Predicting Roulette: A Deep Learning Approach

Arthur Robertson

Bachelor of Science in Computer Science and Mathematics
The University of Bath
2025

Predicting Roulette: A Deep Learning Approach

Submitted by: Arthur Robertson

Copyright

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Abstract

Roulette is a physically deterministic but chaotic system, where small differences in initial conditions can lead to unpredictable outcomes. This project investigates whether deep learning can extract and forecast meaningful aspects of such a system using only raw, noisy video footage. A computer vision pipeline was developed to detect and track roulette ball and wheel motion from publicly available footage, correcting for perspective distortion and visual noise. Using this structured trajectory data, Long Short-Term Memory (LSTM) models were trained to predict system dynamics based solely on early-stage visual input. Results show that these models consistently outperform naïve baselines and improve in accuracy as more of the system's initial behaviour is observed. The findings demonstrate that deep learning can recover short-term predictive structure from chaotic physical processes, offering a general approach to modelling complex systems where traditional physics-based methods are infeasible or unavailable.

Contents

Chapter 1

Introduction

1.1 Context: Roulette as a Chaotic System

Roulette is a deterministic mechanical system with two core components: a ball revolving around a sloped outer track, and an inner wheel rotating in the opposite direction containing the iconic numbered pockets [51]. Although governed by classical physics, roulette exhibits chaotic behaviour. Small differences in starting conditions, such as ball speed or wheel friction, can lead to drastically different outcomes after collisions with deflectors and frets, making precise long-term prediction intractable [32]. This sensitivity to initial conditions is a hallmark of chaos [14].

Figure 1.1: A photo of a real roulette wheel in a casino setting. Photo from [21].

Each full sequence of the ball and wheel rotating before the outcome is determined is referred to as a *spin* [51]. A critical moment in the spin is the drop-off event, when the ball leaves the outer rim and enters the inner wheel area, either by hitting a deflector or descending directly toward the pockets [47, 51]. This transition marks a fundamental change in the ball's motion. After drop-off, the ball's movement is driven by complex interactions with the deflectors and pockets, making precise prediction extremely difficult. However, studies have shown that detailed knowledge of the system at the moment of drop-off can significantly improve understanding of the eventual outcome [47]. While full physical modelling remains out

of reach, information about the drop-off event helps constrain the uncertainty of the chaotic phase that follows.

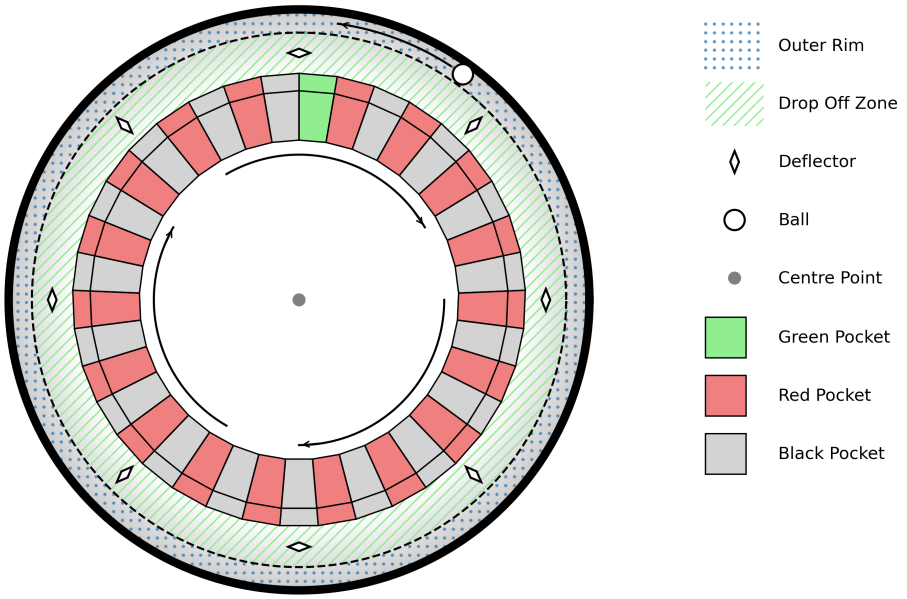


Figure 1.2: A diagram of a roulette wheel. The inner wheel, which rotates clockwise, contains the green, red, and black pockets. The ball is launched anticlockwise from the sloped outer rim and eventually slows and begins to fall in the drop-off zone. There, it may hit a diamond-shaped deflector or pass between them before falling into a pocket in the rotating wheel.

This idea was first exploited in the 1960s by Edward Thorp and Claude Shannon [4, 55, 57], who built rudimentary computers to predict roulette outcomes based on direct physical measurements and detailed physics models . Their work showed that even chaotic systems can be partially tamed if the critical transition point is known.

In contrast, this research investigates whether deep learning can predict drop-off dynamics directly from raw, noisy video footage, without any physical measurements, mechanical models, or manual calibration. Instead of encoding the system's physics by hand, the model must infer predictive structure purely from observational data.

This approach reflects the broader shift from theory-driven to data-driven science [30, 44], with roulette offering a compact proving ground where chaos, noise, and partial observability intersect [51].

1.2 Research Problem and Approach

This research focuses on predicting key aspects of roulette spin dynamics at the moment of drop-off, including the precise timing of the event. The target variables are:

- **Drop-off Time**: The number of frames from the start of an input window to when the ball enters the inner wheel area and starts to drop-off.
- **Ball Angular Trajectory**: The path traced by the ball as it spins around the outer rim, up until the drop-off, expressed as a sequence of angular positions.

- **Wheel Angular Trajectory**: The angular position of the rotating inner wheel, contextualising the ball's path (using the green zero pocket as a reference point).

The aim is to make predictions purely from what can be seen during the early stages of each spin, using public YouTube footage with no access to physical measurements or wheel data. To explore how early predictions can be made, the amount of video frames provided as input is varied between 2 and 10 seconds worth. The goal is to assess whether longer clips improve predictive accuracy by providing more visual information before drop-off.

Several limitations in the available data complicate prediction:

- No Ground Truth Labels: Whilst video footage is available, it lacks any annotations for drop-off times or angular positions; all target information must be inferred algorithmically.
- Visual Noise: Motion blur, occlusions, and lighting variation introduce tracking difficulties across frames [24].
- Perspective Distortion: Oblique and inconsistent camera angles (typically 20-30 degrees from vertical) cause geometric distortion, requiring normalisation [66] to compare different videos and ensure consistent trajectory extraction.

To address these challenges, the project builds and evaluates a data-driven pipeline with the following core objectives:

1. Acquire relevant footage from YouTube and split the continuous frames into individual spins.
2. Detect and track key elements in each frame.
3. Generate a structured dataset of angular trajectories.
4. Train deep learning models to predict drop-off time and angular trajectories.
5. Evaluate performance as a function of input time length, denoted as T.
6. Compare against naïve baselines and assess performance and limitations.

This work investigates whether machine learning, applied to a self-constructed dataset of real-world spins, can overcome these constraints to generate physically consistent predictions.

1.3 Scope of Project

This research focuses exclusively on predicting events up to the drop-off point, defined as the moment when the roulette ball exits the stable outer rim and enters the chaotic inner wheel. Prediction beyond this point is deliberately excluded. While it is possible to model post-drop-off behaviour statistically [47] by estimating landing distributions based on drop-off location, doing so would shift the task from deterministic trajectory prediction to stochastic outcome modelling. This would also require additional data collection, including detailed mapping of deflectors and the physical wheel geometry, which is intentionally avoided here in favour of tackling the more challenging and general problem of learning from raw, noisy visual data alone.

The pipeline is fully data-driven, with all predictive structure learned directly from visual observations from YouTube. No attempt is made to model or understand the underlying physics, by design, introducing a degree of black-box behaviour typical of modern machine learning approaches. Real-time deployment and gambling applications are explicitly excluded from the aims of this academic work.

1.4 Motivation, Significance, and Contributions

Predicting the behaviour of chaotic physical systems from raw observational data is a fundamental challenge across science and engineering [14]. Traditional approaches often rely on detailed physical modelling and controlled environments, but many real-world systems are too complex or poorly understood to model directly [40]. This motivates the use of data-driven methods [30] capable of extracting structure from noisy, incomplete observations [7].

Roulette is a useful case study for these challenges. If machine learning can extract predictive information from raw video footage of roulette spins, it demonstrates broader potential for vision-based modelling of chaotic systems more generally. This study is among the first to explore roulette outcome prediction using machine learning, with the added challenge of relying solely on raw video data.

This research makes the following contributions:

- Demonstrates that accurate tracking of ball and wheel trajectories is possible from uncontrolled, real-world video footage.
- Introduces a scalable pipeline for constructing labelled datasets from noisy video sources.
- Develops and evaluates machine learning models for predicting drop-off time and trajectories based solely on early-phase visual input.
- Provides empirical evidence on how the amount of early information (input window size) affects prediction performance.

These contributions offer a preliminary template for applying machine learning to chaotic, data-limited environments. Although focused on roulette, the underlying methods are applicable across domains where direct physical measurements are difficult, from structured settings like sports analytics to other complex, dynamic systems found in nature and engineering [44, 53].

1.5 Hypotheses

Building on the motivations and contributions outlined above, this research tests three central hypotheses related to the challenges of tracking and predicting roulette dynamics from real-world video footage:

- **H1**: A vision-based pipeline can track the ball and wheel across the full duration of a spin with high accuracy even from noisy, uncontrolled video footage.
- **H2**: Machine learning models trained on early-phase trajectory data outperform naïve baselines in predicting drop-off time and angular trajectories.
- **H3**: Prediction accuracy improves as the input window length T increases, reflecting the value of additional early-phase information.

1.6 Structure of the Research

The remainder of this project is organised as follows:

- **Chapter 2** reviews related work on roulette physics, chaos theory, machine learning for physical prediction, and computer vision techniques relevant to tracking motion from video.

- **Chapter 3** details the construction of the dataset, including footage selection, splitting, and trajectory extraction methods.
- **Chapter 4** describes the model architectures, input formats, and experimental setup used to predict drop-off dynamics from early-phase visual data.
- **Chapter 5** presents the results, evaluating model performance across varying input lengths and comparing against naïve baselines.
- **Chapter 6** discusses the findings, their limitations, ethical considerations, and potential directions for future work.
- **Chapter 7** concludes the research, summarising the main contributions and outlining broader implications for machine learning on chaotic systems.

A glossary of technical terms and abbreviations used throughout the research is available at Appendix A.

Chapter 2

Literature, Technology and Data Survey

2.1 Chaos and Predictability in Physical Systems

Chaos refers to a class of deterministic systems whose outcomes, while governed by precise rules, exhibit extreme sensitivity to initial conditions, making long-term prediction fundamentally challenging [14, 33]. This hallmark of chaos means that small, often imperceptible differences in starting states can lead to vastly divergent trajectories. Famously illustrated by Edward Lorenz's "butterfly effect" [32], this principle has shaped the modern understanding of complex systems.

These behaviours often emerge in nonlinear dynamical systems, where variables interact in interdependent, feedback-driven ways [14, 33]. Examples include weather systems [32], planetary orbits [29], double pendulums [50], and, crucially for this work, roulette [51]. In such systems, long-term prediction becomes unstable not because of inherent randomness, but because of exponential sensitivity to tiny differences in initial conditions or measurement errors [36].

This exponential divergence can be quantified using a Lyapunov exponent [61], which measures the average rate at which nearby trajectories in state space move apart. A positive Lyapunov exponent indicates chaos: the larger the value, the more rapidly predictive accuracy deteriorates over time.

However, chaos is not synonymous with randomness. Many chaotic systems exhibit local predictability, with short periods during which their evolution is more structured and inferable [17]. Within this local window, it is possible to make meaningful short-term forecasts before instability takes hold [9, 14]. This principle is essential to the strategy employed in this research.

Roulette is a prime example of the balance between order and chaos [51]. While initially governed by deterministic forces like gravity, friction, and angular momentum, the system transitions into unpredictability with the ball entering the chaotic region of deflectors and pockets [47]. This research targets the "edge of chaos", the final moments where structured dynamics still prevail and prediction remains physically meaningful. Local forecasting of chaotic behaviour fits into broader efforts to model complex systems [38, 39, 60], with machine learning providing an effective tool for extracting short-term structure from apparent disorder.

2.2 The Physics of Roulette

Roulette consists of a spinning inner wheel and a ball launched along a sloped outer track. Early in its trajectory, the ball travels smoothly along the rim under the influence of classical forces [51]. As it slows, it drops onto the wheel, either bouncing off deflectors or falling directly into the pockets. At this point, the ball's motion becomes highly sensitive to small variations, and chaos quickly dominates. The drop-off moment, when the ball departs from the rim, marks the last opportunity for meaningful physical analysis before predictability collapses [47].

Edward Thorp and Claude Shannon were among the first to exploit this critical phase to infer the ball's behaviour post-drop-off. In the 1960s, they developed the world's first wearable computer [4, 57] that, using manually timed observations, estimated the ball's likely landing region based on its velocity and the precise moment it detached from the rim. Although successfully tested in casinos, their system was calibrated under controlled conditions and relied heavily on a physics-based model to bridge the chaotic transition [55].

Small and Tse [47] pursued a similar strategy, using simplified physical equations alongside a high-speed overhead camera to automate motion capture. Like Thorp and Shannon [57], their approach modelled the ball's pre-drop-off trajectory and then applied statistical inference to predict the likely landing zones once chaotic dynamics took over. Both works demonstrated that accurately estimating the state at drop-off can yield substantial predictive power, with Small and Tse reportedly achieving a betting edge at least as high as $+18$ percent in a casino [47]. This stands in stark contrast to the typical house edge of approximately -2.7 percent in European roulette [56]. However, while the final prediction results are reported, the authors do not provide any implementation details or analysis of the ball's pre-drop-off trajectory accuracy, remaining largely theoretical. This absence of intermediate evaluation and practical transparency limits the ability to assess the reliability of their estimates or to make meaningful performance comparisons.

Nonetheless, their success confirms a key insight: even in chaotic systems, local structure around the transition point can be exploited for meaningful prediction [47, 55]. This research builds on that foundation by focusing specifically on refining the drop-off estimation itself, aiming to maximise the quality of information extracted at the last moment before chaos overwhelms the system. Rather than relying on controlled conditions and explicit physics, we explore whether data-driven learning from uncontrolled video can achieve similarly strong results [30].

2.3 Machine Learning as an Alternative to Physics-Based Modelling

Many physical systems, including roulette, have traditionally been studied through direct physical modelling [51, 56]. Classical approaches attempt to predict outcomes by measuring system parameters precisely and applying known equations of motion. While effective under controlled conditions, this strategy becomes impractical when faced with noisy, partial, or chaotic observations. Small uncertainties, unmeasured factors, or unpredictable interactions can quickly cause classical models to fail or degrade [14, 47].

Machine learning offers a fundamentally different approach. Rather than relying on known physics, models can be trained to infer predictive patterns directly from observed data, even

when the underlying system is only partially understood or partially observable [7]. This shift from theory-driven to data-driven inference [30] has enabled major advances across fields such as weather forecasting [15], fluid dynamics [8], and geosciences [49], where full physical modelling is intractable.

Deep learning methods, in particular, have shown strong performance in extracting structure from complex, noisy time series [22, 30, 44]. By learning representations of dynamic behaviour directly from raw input data, deep learning models can predict system evolution without requiring explicit knowledge of the governing laws [38, 39, 60].

This research adopts the same philosophy. Instead of attempting to model roulette spin dynamics through first principles physics, it investigates whether deep learning can extract meaningful predictive signals directly from noisy video observations, in a domain where physical modelling is impractical.

2.4 Learning Chaotic Dynamics with Sequence Models

Predicting the pre-drop-off dynamics of roulette requires learning complex temporal patterns from noisy, partially observed video input. The task consists of two distinct targets: a scalar (the time until drop-off) and two sequences (the angular trajectories of the ball and wheel). Both must be inferred from variable-length sequences of early motion, capturing short-term transitions and longer-term deceleration trends.

Recurrent neural networks (RNNs) [46] are a natural fit for sequence modelling tasks [31, 54]. Among these, Long Short-Term Memory (LSTM) [26] networks are particularly appropriate. LSTMs address the vanishing gradient problem that hampers standard RNNs [6], enabling the learning of dependencies across extended temporal horizons [23]. They have demonstrated strong performance in noisy, partially observed domains such as pedestrian trajectory forecasting [2], sports analytics [62], and chaotic physical systems like the double pendulum and Lorenz attractor [10, 39, 60]. These characteristics align well with the needs of this project.

Empirical comparisons reveal different trade-offs between architectures [3]. Gated Recurrent Units (GRUs) [11] offer a computationally lighter alternative to LSTMs but often struggle to capture complex temporal patterns, making them less suitable for tasks requiring fine-grained motion forecasting. While powerful, Transformer models [59] typically demand larger datasets than LSTMs and risk overfitting in noise-prone, data-limited settings. For this project's focus on relatively short-term dynamics from observational data, LSTMs offer a potentially more data-efficient and robust alternative.

Physics-informed neural networks (PINNs) [41] incorporate physical laws, typically expressed as differential equations, directly into the training process. By constraining the model with known dynamics, they offer strong performance in domains where such equations are well understood and tractable. However, this research deliberately adopts a different strategy: it explores whether predictive power can emerge purely from observation, without embedding prior physical knowledge into the model. This avoids the need for explicit formulations of the system's dynamics and allows us to evaluate the capacity of sequence models to learn directly from data, even in settings where the underlying physics may be partially unknown, complex, or difficult to formalise.

It is also acknowledged that LSTMs are not without limitations. They can struggle to model

extremely long-range dependencies and are susceptible to compounding errors over long prediction horizons [31]. However, given the moderate data availability and the broader project goals, LSTMs offer the best balance between flexibility, robustness, and practical feasibility [26]. They were therefore chosen as the primary architecture for this project.

2.5 Computer Vision for Dynamics from Video

Extracting physical dynamics from video requires robust object detection and tracking in noisy, uncontrolled conditions. This remains a significant challenge in computer vision [48].

Traditional motion analysis methods, such as optical flow and keypoint tracking, assume small, smooth inter-frame motion and consistent visibility of tracked objects [5]. They perform well under clean, high-resolution conditions, but degrade when these assumptions are violated [52]. In this project's footage, the roulette ball is extremely small, moves rapidly, suffers from heavy motion blur, and is frequently occluded by the wheel structure or croupier hands. These factors disrupt optical flow estimation and cause classical tracking algorithms to fail.

To address this challenge, the project avoids conventional motion estimation techniques and instead applies deep learning-based object detection to extract structured signals from each frame [67]. By detecting objects independently in each frame, the method achieves reliable localisation of the ball and wheel, even under blur, occlusion, or erratic motion. This yields clean, physically meaningful trajectories suitable for downstream prediction.

For object detection, the *You Only Look Once* (YOLO) [42] family of models is adopted. YOLO is a deep learning framework known for its speed and accuracy in detecting objects under challenging conditions, including motion blur, occlusion, and lighting variation [43]. Crucially, it processes each video frame independently, making it well suited for uncontrolled roulette footage where frame-to-frame continuity cannot be relied upon. Its ability to learn spatial features directly from pixel data enables robust detection of small, fast-moving objects like the roulette ball.

Early YOLO models produced only bounding boxes around detected objects. More recent variants, such as YOLOv11 [27] and YOLOv12 [58], support instance segmentation, generating pixel-level masks [25]. This is particularly valuable in the context of this project, where the ball is extremely small and often adjacent to visually similar features like deflectors or frets. Bounding boxes alone tend to yield imprecise or ambiguous centres, whereas instance segmentation enables tighter fitting to the ball's shape. This leads to more accurate centroid estimation, reducing spatial noise and improving the quality of the extracted trajectories.

Following object detection, post-processing techniques are commonly used in video-based physical inference to refine motion signals.

Homography correction is a standard technique for addressing perspective distortion in footage captured from oblique or off-axis camera angles [24, 66]. A homography is a projective transformation that maps points between two planes while preserving straight lines, enabling reconstruction of a top-down or canonical view from angled observations. In domains such as sports analytics [13] and autonomous vehicle tracking [63], homographies are commonly used to align camera views to a standard reference frame, allowing consistent measurement of positions and trajectories. In this project, where the footage involves varied and uncontrolled camera angles, homography correction is essential to transform the distorted elliptical path of

the ball into a consistent circular reference frame. This normalisation enables reliable extraction of angular position and velocity across videos, forming a unified dataset suitable for sequence modelling.

Temporal smoothing reduces high-frequency jitter caused by detection noise by averaging positions across neighbouring frames, while linear interpolation fills short gaps resulting from occlusions or missed detections [20, 34]. Together, these techniques enhance trajectory continuity and stability, making the data more suitable for downstream dynamic modelling. Such preprocessing is standard when extracting structured motion features from noisy or partially observed video.

2.6 Research Gap and Framing

Despite the long-standing interest in roulette as a testbed for chaotic dynamics, no known prior work has applied modern machine learning techniques to predict its behaviour directly from video. Existing approaches have relied on physical modelling and highly controlled conditions [47, 55]. These often involve laboratory setups, high-speed cameras, or direct measurements of ball and wheel properties. While effective under idealised circumstances, these methods are infeasible in most real-world contexts.

There is currently no publicly available dataset of labelled roulette trajectories extracted from video footage, which makes it difficult to evaluate or replicate end-to-end machine learning approaches. In addition, many prior studies provide limited implementation details, making reproduction challenging. Significant differences in both methodology (observational machine learning versus controlled, physics-based modelling) and data sources (incidental video versus direct measurements or high-speed cameras) further complicate direct quantitative comparisons with earlier work.

In light of these challenges, this research explores whether meaningful predictive signals can be extracted from incidental video footage, recorded without calibration, control, or physical instrumentation. By predicting scalar and sequential outcomes (drop-off time and angular trajectories) from unstructured observations, the work presents a fully data-driven pipeline that sidesteps the need for physical modelling entirely.

Beyond roulette, this project serves as a case study in observational inference for chaotic systems. It tests the extent to which machine learning can model the dynamics of real-world physical processes using only pixel-level information. Success in this domain would not only push forward the specific challenge of roulette prediction, but also highlight the wider potential of data-driven methods in tackling complex systems where direct measurement is hard and theory-based modelling is either unworkable or entirely absent [44].

The next chapter covers the practical implementation of this method, starting with the construction of a dataset and the development of a vision pipeline to extract and normalise motion trajectories.

Chapter 3

Dataset Creation with YOLO

This chapter outlines the creation of a structured dataset from publicly available YouTube footage. A complete pipeline was developed to extract motion data while addressing key challenges such as motion blur, occlusion, and perspective distortion. Figure 3.1 provides an overview of the pipeline. The resulting dataset underpins the predictive modelling in Chapter 4 and supports the evaluation of Hypothesis H1 in Chapter 5.

To test H1 and ensure the reliability of the extracted data, the computer vision pipeline will be formally evaluated on four key aspects:

- **Detection Accuracy**: Measures how effectively YOLO detects the roulette ball and wheel in each frame, using precision, recall, and mean average precision (mAP).
- **Tracking Quality**: Assesses the frame-to-frame consistency and accuracy of detected object positions by comparing extracted coordinates against ground truth annotations.
- **Geometric Correction Robustness**: Evaluates the pipeline's ability to normalise object trajectories across varying spins and camera perspectives.
- **Spin Splitting Reliability**: Measures how accurately the pipeline identifies the start and end points of individual spins within continuous footage.

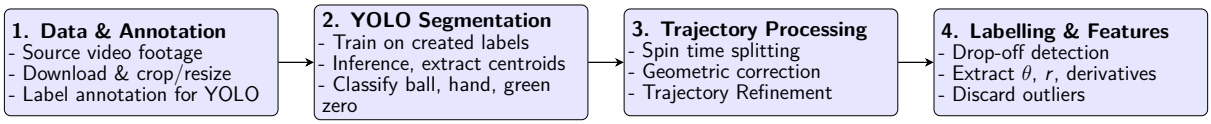


Figure 3.1: An overview of the dataset creation pipeline: from raw video download and annotation, through YOLO segmentation and trajectory processing, to final labelling and feature extraction.

3.1 Video Data Acquisition and Preprocessing

To construct the dataset we first sourced footage from 'The Roulette Channel' [16] on YouTube, a public archive offering approximately 100 hours of uncut roulette spins across various wheel models. Permission to use this footage for research was sought and granted, as detailed in Appendix B. From this archive, we selected 42 videos (30 hours) to capture a diverse range of visual conditions, balancing dataset diversity with the practical constraints of preprocessing and analysis. The modular design allows straightforward adaptation to additional configurations in future work.

(a) Frame from a video with a more vertical viewing angle. While cleaner, branding overlays, high contrast, and motion blur affect segmentation.

(b) Frame from a video with a steep off-axis camera angle. The reflections introduce visual noise, complicating object detection.

Figure 3.2: Example frames (cropped and resized) from the dataset showing the range of visual conditions. These illustrate the variability in perspective, lighting, occlusion, and quality across different videos.

All videos are recorded in 1080p at 50 FPS with fixed camera angles (estimated 20-30° from vertical). Perspectives vary across the 42 selected videos, creating diverse visual conditions. Each video shares a similar layout: a leaderboard on the left and a view of the wheel on the right. Spins occur roughly every 30 to 45 seconds, suggesting an estimated total of around 3,500 potential spins across the 30 hours of footage. Up to the first 20 seconds of each, before the ball settles in a numbered pocket, are our focus.

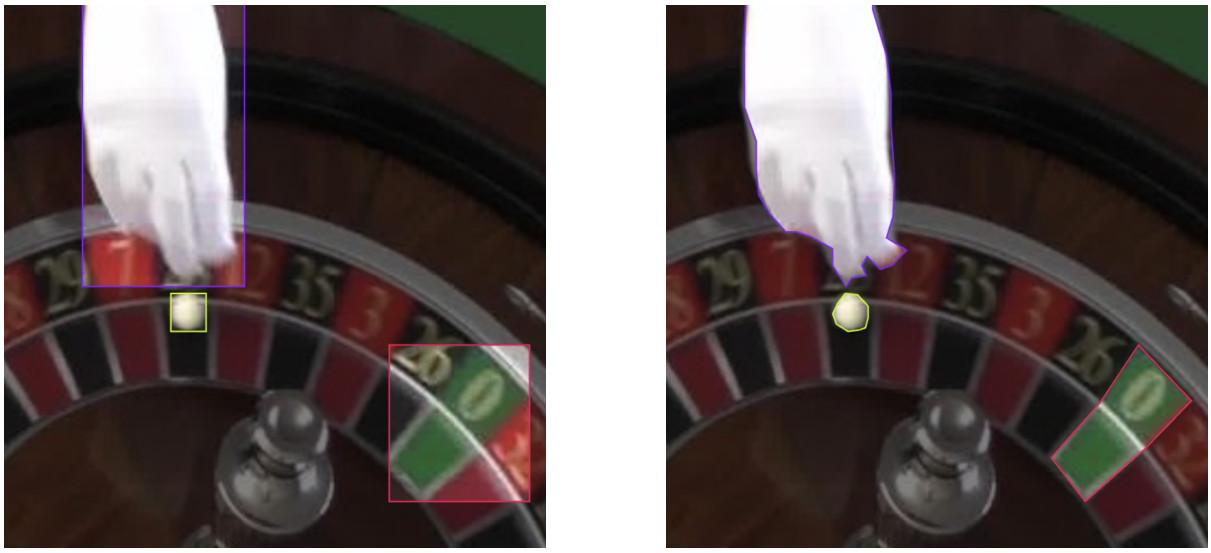
The videos introduce significant visual noise. The off-axis camera angle often hides the ball behind the wheel's edge for extended periods during each spin, making detection difficult (see Figure 3.2). Inconsistent lighting, motion blur, croupier hand occlusions, and occasional on-screen adverts further hinder analysis of the ball and wheel.

Videos were downloaded at 720p using a `yt-dlp` [64] script to balance quality and file size. Frames were cropped with FFMPEG [18] to isolate the wheel and remove extraneous graphics, then resized to 640×640 to fit YOLO's input constraints and ensure the ball, sometimes with just a few pixels visible, remained detectable.

3.2 YOLO Segmentation

YOLOv11 [27], a state-of-the-art object detection model released in October 2024, was used to extract precise positional data from video footage. It segments three key elements: the roulette ball, the green zero pocket, and the croupier's hand. The green zero pocket serves as a fixed reference for measuring the wheel's angular position, while the croupier's hand provides a visual cue for identifying spin boundaries (see Section 3.3). The roulette ball is the primary target for trajectory analysis. Instance segmentation enables accurate, pixel-level

outlines, helping resolve visual ambiguity in noisy footage. See Figure 3.3 for a comparison with bounding box detection.



(a) Bounding box (b) Segmentation

Figure 3.3: Comparison of bounding box detection and instance segmentation on a roulette wheel frame. The left image shows bounding boxes around the hand, ball, and green pocket, providing coarse localisation. The right image shows segmentation masks that more precisely outline each object, offering improved spatial accuracy for identifying pocket boundaries and object contours.

YOLOv11 was selected for its superior speed and precision, outperforming earlier variants such as YOLOv8 in key benchmarks [25]. The small variant was used, offering a strong balance between efficiency and accuracy, which was important for processing over 5 million frames extracted from the collected video footage. Simpler methods like HSV thresholding were initially explored but proved unreliable under varying lighting conditions and required extensive manual tuning.

Detecting a small, fast-moving ball is non-trivial, and as a supervised model, YOLO requires labelled data. We manually annotated 100 frames using Roboflow [45], marking the ball, green zero, and croupier's hand with polygon outlines. To address the rarity of the ball in the critical frames near drop-off and post-drop-off, these moments were oversampled to balance the dataset. Through an iterative cycle of annotation, training, correction, and retraining, we expanded the dataset to 500 verified frames with minimal manual effort. Synthetic augmentations such as randomised brightness and perspective shifts further tripled the dataset size, improving robustness under varied conditions. The impact of augmentation on model performance is evaluated in Chapter 5.

The model was trained and inferred with PyTorch [37] on an NVIDIA H100-class GPU. For efficiency, we exported the trained model to ONNX format [35], an open standard for cross-platform inference. We deployed it in with a Docker container, processing 20 video streams at 0.8ms/frame. Each frame's output included class labels, confidence scores, and polygon coordinates, with ball and green zero centroids calculated for trajectory construction. A Jupyter notebook selected the highest-confidence detections per frame, computed polygon centroids,

and organised the annotations into a structured table mapping frames to coordinates. Further deployment and training details are available in Appendix C.

Detection performance is evaluated using precision, recall, and mean average precision (mAP) on a held-out portion (120 frames) of the annotated training set. However, in this task, mean localisation error, defined as the average Euclidean distance (in pixels) between detected object centroids and manually annotated ground truth positions, provides a more meaningful evaluation metric. While mAP measures how well the model classifies and localises bounding boxes, it is less sensitive to small shifts in position. This makes it less suitable for evaluating small, fast-moving, and unstable objects like a roulette ball. Localisation error, by directly capturing spatial accuracy, is a better proxy for the actual goal of precisely tracking object trajectories over time. Full results are reported in Chapter 5.

3.3 Spin Splitting

With YOLO-based hand annotations, we generated a binary detection signal: 1 when a hand was present, 0 otherwise. This signal allowed us to segment the continuous video into individual roulette spins, using hand presence as a consistent marker. The hand typically appears between spins and disappears just before the ball is launched.

To reduce flickering and false positives, we smoothed the signal using a 100-frame (2-second) rolling average. Spin boundaries were defined using a threshold of 0.4: a spin began when the signal dropped below this value and ended when it rose above it.

Segments outside the 15 to 45 second range were discarded, eliminating incomplete or false detections. This filtering reduced the original set of 3,500 candidate segments to around 3,200 validated spins. Thresholds were tuned via an interactive tool, with accuracy confirmed by manual inspection and analysis (see Chapter 5).

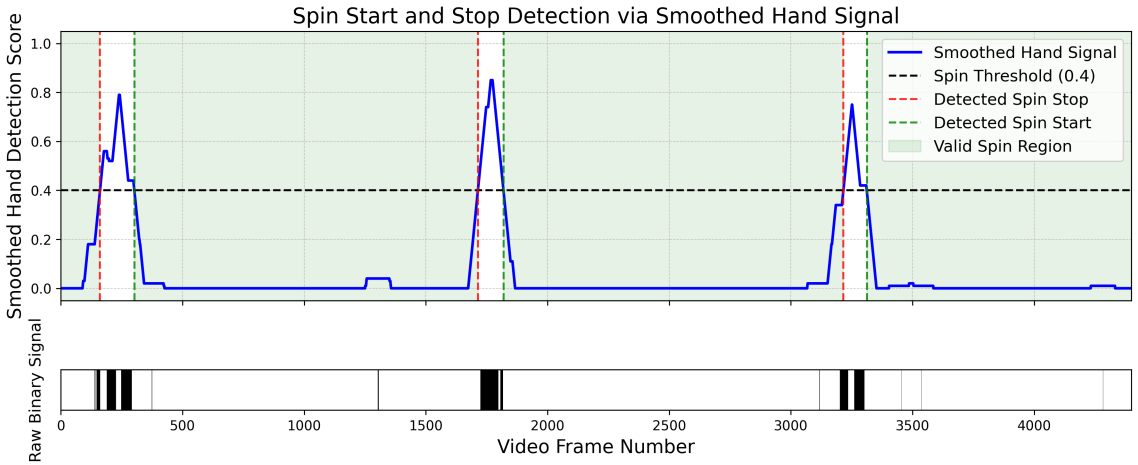


Figure 3.4: Smoothed hand-detection signal used for spin segmentation. The blue line shows the 100-frame rolling average of YOLO's binary hand detections (bottom panel). Transitions across the 0.4 threshold (black dashed line) define spin start (green) and stop (red) points. Green shaded regions indicate valid spins. The signal bar below indicates the raw binary detection signal, with black lines signifying a hand detection.

3.4 Geometric Correction via Homography and RANSAC

After isolating individual spins from the video, we converted the raw (x, y) pixel coordinates of the ball and green zero pocket (our reference for the inner wheel) into normalised polar coordinates. Given the circular nature of roulette motion, polar coordinates provide a more intuitive and invariant representation.

However, varying camera angles across videos cause the ball's circular motion to appear elliptical. Correcting this perspective distortion was essential for accurate radial and angular analysis. To address this, we grouped spins by their source video, each having a fixed camera perspective. For each video, we aggregated the first 20 frames (0.4 seconds) of every spin, capturing the ball's regular, rim-bound motion.

We applied Random Sample Consensus (RANSAC) [19] to fit an ellipse to the aggregated trajectory. RANSAC was chosen for its robustness to outliers, enabling it to handle occasional misdetections or noise. Parameters were tuned empirically and validated visually to ensure a reliable fit.

From the fitted ellipse, we computed a homography mapping it to a unit circle and applied this transformation to all (x, y) coordinates of the ball and green zero pocket. This correction preserved the rim-bound motion, the main region of interest, with high fidelity. Finally, the transformed coordinates were converted into polar form (θ, r), yielding a clean, normalised representation of each spin.

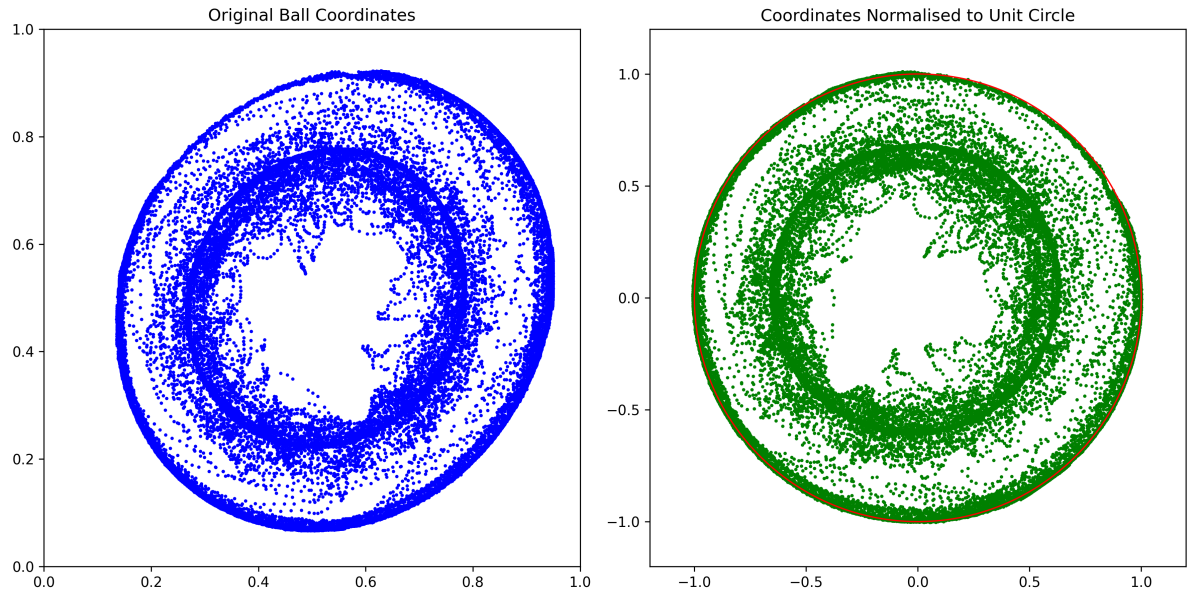


Figure 3.5: Visualisation of all detected ball positions across a single video before (left) and after (right) applying the homography correction. Each dot represents a ball coordinate from one frame, aggregated over hundreds of spins. The left plot reveals elliptical distortion caused by the off-centre camera angle. The right plot shows the same data remapped to a unit circle, standardising the wheel geometry. The denser paths reveal where the ball spends the most time, highlighting structural features of the wheel and typical motion patterns.

We tested alternative pixel-based undistortion methods, but they performed poorly due to inconsistent lighting, occlusions, and design variations in the wheel. The distortion was also

hard to approximate consistently across videos, and these methods were computationally expensive. In contrast, the RANSAC-based ellipse fitting was simpler, more reliable, and better aligned with the geometric nature of our task.

To assess correction quality, we measured the radial stability of the ball across the first 100 frames (2 seconds) of each spin. After transformation, we calculated the standard deviation of the radial distances from the centre. Low variance indicates consistent circular motion, as expected in an idealised spin. Full results are reported in Chapter 5.

3.5 Interpolation and Smoothing

After geometric correction, residual noise from missed detections, occlusions, and false positives was addressed using a backwards-looking moving average with a 12-frame window, applied to both radial and angular components. This provided causal smoothing without future-data leakage.

Missing values were filled via forward linear interpolation to maintain continuity across short occlusions. Outliers, often from flickers or spurious YOLO detections, were removed based on deviation from a rolling mean and standard deviation over the past 150 frames.

Processing steps were visually inspected and parameters tuned to preserve key motion features (see Figure 3.6). Rather than flattening the data, this method selectively filled gaps and suppressed noise while retaining dynamics such as angular jitter, nonlinear deceleration, and radial variation, crucial for downstream modelling.

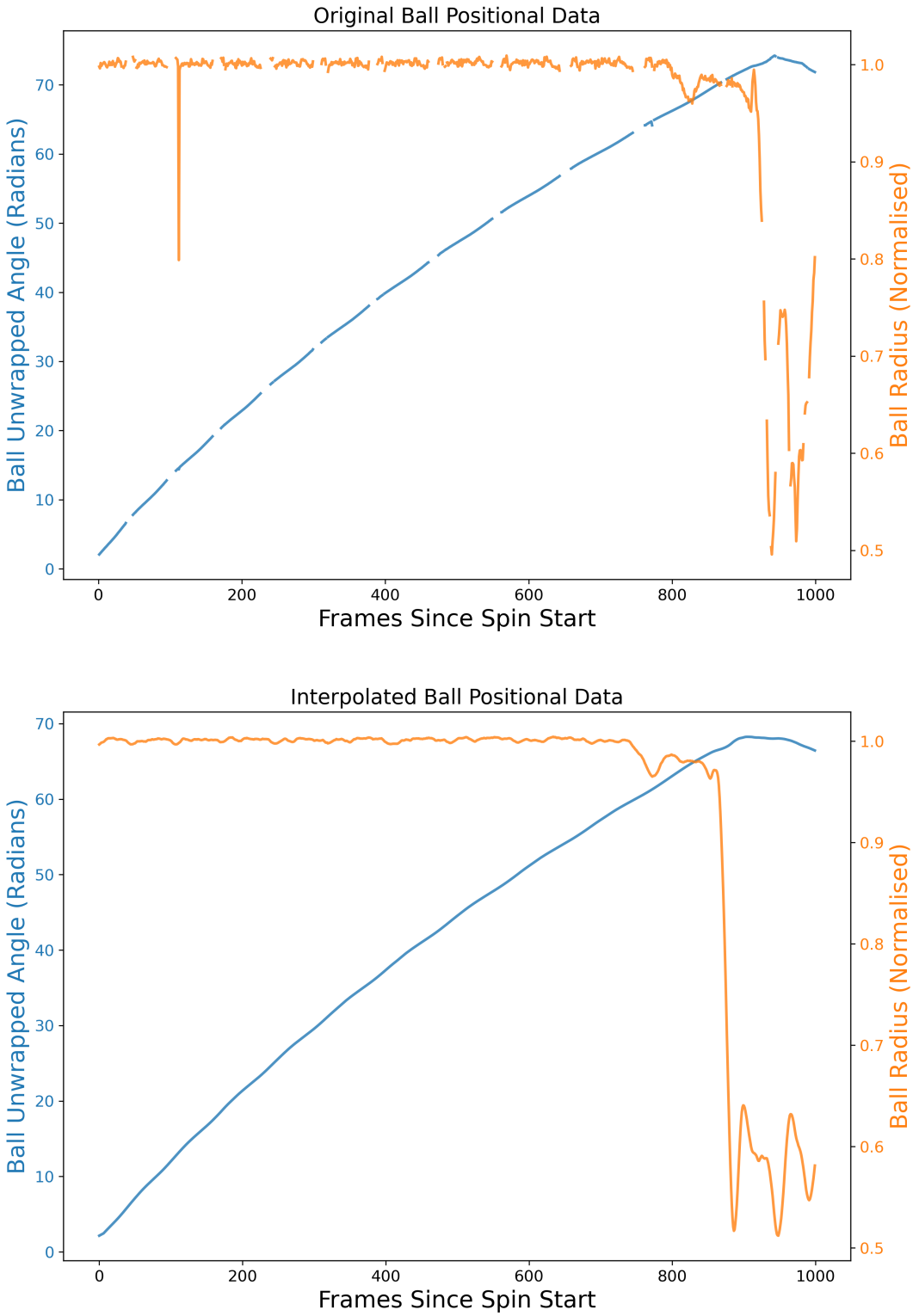


Figure 3.6: Ball trajectory plotted over time after conversion to polar coordinates. The blue line shows the unwrapped angular position (in radians), and the orange line shows the normalised radial distance from the wheel centre, both as functions of time since spin start. The top has raw data showing missed detections, noise, and temporary occlusions. The bottom shows cleaned data after causal smoothing, forward interpolation, and outlier removal.

3.6 Drop-Off Event Detection

Accurately identifying the drop-off event is critical for our predictive objectives. Physically, this occurs when the ball either collides with a deflector or bypasses them and rolls into the inner wheel. Prediction is difficult due to the dual nature of this event and the inconsistent positioning and alternating orientations of deflectors. Although homographic correction provides a top-down view, it does not standardise wheel orientation, so deflector positions vary across spins and videos.

Previous work has used fixed radial thresholds to detect drop-off [47], but this fails to account for variation in deflector layout and cannot reliably distinguish between the two types of events. Instead, we adopted a more robust method based on analysing local trends in radial motion. A 20-frame sliding window was applied to each spin's refined trajectory, and a linear regression was fitted within each window. A drop-off event was marked when the slope fell below -0.006, indicating a sustained inward descent.

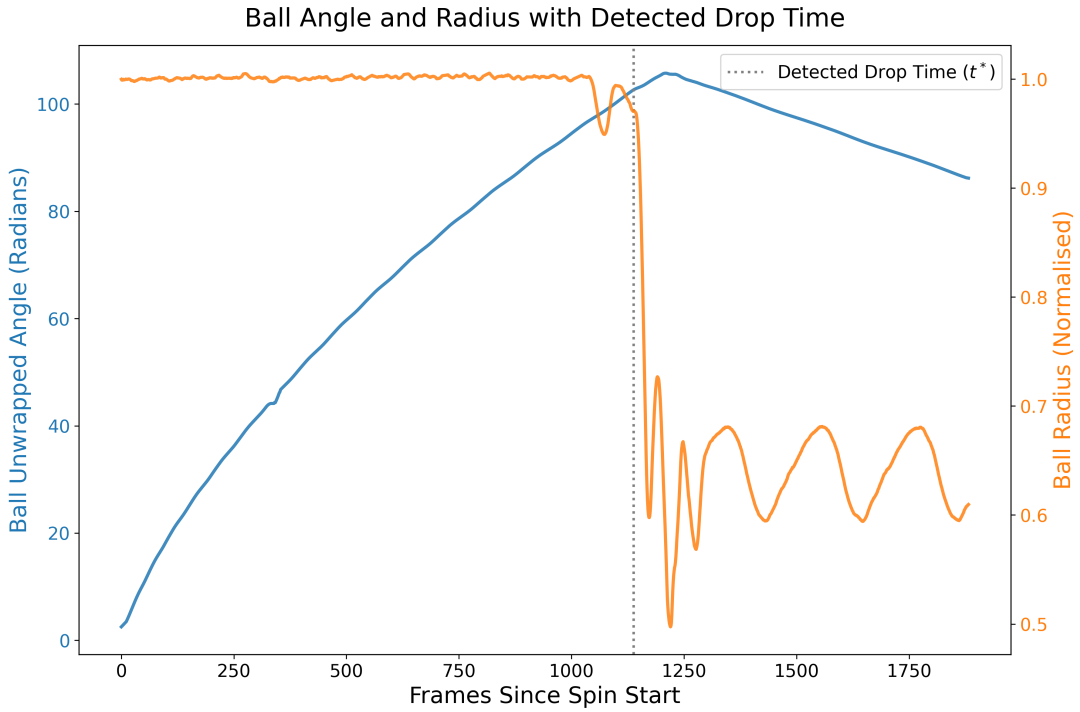


Figure 3.7: Refined trajectory data showing unwrapped angle (radians) and normalised radius as a function of frame index. The detected drop-off point t^* is marked with a vertical line, indicating the transition from stable rim-bound motion to chaotic inner-wheel dynamics.

This threshold was tuned empirically to balance sensitivity to gradual and sudden drops, and was validated through visual inspection. Extensive testing confirmed the method's accuracy across varied spins, reliably separating true drop-off events from early-phase noise. Each event is recorded as a frame number relative to the start of the spin, with frame 0 marking the first observed frame. This drop-off frame is denoted as t^* and always occurs at least 500 frames (or 10 seconds) into the spin, placing it beyond the input window given to the model. Predicting t^* based on early motion is a key goal of our machine learning approach.

3.7 Feature Extraction for Prediction

With all retained spins cleaned, geometrically corrected, and made fully continuous, we extracted structured features to represent their dynamics over time. These features form the direct input to the machine learning models. Each spin was converted into a multivariate time series, with frame-wise features capturing both the motion of the ball and the rotation of the inner wheel, represented by the green zero pocket.

Ball features:

- Angular position, encoded as $\sin(\theta)$ and $\cos(\theta)$ to ensure continuity
- Angular velocity: first-order difference of θ
- Angular acceleration: second-order difference of θ
- Radial position r
- Radial velocity: first-order difference of r
- Radial acceleration: second-order difference of r

Wheel (green zero) features:

- Angular position: $\sin(\theta)$ and $\cos(\theta)$
- Angular velocity: first-order difference
- Angular acceleration: second-order difference

To ensure numerical stability and avoid discontinuities at angular boundaries, angular values were represented using their sine and cosine components. Each spin was encoded as a matrix of shape $(L \times N)$, where L is the number of frames (from launch to end), and $N = 11$ is the total number of features (7 for the ball, 4 for the wheel). Each frame corresponds to 0.02 seconds of real time, with 50 frames per second. Since spin durations vary, L is not fixed across samples. Each spin also has the associated drop-off time t^*, expressed as a frame index relative to the spin's start.

While the selected features are physically meaningful, reflecting quantities such as velocity and acceleration, they are derived entirely from pixel-level data without any enforced physical constraints or embedded equations. This contrasts with physics-informed neural networks (PINNs), discussed in Section 2.4, which explicitly incorporate known dynamics into the training process. In our approach, learning is kept independent of physics assumptions, allowing the model to infer structure directly from the data without imposed biases.

3.8 Filtering Invalid Spins

To ensure data quality before model training, we applied a final filtering step to remove structurally invalid or unreliable spins. The aim was not to discard rare or physically interesting cases, but to exclude data that was clearly broken, implausible, or beyond recovery. Filtering was based on basic physical and continuity checks.

Spins were excluded under the following conditions:

- Implausible radial values (e.g. far outside wheel bounds)
- Missing drop-off detection
- Implausible dynamics, such as extreme angular discontinuities

Flagged spins were reviewed through visualisations to confirm that the issues were irrecoverable and not fixable. Rather than discarding them completely, we retained these spins with metadata marking their invalid status for later ablation analysis.

After filtering, the final modelling dataset comprises 2,765 high-quality spins from 42 videos, each paired with a ground-truth drop-off time. All trajectories are continuous following interpolation and geometric correction, and are structured into consistent, meaningful feature sets suitable for machine learning.

This processed dataset enables testing of Hypothesis H1 in Chapter 5. The next chapter describes the model architectures and training procedures used to take advantage of this data.

Chapter 4

Predictive Modelling Experiment Design

4.1 Overview

This chapter details the experimental methodology used to predict key roulette drop-off dynamics from structured video-derived trajectory data. We define three modelling tasks: (1) predicting the scalar time to drop-off, (2) forecasting the ball's angular trajectory up to drop-off, and (3) forecasting the wheel's angular trajectory over the same interval.

Each task is handled by a dedicated LSTM-based architecture. This modular approach avoids error entanglement across tasks, enables task-specific optimisation, and allows independent evaluation of model capabilities for scalar versus sequence prediction. A multi-task learning approach could have been employed, but was deemed suboptimal due to the different nature of the tasks, which would likely lead to conflicting gradient updates and degraded performance in the different domains [65].

Input to each model consists only of the first T frames of a spin's trajectory, with $T \in \{100, 200, 300, 400, 500\}$ (2 to 10 seconds) to simulate different levels of early context and real-time constraints. This controlled variation allows us to assess how prediction accuracy scales with available information, addressing Hypothesis H3.

Model hyperparameters and ablation studies were tuned once on the $T = 300$ horizon, providing a middle ground between short and long input sequences. The optimised architecture was then retrained independently for each T to investigate how predictive accuracy evolves with input duration.

This design ensures clear interpretability, robustness to overfitting, and extensibility to other video sources or physical systems. The next section defines the problem setup in detail.

4.2 Problem Setup

Each roulette spin is represented as a multivariate time series of frame-wise features derived from video (as described in Section 3.7). These include angular position, velocity, and acceleration for both the ball and wheel, encoded in a geometry-normalised frame.

The predictive models operate strictly on the first T frames of each spin and are trained to forecast one of the following:

- Drop-off Time: A scalar value representing the number of frames from the first input time step to the drop-off frame t^*.
- Ball Angular Trajectory: A sequence of angular positions $[\sin(\theta), \cos(\theta)]$ from frame $T + 1$ to t*.
- Wheel Angular Trajectory: A sequence of angular positions for the green zero pocket over the same interval.

The length of the output sequence is variable across spins, determined by $t^* - T$. Models are trained with dynamic loss masking [54] to handle this variability.

For the angular trajectory prediction tasks, the true drop-off frame t^* is provided as known during training and evaluation. This decision avoids compounding errors from the drop-off time prediction, enables isolated evaluation of trajectory modelling performance, and allows clearer diagnosis of system bottlenecks.

For each task, performance is evaluated on a held-out test portion of the dataset using task-specific metrics (defined in Section 4.5). Models are compared against simple baselines to quantify gains from learned dynamics, with a lack of directly comparable approaches available a limitation. This setup however still reflects a practical real-time scenario: given partial information, can machine learning models make physically meaningful predictions before chaos dominates?

4.3 Model Architecture

Each prediction task is handled by a dedicated LSTM-based model, tailored to its output type: scalar (drop-off time) or sequences (ball and wheel angular trajectories). This modular design enables task-specific optimisation without cross-task interference and simplifies both training and evaluation.

Further architectural details, including model diagrams, hyperparameter search ranges, and training curves, are presented in Appendix D.

4.3.1 Drop-Off Time Model: Sequence-to-Scalar

- Input: Sequence of shape (T, N), where N is the number of input features.
- LSTM Layer: Two stacked LSTMs (128 units, then 96), the first with return_sequences=True.
- Layer Normalisation: Applied after each LSTM layer.
- Output: Dense layer with 1 unit, linear activation.
- Dropout: Not used in the final version, though tested during tuning.

This model predicts the number of frames between the final input timestep and the drop-off event. The two-layer LSTM stack enables richer temporal encoding than a single LSTM, capturing complex motion patterns over time. Layer normalisation is used to stabilise training and mitigate internal covariate shift. A stacked LSTM configuration is expressive and well-suited to this sequence summarisation task.

4.3.2 Ball Trajectory Model: Sequence-to-Sequence

- Input: Sequence of shape (T, N).
- Encoder: LSTM with 128 units and `return_sequences=False`.

- RepeatVector: Replicates the encoder output $(t^* - T)$ times.
- Decoder: LSTM with 128 units and `return_sequences=True`.
- Output: TimeDistributed Dense layer with 2 units (sin, cos), linear activation.
- Dropout: Not included, though high rates (e.g., 0.5) are explored.

This model predicts the ball's angular position at each timestep from the end of the input sequence until the drop-off frame t^*. An encoder-decoder architecture [54] is used to allow flexible output lengths and temporal consistency across the prediction window. Representing angles as $(\sin, \cos)$ preserves continuity across 2π boundaries. Dropout, bidirectional encoders, and deeper decoders are considered, but a single-layer setup offers a balance between performance and training stability.

4.3.3 Wheel Trajectory Model: Sequence-to-Sequence

- Input: Sequence of shape (T, N).
- Encoder: LSTM with 128 units and `return_sequences=False`.
- RepeatVector: Replicates the encoder output $(t^* - T)$ times.
- Decoder: LSTM with 128 units and `return_sequences=True`.
- Output: TimeDistributed Dense layer with 2 units (sin, cos), linear activation.
- Dropout: Not used in the final design, though smaller rates (e.g., 0.2) are evaluated.

The wheel trajectory task involves more regular, low-variance motion and is expected to require a simpler model. However, a higher-capacity architecture identical to the ball trajectory model is found to perform best. This is likely due to the structure of the input features, which encode interactions that benefit from deeper temporal modelling. Using the same architecture across both models also simplifies training and implementation. Angular outputs are represented in $(\sin, \cos)$ form to avoid discontinuities.

4.4 Training Procedure

All models are developed with TensorFlow [1] and Keras [12]. The dataset is split into training (75%), validation (12.5%), and test (12.5%) sets. The validation set is used for hyperparameter tuning and early stopping, while the test set is reserved for final evaluation.

4.4.1 Target Construction

Drop-off Time Model: The target is the absolute frame index t^* corresponding to the ball's drop-off event, taken directly from the labelled data and derived in Section 3.6.

Trajectory Models: The targets are sequences of $[\sin(\theta), \cos(\theta)]$ pairs for either the ball or the wheel, spanning frames $T+1$ to t^*. Because spins vary in length, each sequence is padded to a fixed maximum length. During training, a binary mask ensures that loss is computed only over valid (non-padded) frames.

4.4.2 Loss Functions

Drop-off Time Model: Mean Squared Error (MSE) loss is used to penalise deviations between the predicted and true drop-off frame indices, computed as:

$$\mathcal{L}_{\text{drop}} = \frac{1}{N}\sum_{i=1}^{N}(t_i^* - \widehat{t}_i)^2 \tag{4.1}$$

where t_i^* is the true drop-off frame, and $\widehat{t}_i$ is the predicted value, averaged over N samples.

Trajectory Models: A masked angular loss based on average circular distance is used. For each timestep t in a valid region of a spin:

$$\ell_t = 1 - \Big(\sin(\widehat{\theta}_t)\sin(\theta_t) + \cos(\widehat{\theta}_t)\cos(\theta_t)\Big) \tag{4.2}$$

The final masked loss over valid frames is:

$$\mathcal{L}_{\text{trajectory}} = \frac{1}{\sum m_t}\sum_t m_t \cdot \ell_t \tag{4.3}$$

where $m_t \in \{0, 1\}$ is a binary mask indicating valid (unpadded) timesteps.

This approach correctly measures angular error across the sequence while handling variable-length data via masking.

4.4.3 Optimisation Strategy

All models are trained using the Adam optimiser [28] with the following settings:

- Initial learning rate: $1e-3$ for the drop-off model, $5e-4$ for trajectory models
- Learning rate scheduler: ReduceLROnPlateau with a patience of 5 epochs
- Batch size: 64
- Weight decay (L2 regularisation): $1e-5$
- Early stopping: after 10 epochs without validation improvement (restore best weights)

These choices were tuned and balanced stability with training speed while regularising against overfitting.

4.4.4 Hyperparameter Tuning

Hyperparameter tuning was performed once on the $T = 300$ input horizon. $T = 300$ was selected because it balances early and late spin dynamics, providing a representative challenge. Tuning each of the 3 models at each of the 5 time horizons (T) would have been prohibitively expensive, and 300 frames offer a strong middle ground without favouring short or long extremes.

Each tuning process used a random search strategy across 25 configurations. Each trial was capped to a maximum of 150 early-stopping epochs. Architectures explored included varying LSTM hidden sizes (32-256), number of layers (1-3), and dropout rates (0.1-0.5).

The best configuration (described in Section 4.3) selected based on validation loss, was frozen and reused across all T settings without further adjustment. While with unconstrained compute a more granular approach might be employed, a brief investigation showed that similar hyperparameters consistently performed best across different time frames, leaving the cost of re-tuning for each T unjustified.

4.4.5 Ablation Experiments

Ablation studies were conducted to assess model sensitivity to different input features and preprocessing choices. Experiments were performed on the $T = 300$ input horizon on the ball trajectory model using the same fixed architecture and hyperparameters:

- **Feature Removal**: Retraining models with individual features omitted (e.g., angular acceleration removed) to evaluate their contribution to performance.
- **Noise Robustness**: Injecting Gaussian noise into angular and radial features during training and testing to simulate imperfect detections.
- **Inclusion of Invalid Spins**: Training and evaluating models with and without spins marked as invalid, to assess the impact of corrupted or unreliable data on prediction accuracy and robustness.
- **Reduced Dataset Size**: To evaluate the data efficiency of the models, we evaluate training performance on 50% and 25% of the full dataset.

4.4.6 Input Horizon Experiments

To test Hypothesis H3, models were retrained for each T in $\{100, 200, 300, 400, 500\}$ using the best performing architecture and hyperparameters. This isolates the effect of input duration on predictive performance. Each model was trained with early stopping, for a maximum of 250 epochs.

4.5 Evaluation Metrics

Each model is evaluated using task-specific metrics designed to measure predictive accuracy and compare learned performance against naïve baselines.

4.5.1 Drop-Off Time Model Metrics

- **Mean Absolute Error (MAE)**: Measures the average difference between predicted and actual drop-off times. Reported in frames and seconds (MAE / 50 FPS).
- **Root Mean Squared Error (RMSE)**: Penalises larger errors more heavily, highlighting instability or outlier mispredictions.

MAE offers an interpretable measure of average error, while RMSE captures the model's sensitivity to large deviations.

Baselines:

- **Mean Predictor**: Always predicts the average drop-off time observed in the training set.
- **Random Predictor**: Samples a drop-off time uniformly between the minimum and maximum training set values for each spin.

These baselines are intentionally naïve. The mean predictor captures central tendency and tests whether the model learns more than simple dataset statistics. The random predictor defines a performance floor, approximating how poorly an untrained or uninformed system might perform. While simplistic, these baselines serve as robust anchors for evaluating the predictive value of our models.

4.5.2 Angular Trajectory Models Metrics:

- **Trajectory Mean Absolute Error (radians)**: Measures the model's overall ability to predict the ball's angular trajectory. We compute the absolute error at each timestep within the valid masked region for every spin in the validation set, average over timesteps, and then average across all spins. Range is $[0, \pi]$.
- **Drop-Off Angular Error (radians)**: Angular error at the final valid timestep of each spin (i.e. the drop-off point), measuring long-horizon prediction accuracy where precision is most critical. Range is $[0, \pi]$.

Predictions are converted from $[\sin(\theta), \cos(\theta)]$ to angles via `atan2` during evaluation.

Baselines:

- **Last Angle Predictor:** Repeats the final observed angle at frame T across all future frames.
- **Random Angle Predictor:** Generates a random angle for each future frame, sampled uniformly from the range $[0, 2\pi]$.

Only valid frames ($T + 1$ to t^*) are evaluated, using the same masking scheme described earlier to ignore padded timesteps. Evaluation captures both instantaneous prediction error and cumulative deviation as the ball approaches the drop-off.

These baselines serve as essential reference points. The last-angle predictor tests a naïve persistence strategy, while the random predictor provides a lower bound on performance, highlighting how much signal the model is able to extract beyond chance, and approximating the floor of what an untrained human might guess.

While a physics-based model like that of Small and Tse [47] could, in theory, offer a more grounded comparison, its complexity, lack of a reproducible implementation, and differing end-goals made it impractical within our setup. The chosen baselines strike a balance between simplicity and interpretability, offering clear context for model performance.

4.6 Chapter Summary

This chapter detailed the experimental design for predicting roulette drop-off dynamics using structured video data. Three tasks, scalar regression for time-to-drop-off, and sequence prediction for ball and wheel trajectories, were each handled by modular LSTM-based architectures. Systematic input horizon variation, rigorous training procedures, dynamic output masking, and simple baselines were used to ensure interpretable comparisons and robust conclusions. This framework enables direct testing of Hypotheses H2 and H3 in chapter 5 through empirical results.

Chapter 5

Results

5.1 Computer Vision Pipeline Evaluation

Hypothesis H1 concerns the system's ability to extract accurate ball and wheel trajectories from raw video. This section evaluates the dataset creation pipeline described in Chapter 3, assessing object detection, centroid localisation, geometric correction, and spin segmentation through quantitative and qualitative results.

5.1.1 Quantitative Vision Results

Detection Metrics The vision system uses a YOLOv11-small model to detect the ball, green zero pocket, and croupier's hand. Detection was evaluated on 120 held out annotated test frames using precision, recall, and mean average precision (mAP) at different IoU thresholds on the predicted polygons.

Table 5.1: Detection metrics for YOLOv11-small model on 120 held-out annotated test frames

Class	Instances	Mask Precision	Mask Recall	Mask mAP@50	Mask mAP@50–95
Ball	100	0.990	0.990	0.994	0.523
Green	120	1.000	0.917	0.958	0.744
Hand	23	1.000	1.000	0.995	0.910
All	243	0.997	0.969	0.983	0.726

The YOLO segmentation model achieved strong performance, with a mask mAP50 of 0.983 across all classes. While the ball mask mAP50-95 was lower at 0.523, this is not a major concern for this application. The drop in mAP50-95 is primarily due to the small size of the ball, where minor pixel-level errors disproportionately impact the metric. Because of this sensitivity, object localisation performance is more reliably evaluated by comparing the predicted centroid positions to the actual centroids, rather than relying solely on pixel-level IoU-based metrics.

Localisation evaluation based on centroid predictions showed very strong results. Across the test set, the mean distance between predicted and true centroids was 1.36 pixels, with a maximum error of 6.67 pixels and a minimum of 0.09 pixels. The standard deviation was 1.05 pixels. Given the frame size of 640×640, these errors are negligible for practical purposes, confirming the model's suitability for accurate ball tracking.

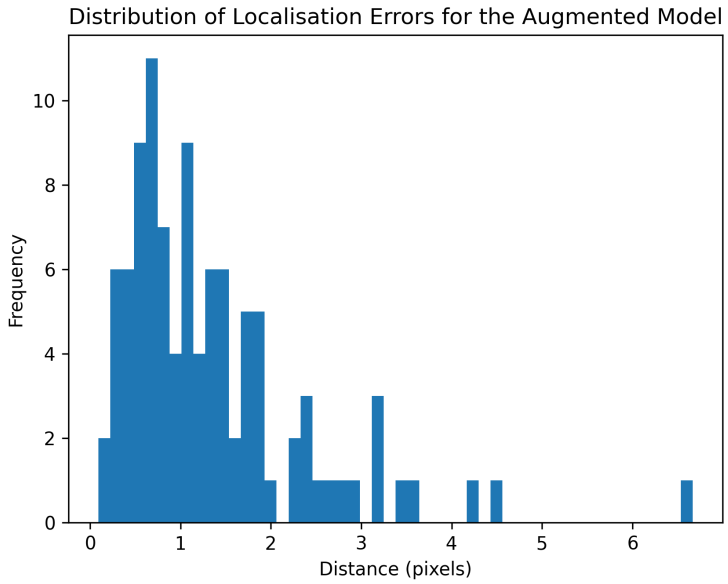


Figure 5.1: Histogram of localisation errors (in pixels) for the augmented model. Decimal values reflect sub-pixel precision in centroid computation from segmentation masks, indicating accurate and fine-grained localisation.

Training without data augmentations led to noticeably worse performance. The mask mAP50 dropped from 0.983 to 0.933, and localisation accuracy also degraded, with the mean centroid error increasing from 1.36 pixels to 1.56 pixels, and the maximum error almost doubling to 12.32 pixels. The Ball class was particularly affected, showing a significant drop in mask mAP50-95 from 0.523 to 0.453. These results highlight the critical role augmentations played in improving generalisation and precision, especially for small, fast-moving objects.

Further training details are provided in Appendix C.

Geometric Correction Quality The homography correction method (Section 3.4) was evaluated by measuring how consistently it mapped ball trajectories onto a unit circle, transforming elliptical motion caused by camera perspective into an idealised circular path. For each spin, the standard deviation of the radial distance from the corrected centre was computed over the first 100 frames. Since the ball is initially expected to follow a near-perfect circular trajectory around the outer rim, deviations from the unit radius directly reflect geometric distortion or noise in tracking. A low radial standard deviation therefore indicates successful geometric normalisation by the homography.

Across the dataset, the mean radial standard deviation was 0.0011, with a standard deviation of 0.0021 across spins, indicating extremely tight adherence to the unit circle as shown in Figure 5.2. Only 6 out of 3349 spins ($<$0.18%) exceeded a threshold of 0.005. These outliers were typically caused by tracking errors and were filtered out during post-processing.

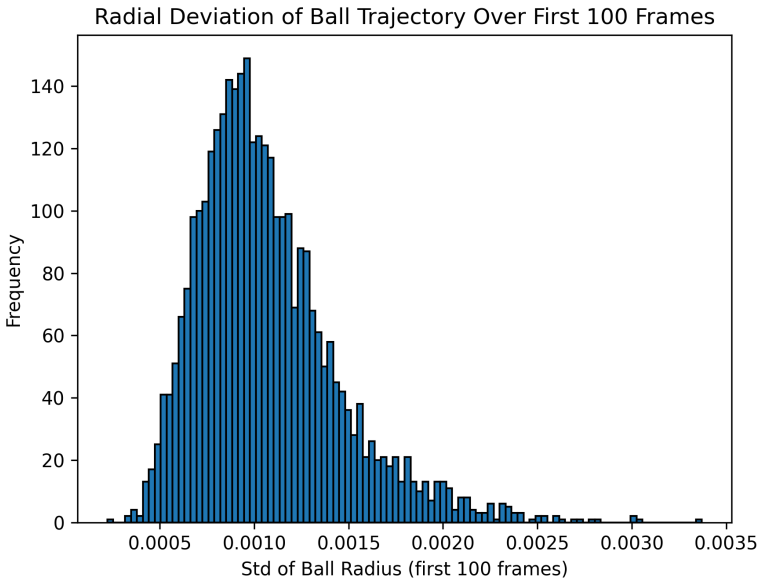


Figure 5.2: Radial deviation of ball trajectory over the first 100 frames, measured after homography correction. The standard deviation is computed relative to a unit circle, where lower values indicate more consistent circular motion. Most spins show minimal deviation, with a small number of outliers (filtered out) exceeding 0.005 due to tracking noise or errors.

Manual inspection further confirmed that the homography preserved real-world geometry without introducing visual artefacts, validating the effectiveness of the correction (Figure 3.5).

Spin Length Analysis Spin splitting reliability was assessed by analysing the distribution of drop times (Figure 5.3), which ranged from 500 to 1468 frames, with a mean of 819.1 and a standard deviation of 182.0. The median was approximately 800 frames. A small number of entries at 0 frames appear in the histogram; these correspond to spins where no drop time was detected and were later marked as invalid. The overall distribution suggests a consistent temporal pattern in spin progression.

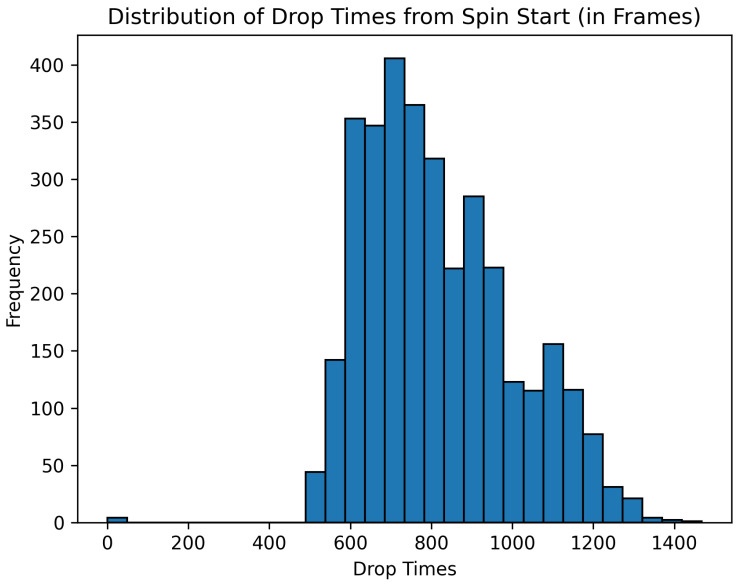


Figure 5.3: Histogram of drop times (in frames) measured from the start of each spin.

5.1.2 Qualitative Vision Analysis

Example frames were reviewed to assess the vision system's performance under real-world conditions.

Object Detection: The ball and green zero pocket were reliably detected across most frames, even under partial occlusion or varying lighting. Occasional detection failures occurred due to heavy blur, hand occlusions, or harsh lighting, but these disruptions were brief and did not lead to persistent detection failures across spins.

Spin Splitting: Hand detection signals enabled clean and consistent spin splitting throughout the footage despite sometimes noisy detection data. Manual review of approximately 50 spins confirmed that split boundaries closely matched true spin start and end points, with no systematic timing errors or visible drift.

Trajectory Interpolation and Smoothing: Visual review of over 50 spins showed that both trajectory interpolation and smoothing significantly improved motion quality. Interpolation effectively recovered short gaps caused by missed detections, while smoothing reduced frame-to-frame jitter without introducing drift, distortion, or loss of true physical structure.

Overall Assessment: Minor tracking imperfections were observed but did not compromise dataset quality. Qualitative and quantitative results together confirm the robustness of the vision pipeline and validate Hypothesis H1.

(a) Standard detection under normal conditions.

(b) Successful detection despite heavy occlusion and distortion.

(c) Example of detection failure due to extreme conditions.

Figure 5.4: Example YOLOv11 segmentation results under different visual conditions. Each frame shows segmentation masks outlining detected objects: the ball, green zero pocket, and hand (if present). The model performs reliably in typical and challenging settings (a, b), with occasional failures under extreme occlusion or distortion (c).

5.1.3 Vision Pipeline Summary

Table 5.2: Summary statistics of the constructed roulette spin dataset.

Statistic	**Value**
Total Spins Extracted	3355
Invalid Spins	590 (17.59%)
Valid Spins	2765
Storage Size	503 MB
Total Unique Videos	42
Total Frames Processed	5,463,775
Total Video Time	30h 21m 15s
Drop Time Range (frames)	500–1468
Mean Drop Time (frames)	819.10

Quantitative and qualitative results confirm that the vision system reliably produces structured trajectory data with the spatial accuracy needed for predictive modelling, validating Hypothesis H1. Object detection, geometric correction, spin splitting, and smoothing all performed to a sufficient standard. Minor localisation errors introduce some downstream noise but do not materially affect system reliability.

5.2 Predictive Modelling Evaluation

This section evaluates model performance on drop-off time prediction, ball trajectory forecasting, and wheel trajectory forecasting. Results are reported across input horizons $T \in \{100, 200, 300, 400, 500\}$ (2 to 10 seconds) to test Hypotheses H2 (models outperform baselines) and H3 (accuracy improves with longer inputs).

Each task is assessed quantitatively against naïve baselines, followed by qualitative analysis of typical prediction behaviour and failure modes.

5.2.1 Drop-Off Time Prediction

The model predicts the absolute frame index at which the ball drops off the rim. As shown in Table 5.3, both Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) decrease consistently with longer input horizons, confirming that earlier observations improve accuracy. Performance improves from an MAE of 59.2 frames (1.18 seconds) at $T = 100$ to 25.9 frames (0.51 seconds) at $T = 500$.

The model outperforms both a mean predictor and a random baseline by a large margin, as seen in Table 5.4, particularly at $T = 500$ where it achieves over $6\times$ lower MAE than the mean predictor. These results support Hypotheses H2 and H3.

Table 5.3: Drop-off time prediction accuracy across input horizons, reported as mean $\pm$ 95% confidence interval (CI) (in frames).

Input Horizon (T)	**MAE**	**RMSE**
100	59.19 $\pm$ 5.80	80.80 $\pm$ 6.51
200	44.88 $\pm$ 5.19	66.60 $\pm$ 5.36
300	37.05 $\pm$ 3.72	51.16 $\pm$ 4.12
400	29.82 $\pm$ 3.12	42.00 $\pm$ 3.38
500	25.94 $\pm$ 2.46	34.88 $\pm$ 2.81

Table 5.4: Baseline comparison for drop-off time prediction at $T = 300$, reported as mean $\pm$ 95% CI (in frames).

Method	**MAE**	**RMSE**
Random Predictor	209.46 $\pm$ 15.46	255.64 $\pm$ 20.58
Mean Predictor	130.40 $\pm$ 9.49	158.39 $\pm$ 12.75
LSTM Model (Ours)	37.05 $\pm$ 3.72	51.16 $\pm$ 4.12

Qualitative Analysis Predicting drop-off time was consistently difficult due to the unpredictability introduced by the wheel's deflectors. These physical elements caused abrupt deviations in the ball's path that were not visible in the input data and could not be inferred from motion alone. Two otherwise identical trajectories, with the same speed, angle, and radius, could result in very different drop-off times purely depending on whether and when a deflector was struck. Despite this, the model performed reasonably well, estimating event times over 15 seconds in the future within 0.5 seconds in many cases.

There was no clear bias in errors across spins of different durations or speeds, suggesting the model generalised well across a range of trajectories. Predicted versus true drop-off times (Figure 5.6) show tight clustering around the diagonal, further indicating that the model captured core dynamics effectively, although it remained limited by the unpredictability of deflector interactions.

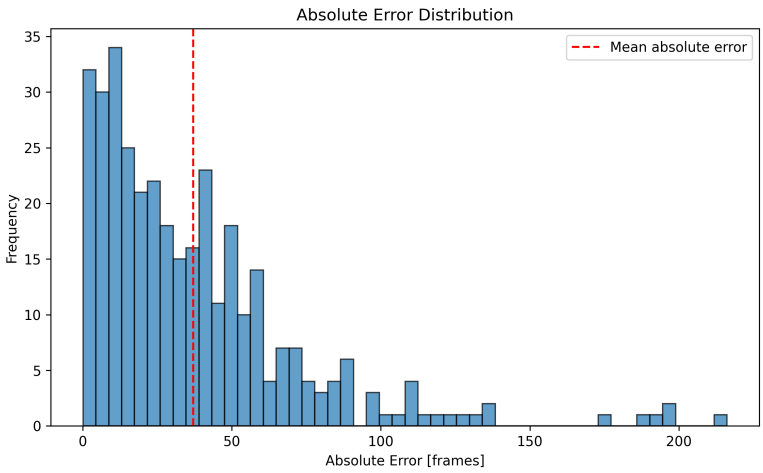


Figure 5.5: Distribution of absolute drop-off prediction time errors at $T = 300$.

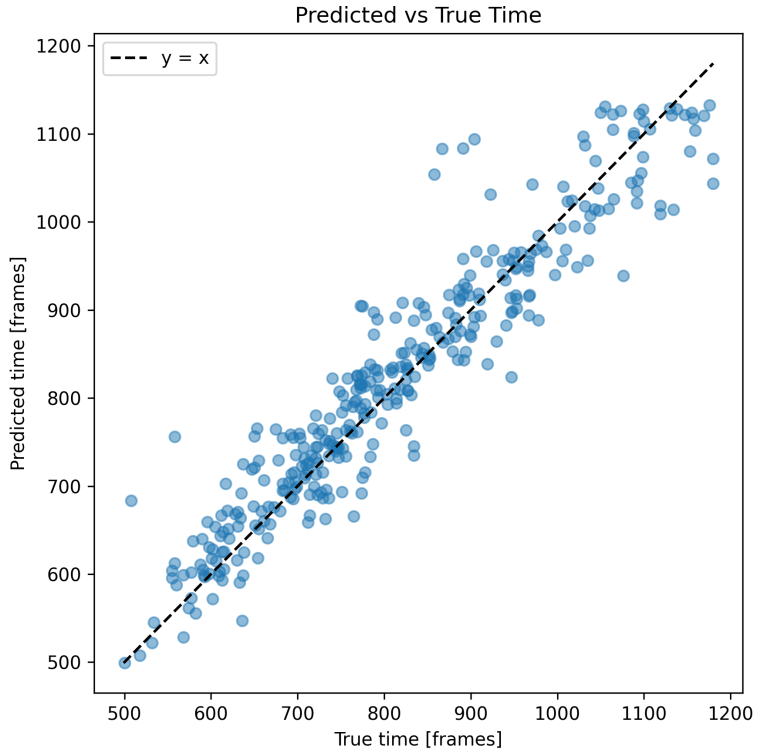


Figure 5.6: Scatter plot of predicted vs true drop-off times for the test set at $T = 300$. Each point represents a single roulette spin. The dashed diagonal line denotes perfect prediction. The clustering around the diagonal indicates that the LSTM model captures the underlying dynamics with reasonable accuracy, and no observed bias over the range of drop-off times.

5.2.2 Ball Angular Trajectory Prediction

Prediction accuracy improved with longer input horizons, particularly between $T = 100$ and $T = 300$, before gains levelled off. At $T = 500$, the model consistently produced accurate trajectory and drop-off estimates, with a trajectory MAE of 0.117 radians (6.7 degrees).

These results demonstrate the model's ability to extract predictive structure from noisy input, outperforming baselines and supporting Hypotheses H2 and H3.

Table 5.5: Ball angular trajectory prediction performance across input horizons. ± 95% CI (in radians).

Input Horizon (T)	Trajectory MAE	Drop-off Angular Error
100	0.845 ± 0.049	1.170 ± 0.0.091
200	0.349 ± 0.037	0.742 ± 0.074
300	0.185 ± 0.022	0.330 ± 0.036
400	0.124 ± 0.011	0.229 ± 0.022
500	0.117 ± 0.011	0.215 ± 0.027

Table 5.6: Baseline comparison for ball trajectory prediction at $T = 300$. Reported as mean ± 95% CI (in radians).

Method	Trajectory MAE	Drop-off Angular Error
Random Angle Predictor	1.572 ± 0.004	1.663 ± 0.099
Last-angle Predictor	1.565 ± 0.007	1.540 ± 0.097
LSTM Model (Ours)	0.185 ± 0.022	0.330 ± 0.036

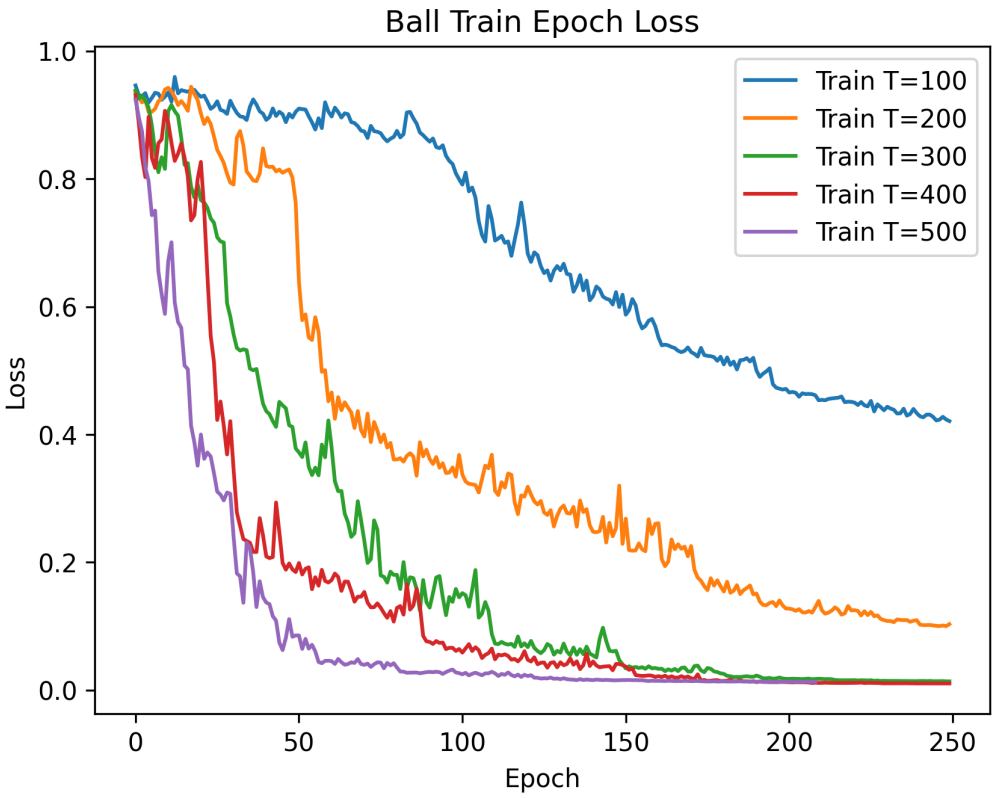


Figure 5.7: Training loss curves for ball trajectory prediction across all input horizons. Models with longer horizons converge quicker and with lower final losses, consistent with improved final performance.

Qualitative Analysis Compared to the baselines, our model outperformed both the random and last angle predictors by over 8 times, clearly validating its robustness. This confirms its ability to reliably track and forecast the ball's angular position, even in noisy conditions with minimal supervision. Both hypotheses H2 and H3 are strongly supported by the empirical results.

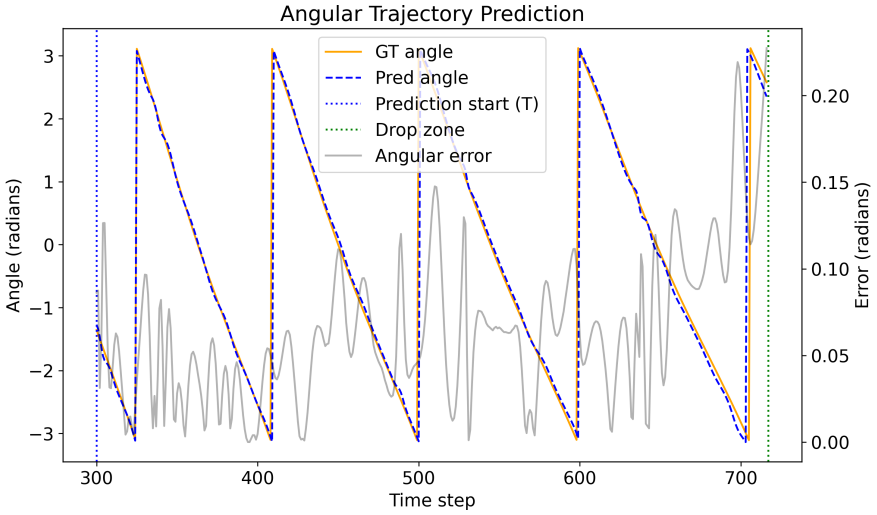


Figure 5.8: Example angular trajectory prediction for a single spin (at $T = 300$). The true trajectory from frame $T + 1$ to drop-off (t^*) is shown in solid orange, while the model's prediction is shown in dashed blue. The model successfully captures the angular dynamics and direction of motion despite increasing uncertainty near drop-off.

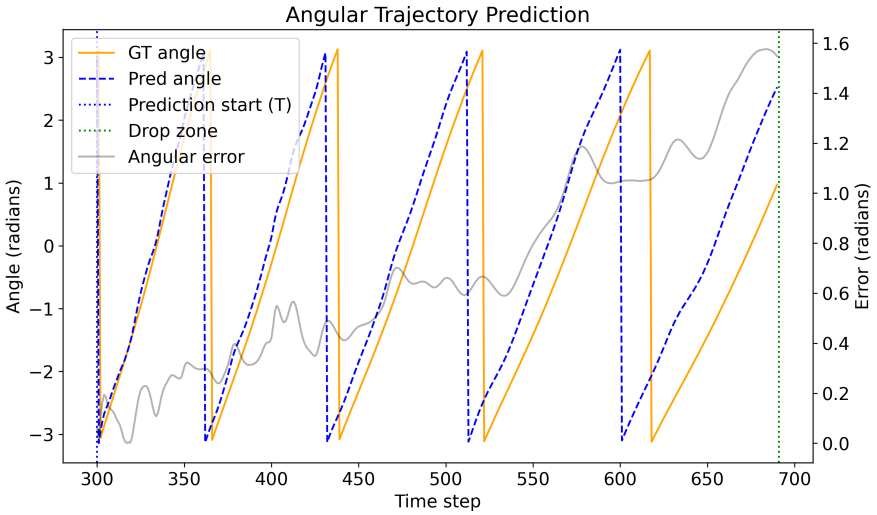


Figure 5.9: Example of a failure case in ball trajectory prediction at $T = 300$. The predicted trajectory (blue) diverges significantly from the ground truth (red) in the final phase before drop-off, highlighting the model's limitations under abrupt deceleration or occlusion.

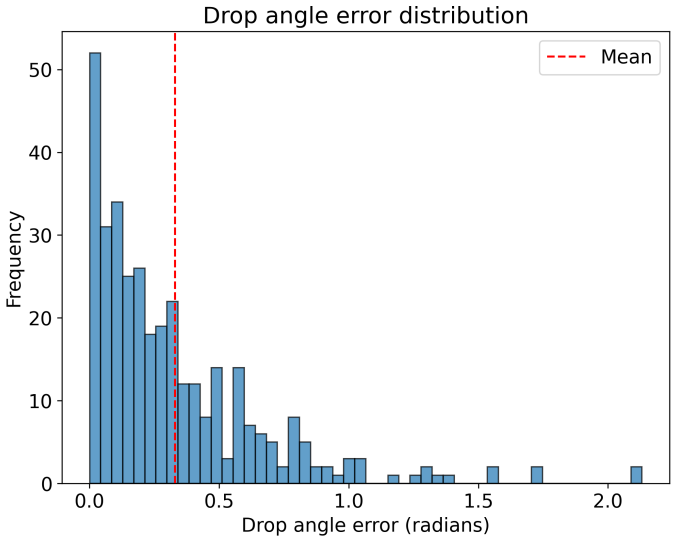


Figure 5.10: Distribution of drop angle prediction errors across test samples at $T = 300$. Most predictions fall below 0.5 radians, with a heavy tail indicating occasional larger errors. The red dashed line marks the mean error.

5.2.3 Wheel Angular Trajectory Prediction

Table 5.7 reports mean absolute errors (MAE) across varying input horizons. Even with only 100 frames observed, the model achieves an MAE of 0.108 radians, equivalent to under 6° of error. This drops to just 0.034 radians (2 degrees) at $T = 300$. By $T = 500$, trajectory MAE falls below 0.025 radians (1.4 degrees), demonstrating near-perfect alignment with the ground truth. Drop-off angular errors follow the same trend, shrinking from 0.189 rad at $T = 100$ to only 0.038 radians at $T = 500$.

Table 5.7: Wheel angular trajectory prediction performance across input horizons. % CI (in radians).

Input Horizon (T)	Trajectory MAE	Drop-off Angular Error
100	0.108 ± 0.014	0.189 ± 0.017
200	0.039 ± 0.004	0.057 ± 0.008
300	0.034 ± 0.003	0.052 ± 0.006
400	0.033 ± 0.005	0.033 ± 0.002
500	0.025 ± 0.002	0.038 ± 0.004

Table 5.8: Baseline comparison for wheel (green) trajectory prediction at $T = 300$. Reported as mean ± 95% CI (in radians).

Method	Trajectory MAE	Drop-off Angular Error
Random Angle Predictor	1.530 ± 0.095	1.545 ± 0.078
Last Angle Predictor	1.491 ± 0.092	1.560 ± 0.010
LSTM Model (Ours)	0.034 ± 0.003	0.052 ± 0.006

Qualitative Analysis Inner wheel trajectory prediction proved to be a much easier task. The wheel rotated far more regularly, with little variation across spins, making its motion highly learnable from visual data. The model produced accurate, stable predictions across the entire sequence, with minimal deviation from the ground truth. Even at longer horizons and shorter input lengths, predicted trajectories remained closely aligned with actual movement, reflecting the consistency and regularity of the wheel's motion. This excellent performance also puts into context just how challenging the ball trajectory prediction is by comparison, where variability and abrupt changes make similar accuracy far more difficult to achieve.

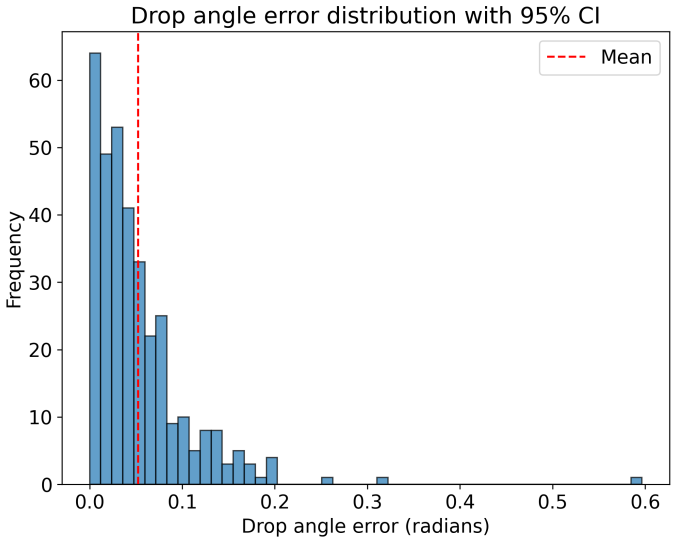


Figure 5.11: Distribution of wheel angle prediction errors across test samples at $T = 300$, with very strong performance with most errors below 0.05 radians (3 degrees).

5.2.4 Robustness and Ablation Studies

Experiments were conducted at $T = 300$ to assess the robustness and sensitivity of the ball trajectory prediction model under varying input conditions. We tested the effects of feature ablation, reduced dataset sizes, noisy inputs, and low-quality data inclusion. Results show that while certain conditions significantly degrade performance, the model retains predictive capability even under constrained or corrupted inputs.

Feature Removal We tested the removal of the derived features introduced in Section 3.8 to assess their contribution to model performance. Removing all derived features resulted in moderate degradation, with trajectory MAE increasing to 0.277 and drop-off error to 0.545. Surprisingly, removing only the derived angular features led to much worse performance (MAE 1.179, drop-off 1.284), suggesting that derived radial features alone cannot compensate for the absence of angular dynamics. In contrast, when all derived features were removed, the model still performed better, indicating that the combined presence of raw features and radial features offers some synergy. Removing only the derived radial features caused a smaller but meaningful drop in accuracy, confirming their complementary value.

Dataset Size Efficiency To examine data efficiency, we trained models on subsets of the training data. Even with just 25% of the original data, the model maintained reasonable

predictive performance. Interestingly, the model trained on 50% of the data performed slightly worse than the 25% model, likely due to stochastic variation or overfitting to a less representative subset. Despite this, performance remained well above baseline predictors, highlighting the model's data efficiency and generalisation ability.

Robustness We introduced Gaussian noise ($\sigma = 0.05$) to all input features during training and evaluation. This led to an increase in trajectory MAE from 0.185 to 0.314, and drop-off angular error from 0.330 to 0.556. While accuracy declined, the model continued to perform well, confirming robustness to moderate input noise.

Invalid Spins Including spins previously flagged as invalid (e.g. occluded, mislabelled, or rule-breaking trajectories) degraded performance across both metrics. This result underscores the model's reliance on high-quality, correctly labelled data and the importance of rigorous curation.

Table 5.9: Robustness and ablation results at $T = 300$. Metrics reported in radians.

Condition	Trajectory MAE	Drop-off Error
Full Model (Baseline)	0.185 ± 0.022	0.330 ± 0.036
Feature Removed:		
All derived features	0.277 ± 0.029	0.545 ± 0.064
Derived angular features	1.179 ± 0.016	1.284 ± 0.094
Derived radial features	0.252 ± 0.025	0.486 ± 0.053
Reduced Dataset:		
50% of training data	1.117 ± 0.018	1.253 ± 0.092
25% of training data	0.977 ± 0.032	1.264 ± 0.088
Gaussian Noise ($\sigma = 0.05$)	0.314 ± 0.030	0.556 ± 0.063
Including Invalid Spins	0.245 ± 0.027	0.488 ± 0.052

5.2.5 Predictive Modelling Summary

The predictive models consistently outperformed naïve baselines across all tasks and input horizons, validating Hypothesis H2. Prediction accuracy improved systematically with longer input sequences, supporting Hypothesis H3. Drop-off time prediction remained the most challenging task due to unpredictable deflector impacts, while wheel trajectory forecasting proved the most stable. Robustness experiments confirmed that while some degradations, particularly the removal of angular features, led to severe performance drops, the model often retained useful predictive capability under noisy or limited data conditions. Across all tasks, the model far outperformed baselines, cutting trajectory and drop-off errors by over 80% in some cases, and proved resilient to noise, feature ablation, and reduced data. These results highlight the importance of feature engineering, smoothing, and careful data curation. Overall, the models demonstrated a strong ability to extract meaningful predictive signals from partial, noisy observations of a chaotic physical system.

Further supplementary training details are provided in Appendix D.

Chapter 6

Discussion

6.1 Recap of Key Results

This research demonstrates that parts of chaotic, noisy physical systems can be meaningfully forecasted using purely visual observation, without embedded physical modelling. Focusing on roulette ball drop-off dynamics, three major outcomes were achieved.

First, the computer vision pipeline reliably extracted accurate ball and wheel trajectories from uncontrolled YouTube footage. YOLOv11-based segmentation, geometric correction, and smoothing achieved high detection precision and recall despite many obstacles, validating Hypothesis H1.

Second, predictive models trained on early-phase trajectories consistently outperformed naïve baselines across all tasks. Drop-off timing and angular trajectory forecasting improved systematically with longer input windows, supporting Hypotheses H2 and H3. Models learned real dynamic structure rather than simple extrapolation.

Third, robustness experiments showed that performance degraded gracefully under added noise, reduced data, and missing features, confirming the resilience of the learned models.

Together, these results suggest that useful predictive structure can be identified in chaotic systems using vision and data-driven learning alone, with potential implications for modelling complex, partially observed real-world systems.

6.2 Interpretation of Results

The results show that reliable short-term predictability is achievable when focusing on early-phase dynamics, particularly around the critical drop-off transition. Machine learning models were able to extract and exploit subtle, repeatable patterns from noisy video alone, without relying on physical calibration or controlled environments. In some tasks (such as predicting the wheel trajectory), the model was able to forecast long-term dynamics with impressive accuracy, even when trained on short input windows.

However, prediction performance for accurately forecasting the precise time of drop-off remained fundamentally limited. The model lacked explicit knowledge of deflector positions, introducing structural ambiguity it could not resolve from early trajectory data alone. Deflectors on roulette

wheels vary both in radial height and orientation, and crucially, their relative positions differed across spins due to inconsistent wheel orientation between videos. Even after geometric correction to simulate a top-down view, the lack of standardised deflector alignment meant that identical ball trajectories could produce different outcomes depending on unseen impacts. Some forecasting errors were therefore not driven by chaotic behaviour itself, but by missing spatial context that was irreducible given the available inputs.

Despite this, the results for drop-off timing prediction remain impressive, especially considering the degree of variation and noise in the dataset.

The vision pipeline, while successful, introduced a secondary dependency: predictive accuracy was sensitive to how effectively invalid trajectories were filtered. Model performance also improved with the addition of basic derived features, though it still performed surprisingly well using only raw trajectory data. This highlights the critical importance of both rigorous data curation and thoughtful feature engineering for enabling robust prediction from noisy visual input.

While results were achieved from a single source, the modular structure of the system suggests potential for generalisation. Expanding to include videos from other channels is technically straightforward in terms of the pipeline, with sourcing suitable footage remaining one of the greatest challenges. Nevertheless, this expansion would be useful to fully validate real-world applicability.

Overall, the findings affirm that data-driven learning can meaningfully tame certain aspects of chaos, but that true forecasting near chaotic transitions faces unavoidable, physics-imposed boundaries.

6.3 Practical Implications

The findings of this research suggest that vision-based machine learning can successfully forecast structured dynamics even in chaotic, noisy environments. Although roulette is a stylised system, the principles extend far beyond: to sports analytics, industrial monitoring, biological systems, and any setting where direct physical modelling is infeasible.

However, the work also highlights critical realities often understated in theoretical studies. Robust, high-quality preprocessing is not a minor engineering detail but a core requirement. Detection, smoothing, and geometric normalisation proved just as important as model design in enabling reliable forecasting. Any real-world deployment of similar techniques must invest heavily in robust, adaptive vision pipelines.

Real-time deployment of the system in roulette scenarios is technically achievable. Although the preprocessing pipeline involves compute-intensive steps such as YOLO-based object detection, geometric correction, and trajectory smoothing, these are largely limited to dataset construction and training. The inference stage is lightweight, relying on basic feature extraction and LSTM forward passes. With suitable optimisations such as model pruning and efficient YOLO deployment, live operation on modest hardware would be realistic. Reliability under varying camera angles, lighting conditions, and occlusions would require further engineering, but the challenge is solvable.

However, applying such a system in a live casino environment would raise serious ethical and legal concerns. Using machine learning to gain predictive advantage in games of chance

undermines the principle of fairness and risks violating gambling regulations. It represents a misuse of technical capability for manipulative purposes, regardless of implementation quality. This research is intended solely for academic purposes. All footage was used with formal permission under ethically approved conditions, and any future work would require similar oversight.

Finally, this research reinforces a broader principle: that meaningful prediction in complex systems often depends less on fully modelling underlying physics, and more on extracting reliable early signals from noisy observations. This offers a powerful strategy for modelling across many fields where chaos, noise, and partial observability are intrinsic to the problem space.

6.4 Reflection on Hypotheses

Hypothesis H1, that vision-based tracking would be feasible and accurate from noisy, uncontrolled footage, was strongly supported. The YOLOv11 detection pipeline, combined with geometric correction and smoothing, consistently produced precise trajectory data under diverse real-world conditions.

Hypothesis H2, that machine learning models trained on early-phase trajectory data would outperform naïve baselines in predicting drop-off time and angular trajectories, was also validated. Across all tasks and input durations, models achieved far lower error rates than baselines, demonstrating genuine learning of system dynamics.

Hypothesis H3, that prediction accuracy would improve with longer input window lengths, was confirmed. Performance systematically improved as the model observed more of the spin's early motion, reflecting the added value of longer pre-drop-off information for forecasting.

Overall, the empirical findings align strongly with the original hypotheses, with the main caveat that structural ambiguity near drop-off imposed irreducible limits on absolute prediction accuracy.

6.5 Future Work

Several clear directions emerge for extending and strengthening this research.

First, future work could focus on learning the positions and orientations of deflectors as latent variables. Explicitly or implicitly inferring deflector layouts could reduce structural ambiguity near drop-off and improve timing predictions, addressing one of the limitations identified in this study without requiring manual annotation.

Second, extending the model to perform statistical inference over the range of possible post-drop-off locations would add depth and is now relatively trivial to implement. With the core vision and trajectory prediction pipeline already in place, this extension would primarily involve incorporating statistical techniques outlined by Small and Tse [47]. It would shift the objective from predicting a single deterministic trajectory to estimating a probabilistic outcome distribution.

Third, with larger datasets available, exploration of Transformer-based architectures could become viable. Transformer models might better capture long-range temporal dependencies

and subtle motion cues compared to LSTM models, potentially improving trajectory forecasting over longer horizons.

Fourth, a hybrid approach embedding lightweight physical constraints into model training, such as enforcing plausible angular velocities or basic conservation laws, could be explored. Such semi-physics-informed methods could balance pure data-driven flexibility with improved robustness to outliers and noise.

Finally, although this research was purely academic, building a prototype real-time system capable of on-the-fly ball and wheel tracking would be a valuable technical exercise. Such work would reveal practical limits around latency, computational cost, and robustness, even if live deployment in gambling environments remains ethically off-limits.

Overall, these extensions would not only refine the predictive performance for roulette but also strengthen the broader applicability of vision-based chaotic system modelling across other domains.

Chapter 7

Conclusion

This project showed that chaotic systems like roulette can be partially forecasted from raw video alone, without any physical modelling or understanding. By building a vision pipeline, extracting structured trajectories from noisy footage, and training deep learning models, meaningful predictive signals were captured despite noise, occlusion, and incomplete observability.

A computer vision system reliably tracked the ball and wheel, normalised perspectives, and created a dataset of over 2700 spins. LSTM-based models, trained on early-phase trajectories, consistently outperformed naïve baselines in predicting drop-off times and angular motion. Prediction accuracy improved with longer observation windows, confirming that early dynamics contain exploitable structure.

More fundamentally, the work showed that the onset of chaos leaves a short but usable window for forecasting. Robust preprocessing, including detection, smoothing, and geometric correction, proved as critical as model design, reinforcing that prediction in complex systems begins with careful observation, not just clever architecture.

Limitations include the restricted dataset and the absence of deflector information, which introduced irreducible uncertainty near drop-off. Expanding across wheels, inferring hidden structures, and testing Transformer-based models are clear paths for future work. Lightweight physical priors could also sharpen predictions without compromising the observational learning focus.

Ultimately, this research demonstrates that chaos is not an absolute barrier but a threshold that careful data-driven methods can approach. The ability to forecast from noisy, partial observations has broader implications for modelling complexity in many domains where full theoretical understanding is out of reach.

Word count: 10925

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X., 2015. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Accessed: 25 April 2025. Available from: `https://www.tensorflow.org/`.

[2] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L. and Savarese, S., 2016. Social LSTM: Human Trajectory Prediction in Crowded Spaces. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp.961–971.

[3] Bai, S., Kolter, J.Z. and Koltun, V., 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv.1803.01271* [Online]. Accessed: 25 April 2025. Available from: `https://doi.org/10.48550/arXiv.1803.01271`.

[4] Bass, T.A., 1985. *The Eudaemonic Pie*. Houghton Mifflin.

[5] Beauchemin, S.S. and Barron, J.L., 1995. The computation of optical flow. *ACM Comput. Surv.*, 27(3), pp.433–466.

[6] Bengio, Y., Simard, P. and Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), pp.157–166.

[7] Bishop, C.M., 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag.

[8] Brunton, S.L., Noack, B.R. and Koumoutsakos, P., 2020. Machine Learning for Fluid Mechanics. *Annual Review of Fluid Mechanics*, 52(1), pp.477–508.

[9] Casdagli, M., 1989. Nonlinear prediction of chaotic time series. *Physica D: Nonlinear Phenomena*, 35(3), pp.335–356.

[10] Chattopadhyay, A., Hassanzadeh, P. and Subramanian, D., 2020. Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, 27, pp.373–389.

[11] Cho, K., Merriënboer, B. van, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. Learning Phrase Representations using RNN Encoder–Decoder

for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp.1724–1734.

[12] Chollet, F. et al., 2015. *Keras*. Accessed: 25 April 2025. Available from: `https://keras.io`.

[13] Claasen, P.J. and De Villiers, J.P., 2024. Video-based sequential Bayesian homography estimation for soccer field registration. *Expert Systems with Applications*, 252.

[14] Crutchfield, J.P., Farmer, J.D., Packard, N.H. and Shaw, R.S., 1986. Chaos. *Scientific American*, 255(6), pp.46–57.

[15] Düben, P. and Bauer, P., 2018. Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development Discussions*, pp.1–17.

[16] Edward and Patrick, 2025. The Roulette Channel [Online]. Accessed: 25 April 2025. Available from: `https://youtube.com/@TheRouletteChannel`.

[17] Farmer, J.D. and Sidorowich, J.J., 1987. Predicting chaotic time series. *Phys. Rev. Lett.*, 59(8), pp.845–848.

[18] FFmpeg Developers, 2000–2025. *FFmpeg*. Accessed: 25 April 2025. Available from: `https://ffmpeg.org/`.

[19] Fischler, M.A. and Bolles, R.C., 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), pp.381–395.

[20] frameinterpolation, D., Mathew, R. and Taubman, D., 2019. Temporal Frame Interpolation With Motion-Divergence-Guided Occlusion Handling. *IEEE Transactions on Circuits and Systems for Video Technology*, 29.

[21] Free Walking Tour Salzburg, 2020. Roulette wheel photo. `https://unsplash.com/photos/WLNdV3xC-fI`. Accessed: 25 April 2025.

[22] Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep Learning*. The MIT Press.

[23] Greff, K., Srivastava, R.K., Koutnik, J., Steunebrink, B.R. and Schmidhuber, J., 2017. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10), pp.2222–2232.

[24] Hartley, R. and Zisserman, A., 2004. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge University Press.

[25] Hidayatullah, P., Syakrani, N., Sholahuddin, M.R., Gelar, T. and Tubagus, R., 2025. YOLOv8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review. *arXiv preprint arXiv:2501.13400* [Online]. Accessed: 25 April 2025. Available from: `https://doi.org/10.48550/arXiv.2501.13400`.

[26] Hochreiter, S. and Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation*, 9, pp.1735–1780.

[27] Jocher, G. and Qiu, J., 2024. *Ultralytics YOLO11*. Accessed: 25 April 2025. Available from: `https://github.com/ultralytics/ultralytics`.

[28] Kingma, D.P. and Ba, J., 2015. Adam: A Method for Stochastic Optimization. In: Y. Bengio and Y. LeCun, eds. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[29] Laskar, J., 1990. The chaotic motion of the solar system: A numerical estimate of the size of the chaotic zones. *Icarus*, 88(2), pp.266–291.

[30] LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep Learning. *Nature*, 521, pp.436–444.

[31] Lipton, Z.C., Berkowitz, J. and Elkan, C., 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv preprint arXiv:1506.00019* [Online]. Accessed: 25 April 2025. Available from: `https://doi.org/10.48550/arXiv.1506.00019`.

[32] Lorenz, E.N., 1963. Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences*, 20(2), pp.130–141.

[33] May, R.M., 1976. Simple mathematical models with very complicated dynamics. *Nature*, 261(5560), pp.459–467.

[34] Mohamed Noor, N., Abdullah, M.M.A.B., Yahaya, A.S. and Ramli, N., 2014. Comparison of Linear Interpolation Method and Mean Method to Replace the Missing Values in Environmental Data Set. *Materials Science Forum*, 803, pp.278–281.

[35] ONNX Contributors, 2017–2025. *Open Neural Network Exchange (ONNX)*. Accessed: 25 April 2025. Available from: `https://onnx.ai/`.

[36] Ott, E., 2002. *Chaos in Dynamical Systems*. 2nd ed. Cambridge University Press.

[37] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library.

[38] Pathak, J., Hunt, B., Girvan, M., Lu, Z. and Ott, E., 2018. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. *Phys. Rev. Lett.*, 120.

[39] Pathak, J., Lu, Z., Hunt, B.R., Girvan, M. and Ott, E., 2017. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12).

[40] Petty, M., 2012. Modeling and validation challenges for complex systems. *Spring Simulation Interoperability Workshop 2012, 2012 Spring SIW*, pp.178–187.

[41] Raissi, M., Perdikaris, P. and Karniadakis, G., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, pp.686–707.

[42] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp.779–788.

[43] Redmon, J. and Farhadi, A., 2018. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767* [Online]. Accessed: 25 April 2025. Available from: `https://doi.org/10.48550/arXiv.1804.02767`.

[44] Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N. and Prabhat, 2019. Deep learning and process understanding for data-driven Earth system science. *Nature*, 566(7743), pp.195–204.

[45] Roboflow, 2020–2025. *Roboflow: Annotate, Train, Deploy Computer Vision Models*. Accessed: 25 April 2025. Available from: `https://roboflow.com/`.

[46] Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning representations by back-propagating errors. *Nature*, 323, pp.533–536.

[47] Small, M. and Tse, C.K., 2012. Predicting the outcome of roulette. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(3).

[48] Smeulders, A.W.M., Chu, D.M., Cucchiara, R., Calderara, S., Dehghan, A. and Shah, M., 2014. Visual Tracking: An Experimental Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7), pp.1442–1468.

[49] Song, W., Jiang, S., Camps-Valls, G., Williams, M., Zhang, L., Reichstein, M., Vereecken, H., He, L., Hu, X. and Shi, L., 2024. Towards data-driven discovery of governing equations in geosciences. *Communications Earth & Environment*, 5(1).

[50] Strogatz, S.H., 2015. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. 2nd ed. CRC Press.

[51] Strzalko, J., Grabski, J., Perlikowski, P., Stefanski, A. and Kapitaniak, T., 2009. *LNP Volume 792: Dynamics of Gambling: Origins of Randomness in Mechanical Systems*, *Lecture Notes in Physics*, vol. 792.

[52] Sun, D., Roth, S. and Black, M.J., 2010. Secrets of optical flow estimation and their principles. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp.2432–2439.

[53] Sun, W., 2022. Sports Performance Prediction Based on Chaos Theory and Machine Learning. *Wireless Communications and Mobile Computing*, 2022(1).

[54] Sutskever, I., Vinyals, O. and Le, Q.V., 2014. Sequence to sequence learning with neural networks. *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. pp.3104–3112.

[55] Thorp, E.O., 1969. Optimal Gambling Systems for Favorable Games. *Revue de l'Institut International de Statistique / Review of the International Statistical Institute*, 37(3), pp.273–293.

[56] Thorp, E.O., 1984. *The Mathematics of Gambling*. Gambling Times.

[57] Thorp, E.O., 1998. The invention of the first wearable computer. *Digest of Papers. Second International Symposium on Wearable Computers (Cat. No.98EX215)*. pp.4–8.

[58] Tian, Y., Ye, Q. and Doermann, D., 2025. *YOLOv12: Attention-Centric Real-Time Object Detectors*. Accessed: 25 April 2025. Available from: `https://github.com/sunsmarterjie/yolov12`.

[59] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention Is All You Need. *Advances in neural information processing systems*, 30.

[60] Vlachas, P.R., Byeon, W., Wan, Z.Y., Sapsis, T.P. and Koumoutsakos, P., 2018. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213).

[61] Wolf, A., Swift, J.B., Swinney, H.L. and Vastano, J.A., 1985. Determining Lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16(3), pp.285–317.

[62] Yeh, R.A., Schwing, A.G., Huang, J. and Murphy, K., 2019. Diverse Generation for Multi-Agent Sports Games. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp.4605–4614.

[63] Yohannes, E., Lin, C.Y., Shih, T.K., Thaipisutikul, T., Enkhbat, A. and Utaminingrum, F., 2023. An Improved Speed Estimation Using Deep Homography Transformation Regression Network on Monocular Videos. *IEEE Access*, 11, pp.5955–5965.

[64] yt-dlp Contributors, 2021–2025. *yt-dlp: A youtube-dl fork with additional features and fixes*. Accessed: 25 April 2025. Available from: `https://github.com/yt-dlp/yt-dlp`.

[65] Zhang, Y. and Yang, Q., 2017. An overview of multi-task learning. *National Science Review*, 5(1), pp.30–43.

[66] Zhang, Z., 2000. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11), pp.1330–1334.

[67] Zhao, Z.Q., Zheng, P., Xu, S.T. and Wu, X., 2019. Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11), pp.3212–3232.

Appendix A

Glossary of Terms

Table A.1: Glossary of Roulette Terms

Term	**Definition**
Angular Trajectory	The path traced by an object, such as the roulette ball or wheel, in terms of its angular position over time.
Ball Angular Trajectory	The sequence of angular positions describing the ball's motion along the outer rim up to drop-off.
Ball Angle	The angular position of the ball, measured relative to the inner wheel's angle at the start of the spin and expressed in radians within $[0, 2\pi)$, denoted θ.
Ball Radius	The distance from the wheel's centre to the ball's position in each frame, normalised such that the outer rim is 1 and the centre is 0, denoted r.
Croupier	The casino staff member responsible for operating the roulette wheel and launching the ball.
Deflector	Raised metal obstacles positioned around the upper part of the wheel that the ball can strike, introducing chaotic bounces.
Drop-off	The event where the ball exits stable rim-bound motion and descends into the inner wheel area.
Drop-off Time	The number of frames from the start of observation until the drop-off event occurs, denoted t^*.
Drop-off Zone	The physical region between the outer rim and inner wheel where the ball transitions after losing stable contact with the rim.
Early-Phase Motion	The initial part of the ball's motion along the outer rim before drop-off, while dynamics remain relatively structured.
European Roulette	A roulette wheel format featuring 37 pockets (numbers 0–36) and a single zero, used exclusively in this project.
Frets	Small raised separators between numbered pockets that cause irregular bounces and influence the ball's final resting place.
Green Zero Pocket	The green-coloured pocket numbered "0" on a European roulette wheel, used as a reference point for wheel angular tracking.
House Edge	The casino's statistical advantage over players, typically around –2.7% for European roulette.
Inner Wheel	The rotating part of the roulette setup containing the numbered pockets where the ball ultimately lands.
Numbered Pocket	One of the 37 compartments on a European roulette wheel where the ball can settle, each associated with a number and colour.
Outer Rim	The sloped outer track of the roulette wheel where the ball initially travels after being launched.
Spin	A full sequence from the ball's launch around the outer rim to its eventual settling in a pocket.
Spin Input Window	The number of frames from the detected spin start that are fed into the model as input, denoted T.
Spin Splitting	The process of segmenting continuous video footage into discrete spin events based on observed signals, such as croupier hand movements.
Wheel Angular Trajectory	The sequence of angular positions describing the inner wheel's rotation during a spin.

Appendix B

Dataset Permission

Permission to use video footage from *The Roulette Channel* (YouTube) was obtained in writing prior to commencing dataset construction. The following is an excerpt from the formal confirmation email received:

> "We are happy to support your project with whatever you need. As long as this material helps you with your dissertation or academic work, you have our consent to use it for your purposes as you asked for. We would happily be mentioned as The Roulette Channel (Edward & Patrick) in the credits."

The Roulette Channel team also expressed interest in the academic focus of the project, clarifying that their consent covers downloading, processing, and analysis of their publicly available videos for academic use.

A full record of the email exchange confirming permission is available upon request.

Appendix C

YOLO Training Details

The YOLOv11 segmentation model was trained on a GPU-equipped DigitalOcean droplet. Dataset creation, video preprocessing, and label generation were all conducted on this server. The model was exported to ONNX format for efficient inference, with deployment handled via Docker containers optimised with CUDA. Inference performance reached approximately 0.8 milliseconds per frame across 20 parallel video streams.

DO-1 (DigitalOcean Server):

- **CPU:** 20 vCPUs
- **RAM:** 240 GB
- **GPU:** 1× NVIDIA H100 80GB
- **Storage:** 720 GB NVMe boot disk + 5 TB NVMe scratch disk

Additional considerations:

- Full disk snapshots were regularly taken to safeguard progress during dataset construction.
- An inode exhaustion issue was encountered due to excessive label files. The pipeline was modified to zip and archive completed label batches, freeing up inodes and maintaining system stability.
- System usage was monitored with `nvidia-smi`, `htop`, and lightweight shell scripts.

Model Configuration

Table C.1: YOLOv11 Training Configuration and Hyperparameters

Model Variant	YOLOv11s-seg.pt (small, segmentation)
Framework	PyTorch 2.1, Ultralytics 8.3
Input Resolution	640 × 640
Batch Size	64
Initial Learning Rate	1e-3
Scheduler	Cosine Decay
Optimizer	Adam
Training Duration	150 epochs
Inference Speed	0.8 ms/frame

Annotation Details

Table C.2: Annotated Frame Statistics

Annotation Tool	Roboflow (polygon masks)
Annotated Frames	596
Classes	Ball, Green Zero, Hand
Test, Train, Val Split	356, 120, 120

Augmentation Parameters

Table C.3: Data Augmentation Pipeline (2 augmentations per original frame)

Flip	Horizontal: true, Vertical: true
Brightness	$\pm$18% (darken and brighten)
Hue Shift	$\pm$15°
Shear	Horizontal: 15°, Vertical: 14°
Rotation	$\pm$45°
Gaussian Noise	1.33% pixel perturbation

(a) Augmented Frame A (b) Augmented Frame B

Figure C.1: Example augmented training frames with applied transformations (e.g., rotation, shear, noise).

Effect of Augmentation

Table C.4: YOLO Segmentation (Mask) Performance With and Without Augmentations

Class	**With Augmentation**				**No Augmentation**			
	P	R	mAP50	mAP50-95	P	R	mAP50	mAP50-95
All	0.997	0.969	0.983	0.726	0.932	0.887	0.933	0.668
Ball	0.990	0.990	0.994	0.523	0.955	0.840	0.908	0.453
Green	1.000	0.917	0.958	0.744	1.000	0.908	0.954	0.744
Hand	1.000	1.000	0.995	0.910	0.840	0.913	0.938	0.808

Localisation Distances:

- **With Augmentation:**
  Mean = 1.36, Max = 6.67, Min = 0.09, Std = 1.05
- **No Augmentation:**
  Mean = 1.56, Max = 12.32, Min = 0.12, Std = 1.62

Appendix D

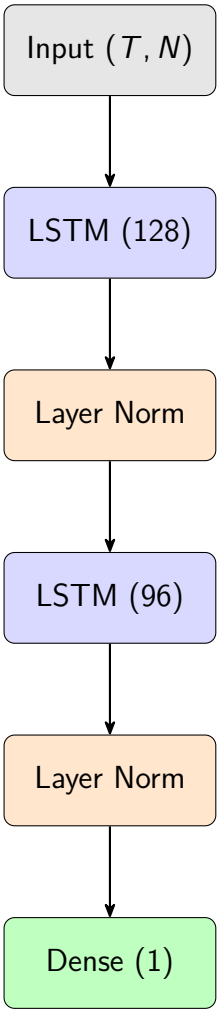
Prediction Training Details

All predictive modelling (based on LSTM architectures) and experimental analysis were performed locally on a MacBook Pro using Apple's Metal Performance Shaders (MPS) for GPU acceleration. Models were trained with TensorFlow, incorporating techniques such as early stopping, dynamic learning rate scheduling, and checkpointing to ensure robust convergence. Experiments included horizon variation, ablation studies, and robustness testing.

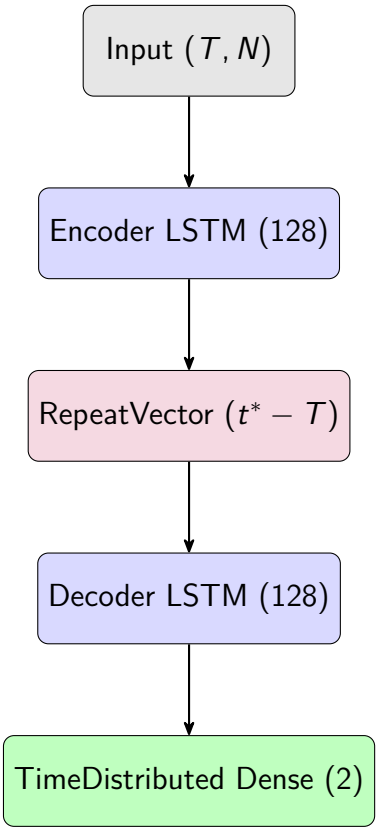
MBP-M4 (2024 MacBook Pro M4 Chip):

- **CPU:** Apple M4 chip
- **RAM:** 16 GB
- **GPU:** Integrated Apple M4 GPU (MPS-accelerated)
- **Storage:** 512 GB SSD

Model Architecture Diagrams



(a) Drop-Off Time model (sequence-to-scalar).



(b) Ball and Wheel models (sequence-to-sequence).

Figure D.1: Model architectures. The Drop-Off Time model reduces the input sequence to a scalar output through a stacked LSTM encoder with layer normalisation. The Ball and Wheel models use a shared encoder-decoder structure with RepeatVector and TimeDistributed layers to predict angular sequences. Despite the simpler nature of the wheel task, the higher-capacity sequence-to-sequence architecture proves effective due to the structure of the input features.

Training Curves for Models at all values of T

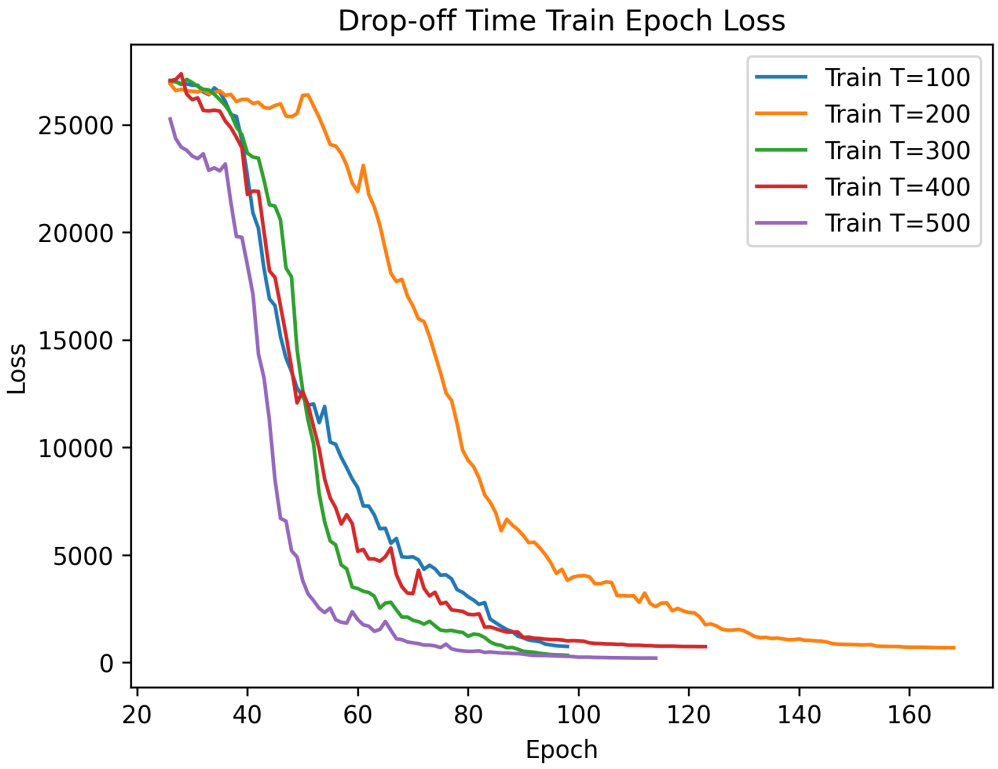


Figure D.2: Training loss for the drop-off time model across input horizons. The plot is cropped to start from epoch 25 to exclude extreme early loss values for visual clarity.

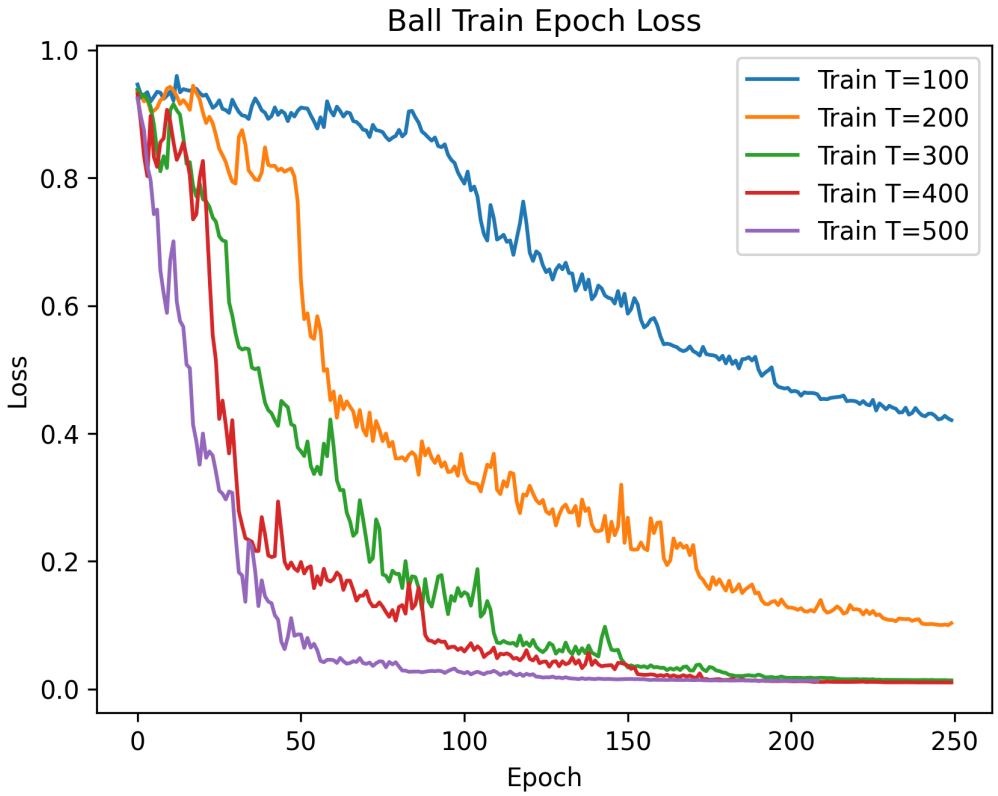


Figure D.3: Training loss for the ball trajectory model across input horizons.

Figure D.4: Training loss for the wheel trajectory model across input horizons (first 60 epochs, after which the models converged).

Hyperparameter Search Space

Drop-off Time Model

Table D.1: Hyperparameter search space for Drop-off Time model (Sequence-to-Scalar).

Parameter	Search Range
LSTM Units	64, 96, 128, 160, 256
Learning Rate	1×10^{-3} to 1×10^{-5}
Weight Decay	1×10^{-6} to 1×10^{-4}
Batch Size	32, 64, 128
Dropout Rate (if added)	0.1 to 0.4 (step 0.1)
Number of Layers	1, 2, 3

Ball Trajectory Model

Table D.2: Hyperparameter search space for Ball Trajectory model (Sequence-to-Sequence).

Parameter	Search Range
Encoder LSTM Units	64, 96, 128, 160, 256
Decoder LSTM Units	64, 96, 128
Learning Rate	1×10^{-3} to 1×10^{-5}
Dropout Rate (if added)	0.2 to 0.5 (step 0.1)
Weight Decay	1×10^{-6} to 1×10^{-4}
Batch Size	32, 64
Number of Layers (Encoder/Decoder)	1, 2

Wheel Trajectory Model

Table D.3: Hyperparameter search space for Wheel Trajectory model (Sequence-to-Sequence).

Parameter	Search Range
Encoder LSTM Units	64, 96, 128, 160
Decoder LSTM Units	32, 64, 96, 128, 160
Learning Rate	1×10^{-3} to 1×10^{-5}
Dropout Rate (if added)	0.1 to 0.4 (step 0.1)
Weight Decay	1×10^{-6} to 1×10^{-4}
Batch Size	32, 64, 128
Number of Layers (Encoder/Decoder)	1, 2