# TRAJECTORY LEARNING IN 2D SPACE AND SPATIO-TEMPORAL TRACKING

# CONTENT

# INTRODUCTION

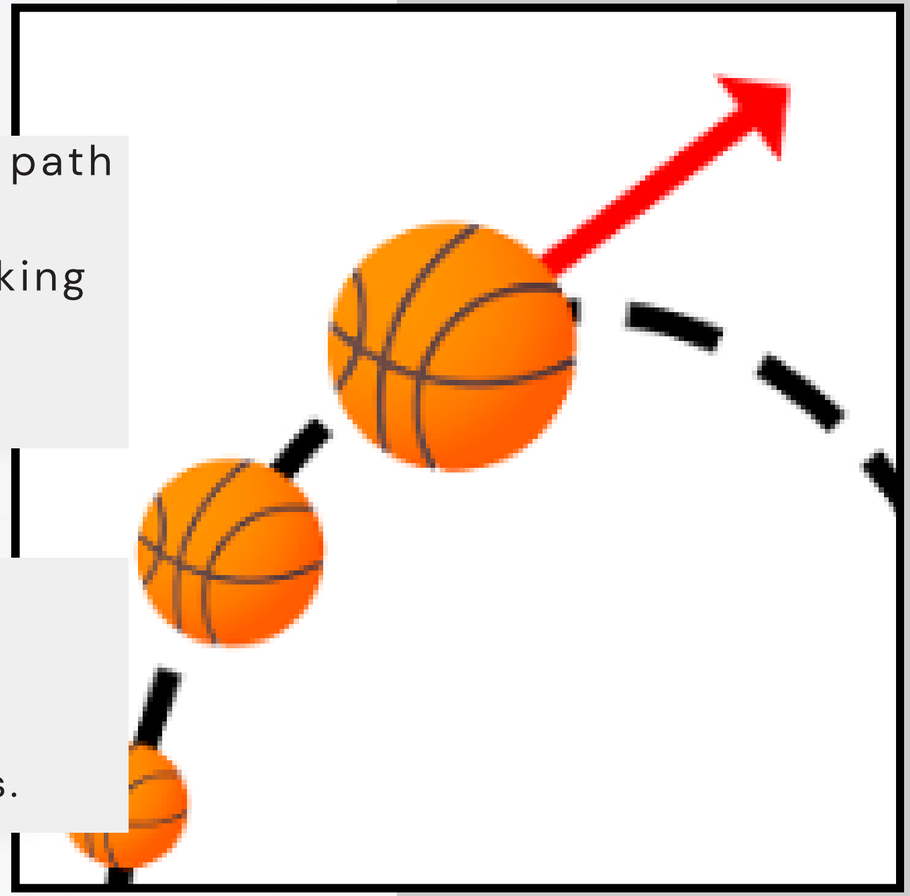
# INTRODUCTION

## What is Trajectory Learning?

- The process of understanding and predicting the path an object follows over time in 2D or 3D space.
- Used in domains requiring accurate position tracking over time, e.g., sports analytics, robotics, or autonomous systems.

## Why Learn Trajectories in 2D Space?

- Many real-world motions occur in 2D space (e.g., shuttlecock in badminton, robot navigation).
- Easier to simulate and visualize than 3D, making it a starting point for understanding trajectory behaviors.



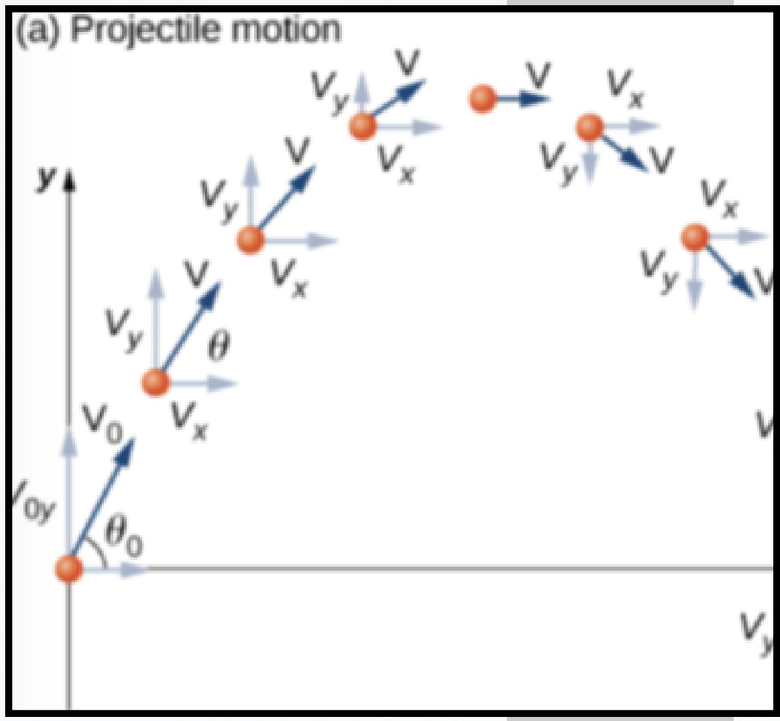
# PHYSICS OF 2D MOTION

# PHYSICS OF 2D MOTION

## Basic Equations of Motion

- Objects moving under the influence of gravity follow predictable trajectories.
- For a projectile (like a shuttlecock), its position at any time ttt is given by:

$$x(t) = v_x \cdot t$$
$$y(t) = v_y \cdot t - \frac{1}{2} g \cdot t^2$$




Where:

- $vx, vy$ : Initial velocity components in the x and y directions.
- $g$ : Acceleration due to gravity.

# PHYSICS OF 2D MOTION

## Effect of Drag and Air Resistance

Real-world trajectories deviate due to forces like drag:

$$F_{drag} = -c_d \cdot v^2$$

Where $c_d$ is the drag coefficient, and $v$ is the velocity.

⚠ Shuttlecock motion is significantly affected by drag, causing steeper descent compared to ideal parabolas.

# PHYSICS OF 2D MOTION

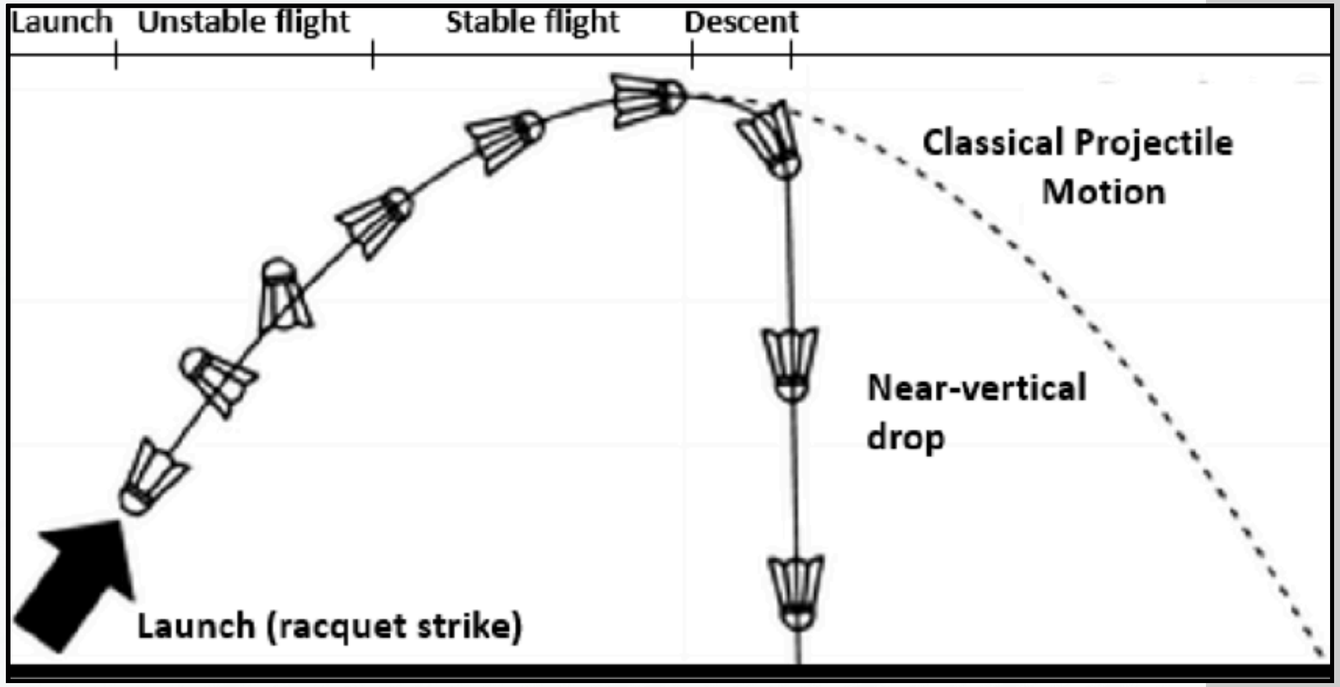
## Shuttlecock Motion

Design Features:

- Feathered structure increases drag.
- High air resistance slows motion quickly, ensuring predictable landings.

Unique Motion Characteristics:

- High initial velocity due to the player's hit.
- Rapid deceleration shortly after.
- Asymmetric curve with a sharp downward slope.



# NEURAL NETWORKS FOR TRAJECTORIES

# NEURAL NETWORKS FOR TRAJECTORIES

## Why Use Neural Networks for Trajectory Learning?

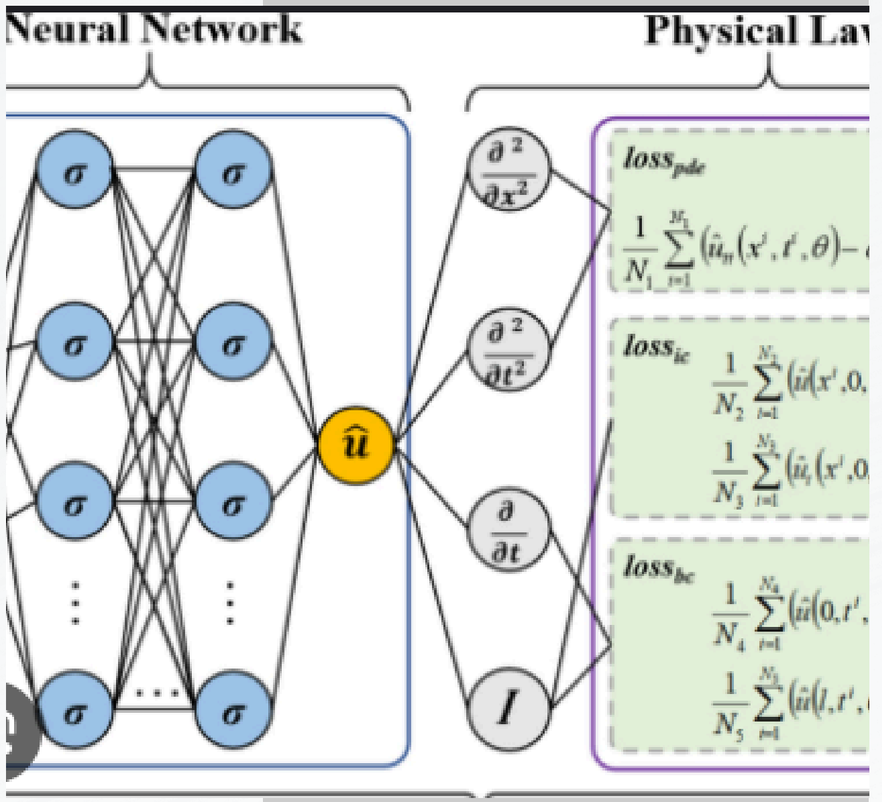
- **Modeling Complex Dynamics:**
  - Neural networks can approximate non-linear functions, making them ideal for learning physical trajectories influenced by drag and other forces.
- **Handling Noisy Data:**
  - Real-world measurements often contain noise. Neural networks can learn to smooth data and infer missing information.
- **Flexibility:**
  - Adaptable to varying conditions, such as different velocities, angles, or drag coefficients.

# NEURAL NETWORKS FOR TRAJECTORIES

## Physics-Informed Neural Networks (PiNNs)

### Definition:

- Neural networks designed to integrate the laws of physics directly into their training process, ensuring predictions remain consistent with physical principles.




### Core Idea:

- Augment the learning process by combining:
  - Data-Driven Learning: From observed or simulated data.
  - Physics-Based Constraints: From governing equations like Newton's laws, drag force, etc.

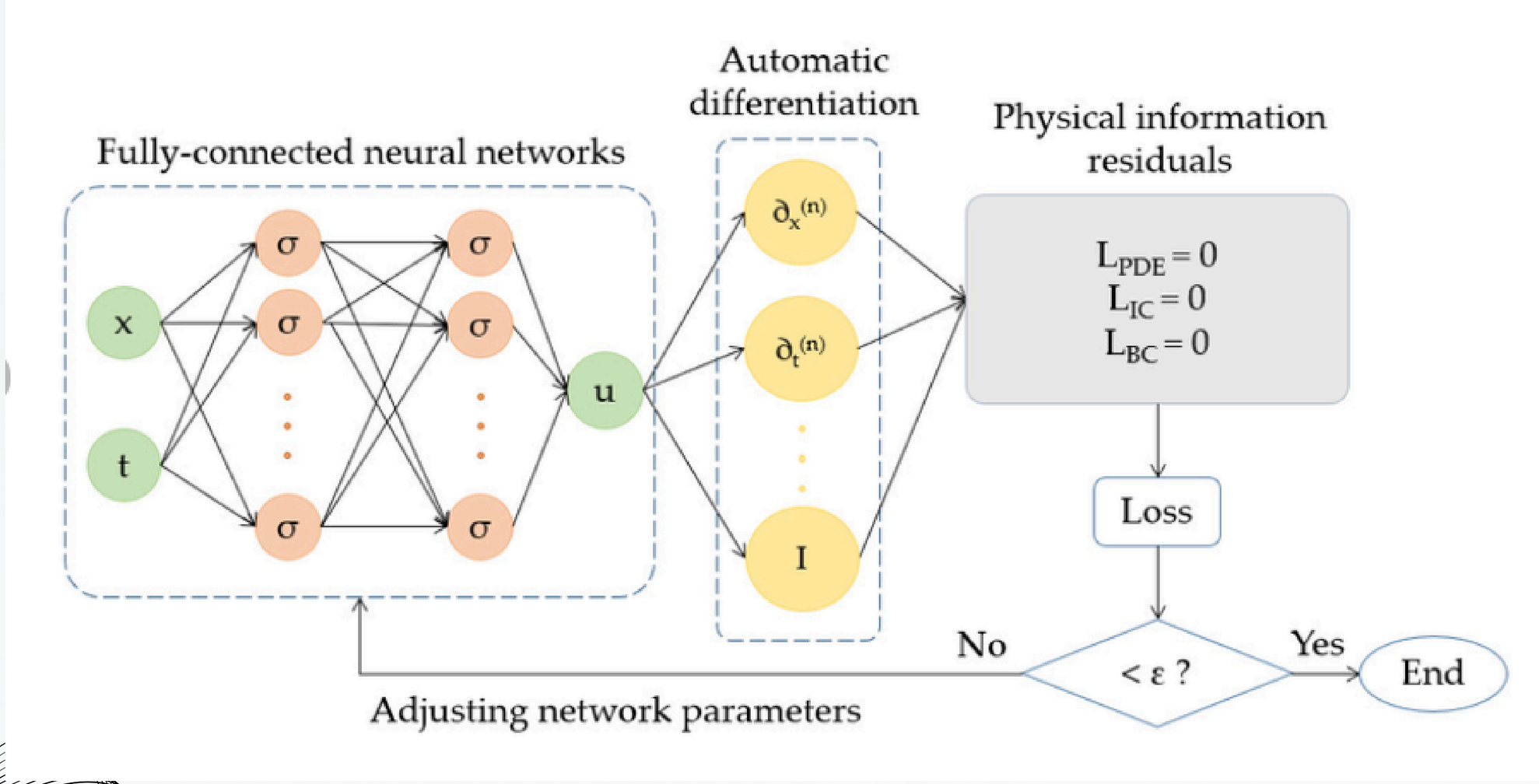
# NEURAL NETWORKS FOR TRAJECTORIES

**Why PiNNs for Trajectory Learning?**

- **Real-World Relevance:**
  - Trajectories are governed by well-defined physical equations (e.g., gravity, air resistance).
- **Advantages of PiNNs:**
  - Data Efficiency: Requires fewer data points as physics acts as a regularizer.
  - Noise Robustness: Predictions remain accurate even with noisy or incomplete data.
  - Generalization: Learns trajectories consistent with physics, even in unseen conditions.

# NEURAL NETWORKS FOR TRAJECTORIES

**Key Components of PiNNs**



# NEURAL NETWORKS FOR TRAJECTORIES

## Key Components of PiNNs

Inputs:

- Time , $t$ initial conditions ($v_0$, $\theta$), or noisy data.

Physics-Driven Loss Function:

- Loss includes terms to enforce consistency with physical equations, e.g.:

$$Loss = MSE(predicted, true) + \lambda \cdot PhysicsResiduals$$

Where λ balances data accuracy with adherence to physics.

Governing Equations as Residuals:

- For example, for motion:

$$PhysicsResidual = |\frac{d^2y}{dt^2} + g - \frac{F_{drag}}{m}|$$

- Residuals penalize deviations from physical laws.

Neural Network:

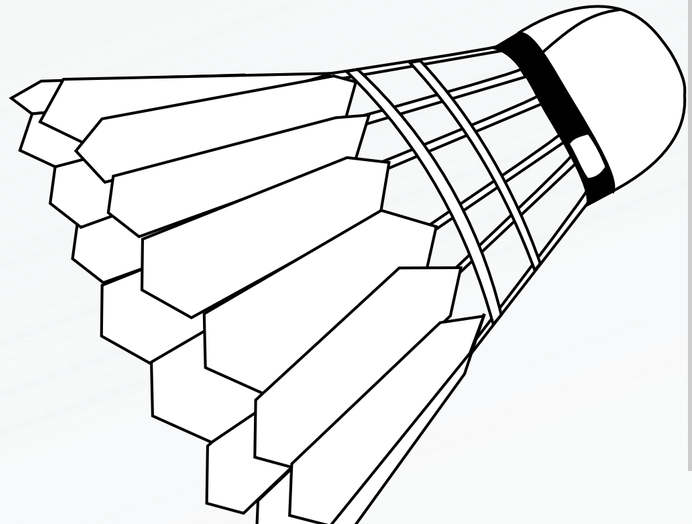
- Standard feedforward architecture, possibly with temporal modeling (e.g., LSTMs).

# SIMULATED DATA GENERATION FOR SHUTTLECOCK TRAJECTORY

# SIMULATED DATA GENERATION FOR SHUTTLECOCK TRAJECTORY

**Why Simulate Shuttlecock Trajectories?**

- **Practical Challenges:** Real-world shuttlecock trajectory data is difficult to collect due to rapid motion and environmental factors.
- **Controlled Environment:** Simulation allows for precise manipulation of variables like air resistance, initial velocity, and angle.
- **Validation Tool:** Provides benchmarks to test models against.



# SIMULATED DATA GENERATION FOR SHUTTLECOCK TRAJECTORY

## Governing Motion Equations for Shuttlecock Data

**Horizontal and Vertical Motion:**

- Derived from Newton's Second Law, considering drag proportional to the square of velocity:

$$F_{drag} = -bv^2$$

- Horizontal Position (x(t)):

$$x(t) = \frac{mv_{x0}}{b}(1 - e^{-\frac{b}{m}t})$$

  Where $v_{x0}$ is the initial horizontal velocity.

- Vertical Position (y(t)):

$$y(t) = \frac{m}{b}\left(v_{y0} - v_t\right) \times \left(1 - e^{-\frac{b}{m}t}\right) + v_t t$$

Where $v_t = \sqrt{\frac{mg}{b}}$ is the terminal velocity.

**Forces Acting on the Shuttlecock:**

- Gravitational force W=mg
- Aerodynamic drag $F_{drag} = -bv^n$
  - Quadratic drag (n=2): Most accurate for shuttlecocks.
- Negligible buoyancy due to low air displacement.

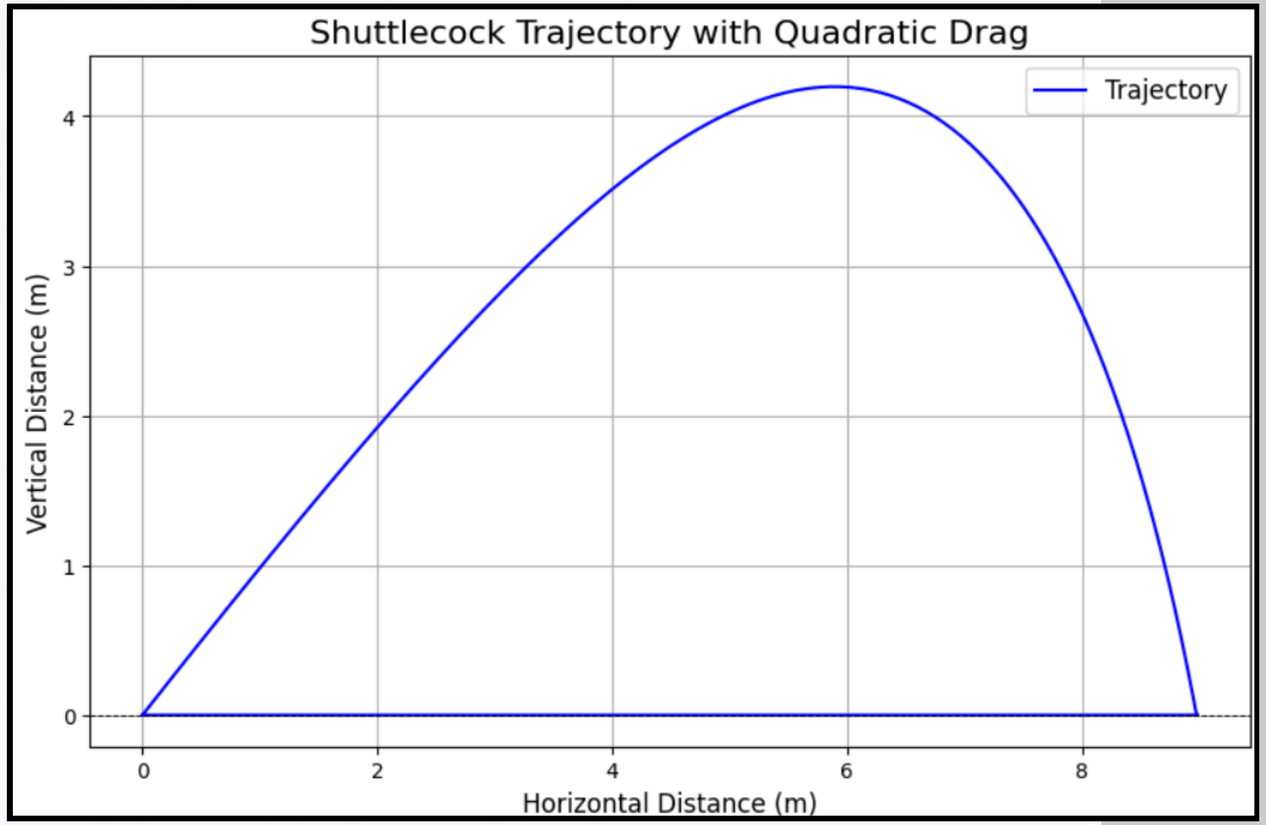
**Trajectory Combination:**

- Combine x(t) and y(t) for the full 2D trajectory.

# SIMULATED DATA GENERATION FOR SHUTTLECOCK TRAJECTORY

## Simulation Process

- **Step 1: Define Initial Conditions**
  - Launch angle (θ): 30°, 45°, 60°.
  - Initial velocity (Vo): Depends on stroke type (e.g., smash, clear).
- **Step 2: Compute Trajectories**
  - Use motion equations to calculate x(t) and y(t) at each time step.
- **Step 3: Add Noise and Missing Data**
  - Introduce random noise to simulate real-world inaccuracies.
  - Simulate missing measurements to test robustness.



# APPLICATION OF PINN TO SHUTTLECOCK TRAJECTORY

# APPLICATION OF PINN TO SHUTTLECOCK TRAJECTORY

**Application to Shuttlecock Trajectory**

- **Simulated Data:**
  - Generate trajectories using equations of motion with drag.
- **Physics-Informed Layers:**
  - Use drag and gravity as constraints in the loss function.
- **Handling Noise:**
  - Incorporate noisy/missing data to train for real-world conditions.

# APPLICATION OF PINN TO SHUTTLECOCK TRAJECTORY

**Benefits of PiNNs in This Context**

- **Trajectory Learning:**
  - Consistent with shuttlecock's real-world motion (e.g., sharp drop due to drag).
- **Reduced Overfitting:**
  - Physics guides the network, reducing reliance on large datasets.
- **Noise Tolerance:**
  - Handles noisy input better than purely data-driven models.

# APPLICATION OF PINN TO SHUTTLECOCK TRAJECTORY

**Model Components**

- Input: Time (t)
- Output: [x,y,vx,vy]
- Structure:
- 3 fully connected hidden layers with 256 neurons each
- Activation: Tanh function
- Output: 4 neurons representing the trajectory state

**Loss Function**

- Data Loss: Measures the difference between predicted (y^) and actual (y) trajectory points:
- Physics Loss: Ensures compliance with Newtonian mechanics
  where:
  - ax,ay: Predicted accelerations (second derivatives computed via autograd).
  - fx,fy: Forces acting on the shuttlecock
  - Cd: Drag coefficient, g: Gravity, v: Speed magnitude.
- Total Loss: Combines data and physics losses with a regularization term (λreg): Total Loss=Data Loss+λreg · Physics Loss

# APPLICATION OF PINN TO SHUTTLECOCK TRAJECTORY

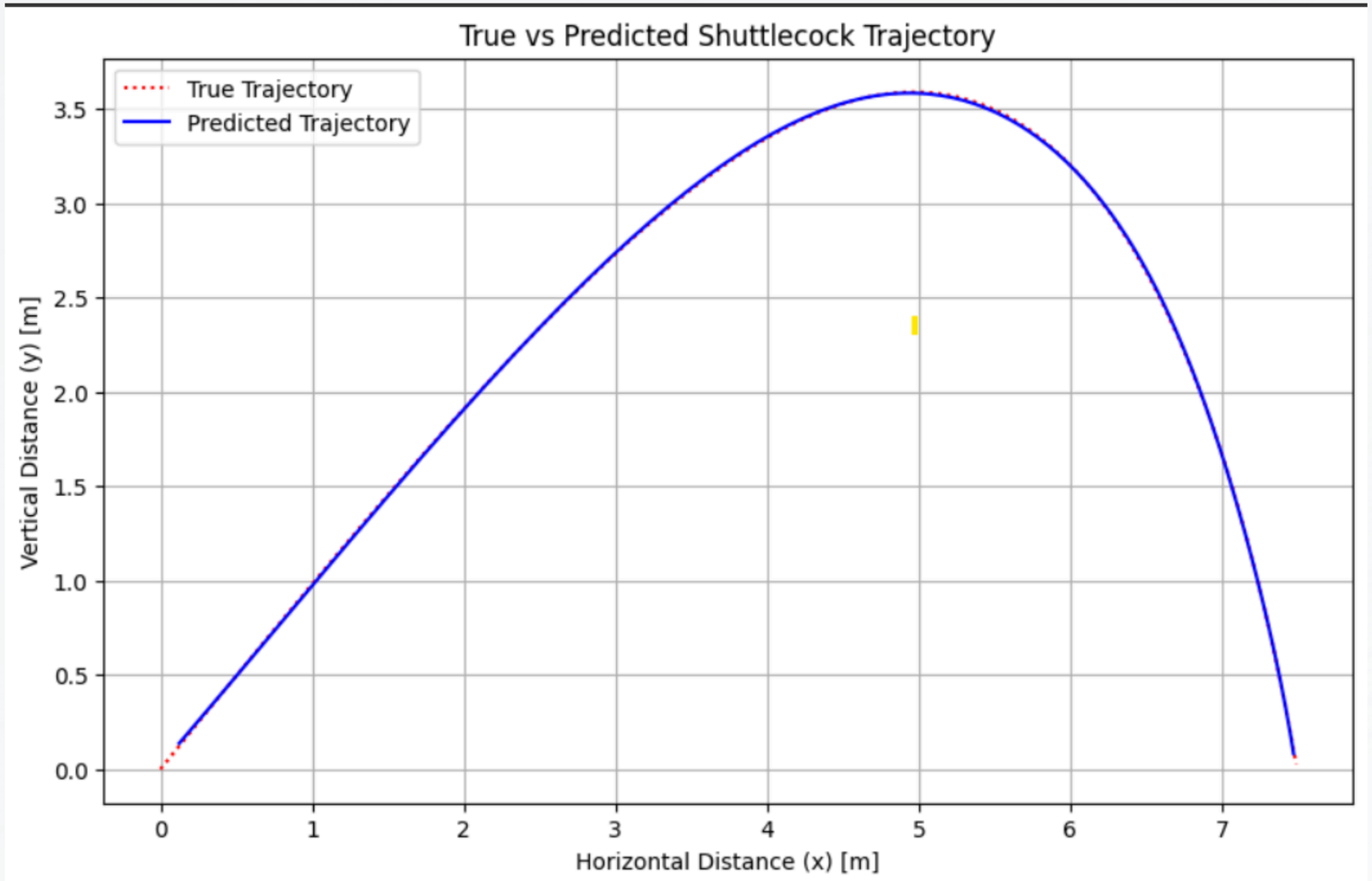
## Training Configurations

- The PiNN was trained under the following configurations:
  - Training on the Full Dataset.
  - Training on only 20 Representative Points.

# RESULTS

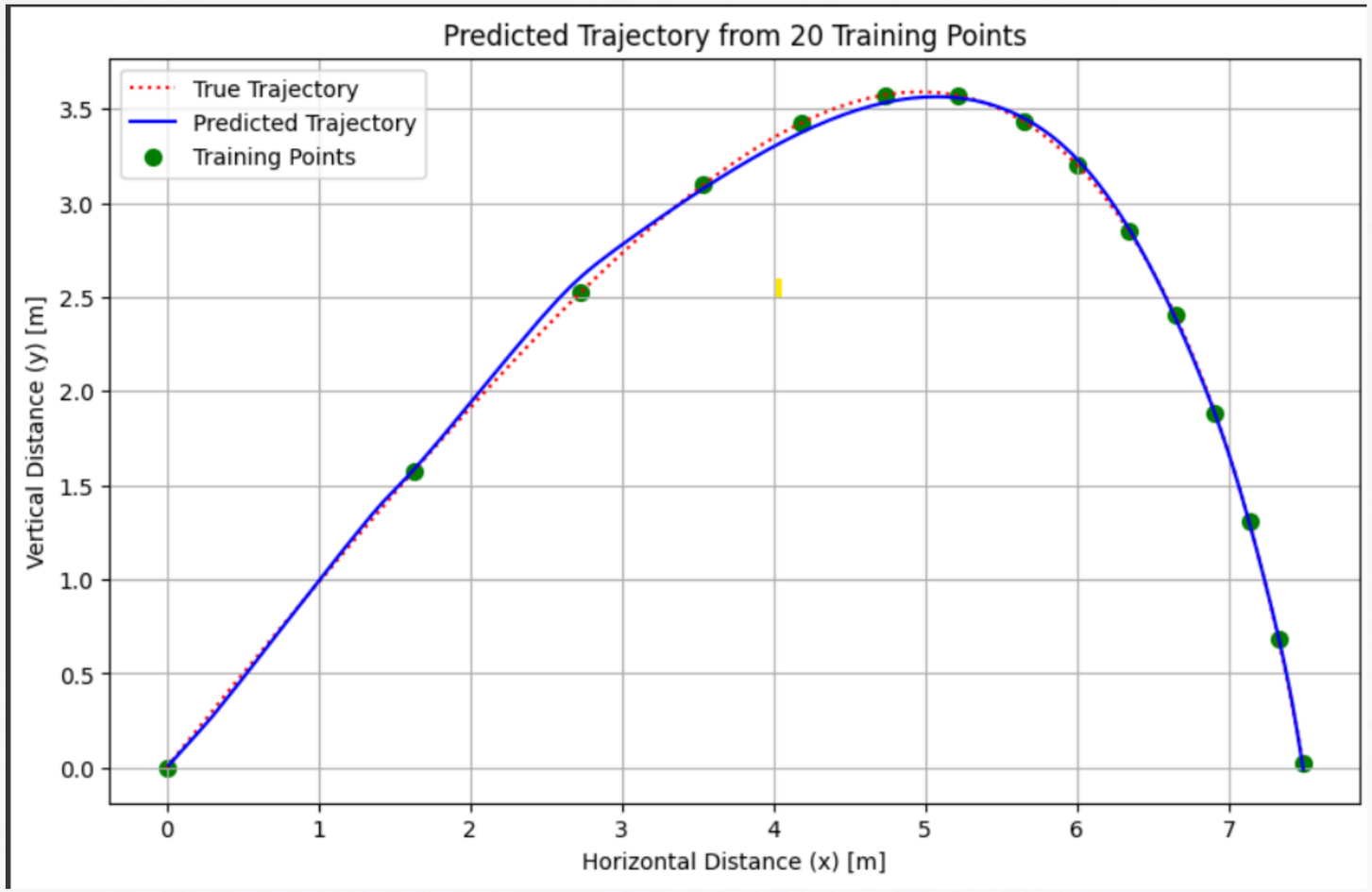
# RESULTS

## Training on the Full Dataset :



# RESULTS

## Training on only 20 Representative Points :



# RESULTS

## Quantitative Results

| Metric | Full Dataset | 20 Points Dataset |
|---|---|---|
| Final Loss | 4.82 | 3.55 |
| Detection Accuracy | 100% | 97.63% |
| False Positives (FP) | 0 | 2 |
| False Negatives (FN) | 0 | 2 |
| Inference Latency | 0.0026 seconds | 0.0033 seconds |

# CONCLUSION

# CONCLUSION

Physics-Informed Neural Networks (PiNNs) revolutionize trajectory estimation by embedding physical laws, such as gravity and drag, into neural networks, ensuring accurate and reliable predictions. Their ability to handle noisy or incomplete data makes them particularly robust for modeling shuttlecock motion. PiNNs have practical applications in sports analytics for optimizing player performance, robotics for precise object tracking, and broader fields like autonomous navigation and meteorology. By combining the rigor of physics with AI's adaptability, PiNNs offer a transformative approach to trajectory prediction and real-world tracking challenges.

# THANK'S FOR YOUR ATTENTION