Basketbolda Oyuncu ve Top Gezingelerinin İkili Dönüştürücü ile Kestirimi
Dual Transformer-based Prediction of Player and Ball Trajectories in Basketball

Stavros Spyrou
Utrecht University
Utrecht, The Netherlands

Daniel Karavolos
Independent Researcher
Alkmaar, The Netherlands

Albert Ali Salah
Utrecht University
Utrecht, The Netherlands

***Özetçe* —Modern basketbolda gezinge kestirimi ve gerçek zamanlı geri bildirim sistemleri, oyun analizi, taktik planlama ve karar verme için faydalı araçlar haline gelmiştir. Bu çalışmamızda, basketbol maçları sırasındaki hareket ve bağlam bilgilerini girdi olarak kullanarak oyuncuların ve topun kısa vadeli hareketini kestiriyoruz. Bir spor analitik şirketi tarafından sağlanan ve 500 NBA maçından elde edilmiş verileri kullanıyoruz. Ham videolari, 60 Hz örnekleme hızıyla sahadaki on oyuncunun ve topun üç boyutlu konumlarını çıkarmak için işledik. Ön eğitim için maskelenmiş oto-kodlayıcı (masked autoencoder) kullanan bir çifte dönüştürücü (dual transformer) mimarisi önerdik ve bu mimariyle iyi sonuçlar elde ettik. Ayrıca, tahmin edilen gezingelerin gerçekçi göründüğünü doğrulamak için algısal bir çalışma yaptık.**

Anahtar Kelimeler—Spor analitiği, dönüştürücü modelleri, gezinge kestirimi, maskeli otokodlayıcı

***Abstract*—In modern basketball, trajectory prediction and real-time feedback systems have become useful tools for game analysis, tactical planning, and decision-making. Our paper is a contribution on short-term prediction of player and ball movement in professional basketball, using historical movement and contextual game information as input. We use data from 500 games from a professional basketball league, provided by a sports analytics company. The raw video feeds are processed to extract 3D positions of all ten players and the ball, sampled at 60 Hz. We propose a dual transformer architecture, using masked autoencoders for pretraining, and report very good results. We additionally conducted a perceptual study to verify that the predicted trajectories look realistic.**[1]

Keywords—Sports analytics, transformer models, trajectory prediction, masked autoencoder

I. Introduction

Trajectory prediction is important in modern basketball for game analysis, tactical planning, and decision support. In this paper we present a transformer-based framework for short-term prediction of player and ball trajectories in professional basketball. Using spatiotemporal tracking data from 500 games from a professional basketball league collected across two recent seasons, the proposed approach predicts the next 3 seconds of 2D motion for all ten players and the ball from the previous 5 seconds of observed gameplay. The framework combines masked autoencoder pretraining with transformer-based forecasting models that leverage positional, velocity, team-side, and temporal game-state information.

In addition to quantitative comparisons with state of the art models, a perceptual study was conducted to assess the realism of the predicted trajectories. These results suggest that pretrained transformer architectures are effective for modeling complex basketball motion patterns and provide a promising foundation for tactical analysis and hypothetical outcome visualization in professional basketball.

II. Related Work

Trajectory prediction is an important task in fields such as autonomous vehicles, pedestrian motion analysis, and sports analytics, but it is difficult as real-world settings are uncertain, interactions are complex, and multiple outcomes are often possible [1]. Early approaches were rooted in classical mechanics and deterministic models, which worked well in controlled settings but struggled with uncertainty and noise. A major step forward came with probabilistic methods such as the Kalman Filter, Gaussian Processes, and Particle Filters, which improved prediction in noisy and partially observable environments [2]. However, these methods still relied heavily on handcrafted assumptions and did not scale well to complex, high-dimensional multi-agent interactions. More recently, deep learning methods became dominant in trajectory prediction. Recurrent neural networks and long short-term memory models improved sequence modeling, while more recent transformer-based architectures and diffusion models further advanced the field by capturing long-range temporal dependencies [3], [4], [5], [6]. These developments are especially relevant in sports, where movement is strategic, interactive, and highly context-dependent.

[1]This is the uncorrected author proof that includes the Supplementary Material. Copyright with IEEE. Please cite as: Spyrou, S., D. Karavolos, A.A. Salah, "Dual Transformer-based Prediction of Player and Ball Trajectories in Basketball," IEEE 34th Signal Processing and Communications Applications Conference (SIU), Istanbul, 2026.

Basketball is a particularly challenging setting due to dense player interactions, rapid motion changes, contextual game effects, and the need for computationally efficient prediction. Existing basketball-specific work includes Baller2Vec++, which introduced a transformer-based framework for basketball tracking data [7], and HoopTransformer, which applied self-supervised learning to play representation learning [8]. More recent diffusion-based approaches, such as MADiff and Scalable Diffusion Transformer (SDT) model multiple plausible futures and report strong quantitative performance in multi-agent prediction [9], [5].

III. Methodology

Existing computer vision approaches can already determine the current positions of the players and the ball in real time. The objective of this work is to predict the future positions of basketball players and the ball, given the current state. Let $x^t \in \mathbb{R}^2$ represent the (x, y) court coordinates at time t, and $A = \{a_1, a_2, \ldots, a_{11}\}$ be the set of all agents (10 players + ball). A trajectory for agent $a \in A$ over T frames is defined as a sequence of 2D positions: $\tau^a_{1:T} = \{(x^a_t, y^a_t)\}^T_{t=1}$.

The model input consists of agent trajectories and contextual features over a past time window of $T_p = 5$ seconds $\times$ 10 Hz $= 50$ frames:

$$\mathcal{X} = \left\{ \tau^a_{t-T_p+1:t},\ v^a_t,\ c_t \right\}_{a \in A} \tag{1}$$

where v is the velocity, and c is the game context vector. The target output is the agent trajectories over the future window of $T_f = 3$ seconds $\times$ 10 Hz $= 30$ frames:

$$\mathcal{Y} = \left\{ \tau^a_{t+1:t+T_f} \right\}_{a \in A} \tag{2}$$

Following the literature, we define the observation window as $T_{\text{obs}} = 5$ seconds and the prediction window as $T_{\text{pred}} = 3$ seconds. All trajectories are sampled at 10 Hz, resulting in 50 observed frames and 30 predicted frames per sequence. Each input timestep may also include extra features such as velocity (v^t_x, v^t_z), player identity, team-side encoding, and contextual embeddings (e.g., game clock, match score). The output is either an absolute position or a displacement vector $\Delta x^t = x^t - x^{t-1}$, depending on the decoder design. Ground truth trajectories are available for all predicted sequences. Prediction accuracy is evaluated using Mean Squared Error (MSE). Average Displacement Error (ADE), Final Displacement Error (FDE), and their multimodal counterparts (minADE$_k$, minFDE$_k$) are also reported.

A. Prediction methodology

1) Masked Autoencoder (MAE) – Transformer for Player Trajectories: The player-specific trajectory prediction we propose consists of two interconnected stages. First, following [10], a Masked Autoencoder (MAE) is pretrained using segment-wise masking to learn representations of player trajectories, which include segment-wise masking applied independently to each agent's trajectory and a transformer encoder that separately applies attention over time and across agents. This design ensures that the model treats all agents equally, no matter what order they are input in. Specifically, the encoder first embeds and masks trajectory segments using a randomized shuffle mask, then reconstructs masked segments using denoising based on MSE. This approach is computationally efficient and does not rely on handcrafted features. Next, a transformer model is implemented to reuse the pretrained encoder from the MAE to predict future player positions based on historical movements and contextual data.

The model begins with an input embedding layer that encodes player positional coordinates (x, y), velocities (v_x, v_y), and team-side indicators into a high-dimensional latent space, enabling the representation of player dynamics and interactions. Context embeddings are also included and incorporate temporal features (game period, minute, second, shot clock), game state features (match scores), and ball representations obtained from the Ball MAE described in Section III-A2.

During pretraining, the model applies segment-wise masking to the player trajectories. These are divided into segments, and random segments are masked to encourage the model to reconstruct and learn robust trajectory patterns. To quantify the reconstruction accuracy, we use a MSE loss, which compares the model's prediction of masked segments with the actual ground truth positions.

The goal of the MAE pretraining stage is to reconstruct masked trajectory segments using contextual information. Let $x^t \in \mathbb{R}^2$ be the ground truth 2D position (x, y) of the target at time t, and let $\hat{x}^t$ be the predicted position. The masked segment m includes a range of timesteps from u to v, where the true positions are hidden from the model during encoding and must be reconstructed during decoding.

$$\mathcal{L}_{\text{MAE}} = \mathbb{E}_{m \sim \text{mask}} \left[\frac{1}{|m|} \sum_{t=u}^{v} \|\hat{x}^t - x^t\|_2^2 \right] \tag{3}$$

Here, $\|\cdot\|_2$ denotes the Euclidean norm. This loss promotes the learning of consistent short-term motion features and MSE has been shown to be highly effective for pretraining purposes [11].

The overall architecture follows a transformer encoder-decoder framework [12]. Stacked layers of multi-head self-attention and feedforward neural networks process the inputs. The encoder uses the pretrained encoder from the MAE to extract features from the historical trajectory data, while the decoder generates future trajectories based on these encoded representations and the contextual inputs.

At the center of the transformer is the self-attention mechanism, which determines how much each timestep in a sequence should focus on others. This is essential in modeling coordinated movement and temporal dependencies between players. The multi-head mechanism helps capture diverse temporal and spatial dependencies, such as player spacing, timing, and movement interactions. Finally, to ensure that the model focuses on relative motion rather than absolute position, we project the output to predict displacements relative to each player's current location. This mitigates the issue of "regression to the mean," where the model might predict average positions that rarely occur in real games[2].

[2]For more details, see the Appendix

2) Masked Autoencoder (MAE) – Transformer for Ball Trajectories: Ball trajectory prediction follows the same two-stage architecture used in the player-specific model (Section III-A1): a MAE pretraining phase followed by a transformer prediction phase. However, key differences exist in the inputs and contextual features tailored to the ball's movement. The input embedding layer encodes ball positional coordinates (x, y, z) and velocities (v_x, v_y, v_z) into a high-dimensional embedding space to effectively capture ball dynamics. Context embeddings include relative player positions (computed with respect to the ball's current position), temporal features (game period, minute, second, shot clock), and game state information (match scores). The output projection layer transforms the decoder's latent representations into predicted ball coordinates relative to the ball's previous position. This strategy enhances accuracy by predicting incremental displacements instead of absolute positions. All other architectural components remain consistent with the player-specific model.

3) Transformer Model Prediction Framework: To predict trajectories, the two models are applied in a step-by-step simulation, where player and ball trajectories are predicted alternately. At each timestep, the output of one model is used as contextual input for the other. This means that two passes are done for each timestep. The resulting trajectories are sequentially stored in a suitable format for downstream evaluation and visualization.

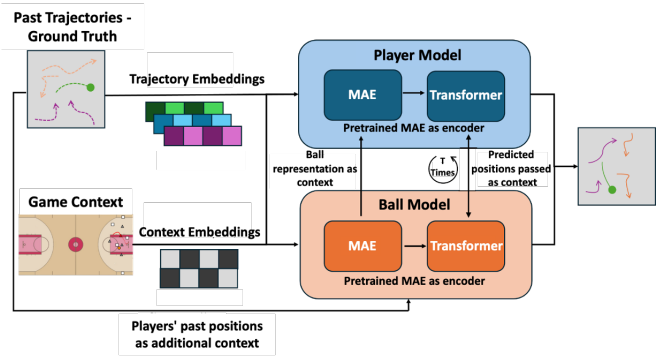

Figure 1: Overview of the proposed model

B. Data Preparation

We use player and ball tracking data from a professional basketball league, collected across two recent seasons and provided by a sports analytics company. The data are captured at 60 frames per second (fps) and include detailed positional and rotational information for players, the ball, and contextual game elements. We used 500 games, each consisting of approximately 350,000 frames before filtering and downsampling. The dataset is organized at the frame level, with each frame containing a complete snapshot of the game state (Table I). Every frame captures the positions of five players per team and the ball. This scale allows for accurate modeling of the players' and balls' motion patterns. No player-specific models are considered in this work. While individual player behavior (e.g. a fast moving guard vs a big rim protecting center) could lead to different movement trajectories, we focused on generalizable models that are applicable across players.

Category	Field	Description
Frame-Level Information	FrameCount	Sequential counter for frames
	TimestampUTC	Timestamp of the frame (in ms)
Player Data	Id	Unique ID for each player
	PoseJoints	3D coordinates (x, y, z) for 29 body joints
	Position	Global player position (x, y, z) in court space
	TeamSide	Indicates player's team
	JerseyNumber	Player's jersey number
	RotationalDataContext	Rotational data for joints
	HasBallPossession	Whether player possesses the ball
Ball Data	Position	3D coordinates of ball (x, y, z)
	BallState	Indicates if the ball is in play ("Alive")
	Possession	Team possessing the ball
Game Context	GameClockContext	Quarter, min., sec. remaining, game clock status
	ShotClockContext	Remaining shot clock time, running status
	Venue Information	Venue ID and name of the game location

Table I: Detailed structure of the dataset

To ensure the dataset is well-structured and optimized for model training, several preprocessing steps are applied. Linear interpolation is applied to fill gaps shorter than a predefined threshold (1 second). To improve computational efficiency and align with common practices in recent trajectory prediction studies [5], [7], [9], the data is downsampled to 10 Hz by retaining every sixth frame. To focus on meaningful game moments, only frames where the shot clock is running were included (with ± 2 window). Velocities for both players and the ball are computed using positional differences between consecutive frames, and added to modeling as context. To help with model training and evaluation, the dataset is split into individual "plays," defined as continuous sequences of frames with no gap larger than 50 frames. To ensure fair evaluation, the dataset is split by entire games rather than randomly sampled frames. This approach mimics the real-world deployment setting, where models are expected to predict future game scenarios without access to future data during training. Specifically, games are partitioned into three disjoint sets: 70% of games are used for training, 15% for validation (used for model tuning and early stopping), and 15% for final testing. We make sure that no plays overlap between the training, validation, and test sets.

Table II: Dataset split after filtering and downsampling

Split	#Games	#Unfiltered Frames (10 Hz)	#Frames (10 Hz)	#Plays
Train	350	~20,500,000	~9,100,000	~128,500
Validation	75	~4,400,000	~1,950,000	~27,900
Test	75	~4,400,000	~1,950,000	~27,400

C. Validation Approach

To benchmark model performance, we compare against several baselines and recent state of the art trajectory prediction models. We used Baller2Vec [7], a transformer-based architecture specifically designed for basketball players, as well as the Scalable Diffusion Transformer (SDT) [5], a modern diffusion-based trajectory predictor. For ball movement prediction, as Baller2Vec does not predict ball locations, we include an additional linear extrapolation baseline. All models undergo training and evaluation on an identical dataset, processed

uniformly according to the procedures and metrics outlined earlier. In addition, we performed a qualitative evaluation to assess the realism, interpretability, and practical relevance of the predicted trajectories. Participants were shown short, top down visualizations comparing real professional basketball trajectories with model generated trajectories over the same time window. The main task was to determine whether participants could distinguish real trajectories from generated ones and to collect subjective judgments of realism and smoothness.

IV. Experimental Results

Table III summarizes test-set results for player and ball prediction models. Lower is better for ADE/FDE and minADE/minFDE; higher is better for Top-1/Top-3. Note that the latter are only reported for models that predict on a discrete grid. The diffusion models and the linear baseline generate continuous trajectories, so these metrics are not applicable.

Table III: Quantitative results for player (top) and ball (bottom) trajectory prediction.

Model	**ADE** ↓	**FDE** ↓	**minADE@20** ↓	**minFDE@20** ↓	**Top-1** ↑	**Top-3** ↑
Transformer	0.27	0.37	0.10	0.10	0.30	0.45
SDT	1.91	2.78	1.13	1.12	–	–
Baller2Vec++	0.87	0.98	0.61	0.58	0.36	0.67
Transformer	1.30	2.10	0.56	0.54	0.44	0.75
SDT	2.52	2.91	1.87	1.40	–	–
Linear baseline	1.55	2.35	–	–	–	–

Table IV: Dataset split after filtering and downsampling

Split	Games	Unfiltered Frames (10 Hz)	Frames (10 Hz)	Plays
Train	350	~20.5M	~9.1M	~128.5K
Validation	75	~4.4M	~1.95M	~27.9K
Test	75	~4.4M	~1.95M	~27.4K

On the validation set, the proposed transformer approach performed much better than the competing approaches, consistent with the held-out test results. Diffusion models showed higher ADE on all splits, which may be expected since they optimize a heavier denoising objective and required significant fine-tuning. We note that the convergence of diffusion models is also much slower, and more noisy.

For the qualitative study, participants viewed 3 second top-down clips of 2D trajectories in two separate tests: a player test and a ball test. Each test contained five items, where each item displayed one real and one generated clip for the same time window. Participants selected which clip they believed was real and provided ratings of realism, smoothness, confidence and for player clips, tactical plausibility. Optional open-ended comments were also collected. A total of $N = 27$ participants completed the study. For player trajectories, participants achieved 0.60 accuracy across 135 judgments, only slightly above chance, suggesting that generated player motion was often difficult to distinguish from real motion. For ball trajectories, accuracy increased to 0.72. However, we note that we did not smooth ball trajectories, and did not ensure that they leave from player positions. Qualitative feedback showed that participants mainly noticed jitter, overly constant motion, unrealistic direction changes, and tactically implausible movement patterns. Figure 2 shows how the play instance is visualized.

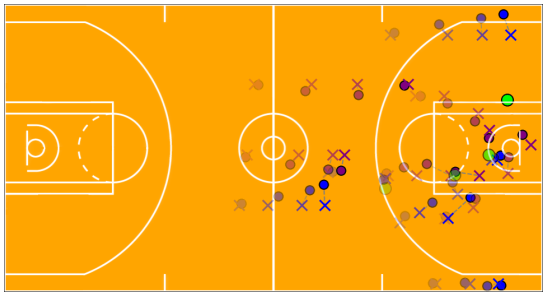

Figure 2: Illustration of multi-step player trajectory rollout on the basketball court as a history image. Circles indicate ground-truth, while crosses indicate the model predictions.

V. Discussion and Conclusions

The proposed transformer achieves strong quantitative results, but errors arise from small step-wise inaccuracies that accumulate over the three-second rollout (i.e. trajectory drift). Unusual situations such as deflections or broken plays are rare in training data, so the model performs less reliably in these cases. In addition, when the ball drives the action, predictions tend to lag if the model is not given sufficiently strong information about the relative ball-player configuration. Larger player transformers and diffusion variants also showed sensitivity to training stability, and longer training, gradient clipping, and steadier learning rate schedules generally improved performance. Even with low error scores, some predictions may appear unrealistic on visual inspection. The models are trained to minimize MSE, which favors safe, smoothed trajectories over decisive cuts or passes. Another limitation is that the inputs do not fully capture intent or vertical ball behavior. These effects are most noticeable during fast ball movement and towards the end of rollout. The perception study suggests that strong ADE/FDE scores do not guarantee realistic-looking motion, confirming similar findings in the literature [13].

References

[1] C. Barata *et al.*, "Sparse motion fields for trajectory prediction," *Pattern Recognition*, vol. 110, p. 107631, 2021.

[2] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[3] C. Jiang *et al.*, "Motiondiffuser: Controllable multi-agent motion prediction using diffusion," in *Proc. CVPR*, 2023, pp. 9644–9653.

[4] G. Capellera *et al.*, "Footbots: A transformer-based architecture for motion prediction in soccer," in *Proc. ICIP*, 2024, pp. 2313–2319.

[5] S. Zhang *et al.*, "Multi-agent trajectory prediction with scalable diffusion transformer," in *Proc. DAI*, 2024.

[6] A. Graser, "MobilityDL: a review of deep learning from trajectory data," *GeoInformatica*, vol. 29, no. 1, pp. 115–147, 2024.

[7] M. A. Alcorn and A. Nguyen, "baller2vec: A multi-entity transformer for multi-agent spatiotemporal modeling," *arXiv preprint arXiv:2102.03291*, 2021.

[8] X. Wang *et al.*, "Hooptransformer: Advancing NBA offensive play recognition with self-supervised learning from player trajectories," *Sports Medicine*, pp. 1–11, 2024.

[9] Z. Zhu *et al.*, "Madiff: Offline multi-agent learning with diffusion models," *arXiv preprint arXiv:2305.17330*, 2023.

[10] Y. Rudolph and U. Brefeld, "Masked autoencoder for multiagent trajectories," *Machine Learning*, vol. 114, 2025.

[11] K. He *et al.*, "Masked autoencoders are scalable vision learners," in *Proc. CVPR*, 2022, pp. 16 000–16 009.

[12] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NeurIPS*, 2017.

[13] Y. Fu *et al.*, "MoFlow: One-step flow matching for human trajectory forecasting via implicit maximum likelihood estimation based distillation," in *Proc. CVPR*, 2025.

APPENDIX

This appendix provides additional implementation and experimental details for the models used in the main paper, as a Supplementary Material. Due to space constraints, such details were omitted from the main paper.

A. Proposed Model Configuration

Table V summarizes the final configuration of the two proposed Transformer models. The same general MAE-initialized Transformer design is used for player and ball prediction, but the output discretization differs between models.

Table V: Configuration of the proposed Transformer models.

Setting	Player Transformer	Ball Transformer
Pretraining	Segment-wise MAE	Segment-wise MAE
Transformer layers	6	6
Attention heads	8	8
MAE masking ratio	30%	30%
Output type	Relative displacement grid	Relative displacement grid
Grid size	11×11	20×20
Cell size	0.10 m	0.05 m
Trainable parameters	$\sim$17.4M	$\sim$17.5M
Training time	$\sim$6.5 hours	$\sim$4.0 hours
Inference time per 30-frame rollout	$\sim$50 ms	$\sim$50 ms

The final configuration was selected after tuning Transformer depth, attention heads, embedding size, feed-forward size, masking ratio, and batch size.

B. Baseline Configuration

Table VI summarizes the baseline settings used for the comparison with Baller2Vec++ and SDT [7], [5]. All baselines follow the same preprocessing, train/validation/test split, observation window, prediction horizon, and evaluation metrics as the proposed models.

Table VI: Baseline models and comparison settings.

Model	Task	Output type	Main settings
Baller2Vec++	Player	Grid / trajectory prediction	Re-implemented basketball Transformer baseline
SDT	Player and ball	Continuous trajectories	50 denoising steps, unguided sampling
Linear extrapolation	Ball	Continuous trajectories	Constant-velocity extrapolation

Baller2Vec++ is included only for player trajectory prediction because it is designed for player movement and does not predict ball locations [7]. For ball prediction, the linear extrapolation baseline assumes that the recent ball velocity remains constant over the prediction horizon. The SDT baseline is evaluated for both player and ball prediction using 50 denoising steps with unguided sampling [5].

C. Computational Resources and Cost

All models were implemented in PyTorch and trained using GPU acceleration. The main experiments were run on an NVIDIA A100 GPU using mixed-precision training. Validation ADE was used for model selection and early stopping. Exact floating-point operation counts (FLOPs) were not directly measured. Therefore, computational cost is reported using: model size, training time, inference time, architecture depth, number of attention heads, output-grid size, number of diffusion sampling steps, hardware, and training setup. The final player Transformer contains approximately 17.4M trainable parameters and required approximately 6.5 hours to train on an NVIDIA A100 GPU. The final ball Transformer contains approximately 17.5M trainable parameters and required approximately 4 hours to train on the same hardware. Inference for one full 30-frame rollout required approximately 50 ms for both the player and ball models.

D. Player and Ball Results

Tables VII and VIII report the quantitative results separately for player and ball trajectory prediction. Top-1 and Top-3 are only reported for grid-based models. Continuous-output models such as SDT and the linear baseline do not produce grid-class probabilities, so these metrics are not applicable. Similarly, minADE@20 and minFDE@20 are not reported for the deterministic linear baseline because it does not generate multiple trajectory samples.

Table VII: Player trajectory prediction results. Lower is better for ADE/FDE and minADE/minFDE; higher is better for Top-1/Top-3.

Model	ADE ↓	FDE ↓	minADE@20 ↓	minFDE@20 ↓	Top-1 ↑	Top-3 ↑
Transformer	0.27	0.37	0.10	0.10	0.30	0.45
SDT	1.91	2.78	1.13	1.12	–	–
Baller2Vec++	0.87	0.98	0.61	0.58	0.36	0.67

Table VIII: Ball trajectory prediction results. Lower is better for ADE/FDE and minADE/minFDE; higher is better for Top-1/Top-3.

Model	ADE ↓	FDE ↓	minADE@20 ↓	minFDE@20 ↓	Top-1 ↑	Top-3 ↑
Transformer	1.30	2.10	0.56	0.54	0.44	0.75
SDT	2.52	2.91	1.87	1.40	–	–
Linear baseline	1.55	2.35	–	–	–	–

E. Possible Accuracy Improvements

Several directions could further improve both player and ball trajectory accuracy. First, richer contextual inputs could be added, such as explicit possession labels, on-ball/off-ball role indicators, screen and cut cues, defensive pressure, and stronger relative ball-player context. For ball prediction, vertical information such as approximate height or pass-arc indicators could help distinguish bounce passes, lob passes, and shots, which are difficult to separate in the current 2D formulation. Second, training stability and prediction quality could be improved through longer warm up schedules, gradient clipping, steadier learning rate schedules, and scheduled sampling. Finally, performance on rare events could be improved by adding or oversampling examples such as deflections, traps, loose balls, and long diagonal passes.