudication. Such a full-scale design would partner neural perception with explicit game logic, yielding not only binary verdicts but also fine-grained foul types and natural-language justifications. Whether used as an in-official aid or a post-game analytics tool, this multimodal approach points toward a future where AI augments human referees, making high-stakes competitions fairer and more transparent.

In summary, by combining speech-to-text, large-language understanding, and state-of-the-art object detectors, our system provides a lightweight yet powerful referee review toolkit. It shows that even modest compute resources and publicly available models can transform how we audit and improve officiating in fast-paced team sports.

References

[1] S. Alashhab et al. “Hand gesture detection with convolutional neural networks”. In: *Distributed Computing and Artificial Intelligence, 15th International Conference*. 2018, pp. 45–52.

[2] A. Bialkowski et al. “Large-scale analysis of soccer matches using spatiotemporal tracking data”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 24 (2014), pp. 1513–1520.

[3] N. H. Dardas and N. D. Georganas. “Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques”. In: vol. 60. 11. 2011, pp. 3592–3607.

[4] S. Dobson et al. “Are football referees really biased and inconsistent? Evidence on the incidence of disciplinary sanction in the English Premier League”. In: *Journal of the Royal Statistical Society Series A* 170 (2007), pp. 231–250.

[5] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. “HMDB51: A Large Video Database for Human Motion Recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2011, pp. 2556–2563.

[6] H.-I. Lin et al. “Human hand gesture recognition using a convolution neural network”. In: *2014 IEEE International Conference on Automation Science and Engineering (CASE)*. 2014, pp. 1038–1043.

[7] W.-L. Lu, J.-A. Ting, J. J. Little, and K. P. Murphy. “Learning to track and identify players from broadcast sports videos”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.7 (2013), pp. 1704–1716.

[8] I. Mchale et al. “On the development of a soccer player performance rating system for the English Premier League”. In: *Interfaces* 42 (2012), pp. 339–351.

[9] G. Miao, G. Zhu, S. Jiang, C. Xu, and W. Gao. “A Real-Time Score Detection and Recognition Approach for Broadcast Basketball Video”. In: *Proceedings of the IEEE International Conference on Image Processing*. 2007, pp. 1691–1694.

[10] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal,

Patrick Labatut, Armand Joulin, and Piotr Bojanowski. *DINOv2: Learning Robust Visual Features without Supervision*. 2023.

[11] Isaac Robinson, Peter Robicheaux, and Matvei Popov. *RF-DETR*. https://github.com/roboflow/rf-detr. SOTA Real-Time Object Detection Model. 2025.

[12] Bishal Sadi Siddiqui, Zeeshan Ahmed Mridul, Zaki Habib, Ibrahim Sakib, and Md. Ahmarul Islam Chowdhury. "Real-Time Foul Detection in Football Matches Using Machine Learning Techniques". MA thesis. Brac University, 2024.

[13] K. Simonyan and A. Zisserman. "Two-stream convolutional networks for action recognition in videos". In: *arXiv preprint arXiv:1406.2199* (2014).

[14] S. Su, J. P. Hong, J. Shi, and H. S. Park. "Predicting Behaviors of Basketball Players From First-Person Videos". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[15] Yunjie Tian, Qixiang Ye, and David Doermann. "YOLOv12: Attention-Centric Real-Time Object Detectors". In: *arXiv preprint arXiv:2502.12524* (2025).

[16] Yunjie Tian, Qixiang Ye, and David Doermann. *YOLOv12: Attention-Centric Real-Time Object Detectors*. 2025. URL: https://github.com/sunsmarterjie/yolov12.

[17] T. Wang et al. "Abnormal event detection based on analysis of movement information of video sequence". In: *Optik* 152 (2018), pp. 50–60.

[18] Y. Wang, H. Jiang, M. S. Drew, Z.-N. Li, and G. Mori. "Unsupervised Discovery of Action Classes". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1654–1661.

[19] X. Wu et al. "A detection system for human abnormal behavior". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 1204–1208.

[20] Y. Zhang et al. "A sport athlete object tracking based on DeepSORT and YOLOv4 in case of camera movement". In: *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*. 2020.

[21] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. "DETRs Beat YOLOs on Real-time Object Detection". In: *arXiv preprint arXiv:2304.08069* (2024).

[22] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. “Deformable DETR: Deformable Transformers for End-to-End Object Detection”. In: *arXiv preprint arXiv:2010.04159* (2020).

GitHub repository link: https://github.com/vkalinovski/-Basketball_foul_detection