Distributed IoT Data Acquisition and Analytics for Animal Behavior Analysis: A Technical Report

Kudret Esmer
Politecnico di Milano
Milan, Italy
kudretlesmer@gmail.com

Abstract

Animal behavior analysis is essential for effective wildlife management and agricultural practices, especially for monitoring remote herds. This report details the development and deployment of a distributed IoT data acquisition and analytics system designed to simulate and analyze animal movements in real-time. A simulator was created to generate realistic animal movement data, and an inference module was developed to detect anomalies in these movements. The entire system was deployed on a Kubernetes cluster to evaluate its scalability and performance under varying computational loads. The approach demonstrated high accuracy in identifying deviations from normal behavioral patterns, providing valuable insights for livestock management and conservation efforts.

Keywords

Animal behavior analysis, IoT, distributed systems, anomaly detection

1 Introduction

The study of animal behavior has seen significant advancements with the integration of GPS-based radiotelemetry into animal ecology. This technological progression has facilitated detailed insights into movement patterns, habitat use, and ecological interactions at various scales, providing a robust foundation for understanding complex animal behaviors [1]. As ecological research increasingly emphasizes cross-scale analysis, the movement data captured through GPS technology has proven invaluable, allowing researchers to explore the intricacies of animal movement, such as speed, direction, and the influence of environmental factors [2].

At the same time, the development of distributed behavioral models has opened new avenues for simulating and analyzing collective behaviors like flocking, herding, and schooling. These models have significantly enhanced our understanding of the underlying mechanisms driving these behaviors, offering insights that are crucial for both ecological studies and practical applications in wildlife management and agriculture [3].

Parallel to these advancements, the proliferation of the Internet of Things (IoT) has introduced both opportunities and challenges in the field of anomaly detection. IoT networks generate vast amounts of data from diverse sources, necessitating robust and adaptive methods to identify unusual patterns and behaviors. Recent research in IoT anomaly detection has explored various machine learning techniques to address these challenges, underscoring the importance of effective detection systems in maintaining the security and reliability of IoT networks [4].

This paper presents a comprehensive approach that combines the strengths of modern tracking technologies, distributed behavioral models, and IoT-based anomaly detection to monitor and analyze animal behavior in real-time. By simulating a herd of animals and employing advanced anomaly detection techniques, this work aims to provide a scalable and efficient solution for detecting deviations in animal behavior, with significant implications for both wildlife conservation and agricultural management.

The remainder of this paper is organized as follows: Section 2 outlines the problem definition and objectives of the project. Sections 3 and 4 detail the design and implementation of the simulator, inference module, and anomaly detection model. Section 5 introduces the parameters that are specifically chosen and discusses the results. Section 6 concludes the work.

,
.

2 Problem Definition and Objectives

Monitoring and analyzing the behavior of animal herds in real-time is crucial for effective wildlife management, livestock monitoring, and ecological research. Traditional methods of observation are often labor-intensive and limited in scope, making it difficult to detect subtle changes in behavior that may indicate anomalies such as illness, predator presence, or environmental stressors.

The objective of this project is to develop a simulation of a herd of animals and construct an application capable of detecting anomalies in their behavior. The simulation will model the movement and interactions of the herd under normal conditions, including states such as Grazing, Resting and Foraging. The application will utilize real-time data processing and deep learning techniques to identify deviations from these normal states, particularly in response to potential threats or abnormal environmental conditions.

Given the limited availability of anomalous data in real-life scenarios, we approach the problem as an unsupervised learning task. This allows the model to learn the normal patterns of behavior from the available data and identify deviations as potential anomalies, without relying on extensive labeled datasets of anomalous events.

To achieve these objectives, the project will involve:

- Designing and implementing a simulator that accurately models the behavior of a herd under various states.
- Developing an inference module that processes the simulated data in real-time, maintaining a rolling window of observations for each agent in the herd.
- Training and deploying an autoencoder-based anomaly detection model to identify unusual patterns in the herd's movement.

- Testing the system's effectiveness by simulating scenarios that include both normal and anomalous behaviors, such as the Fleeing state, and evaluating the model's performance in detecting these anomalies.

This approach will provide a scalable and efficient method for detecting behavioral anomalies in herds, with potential applications in both wildlife conservation and agricultural management.

3 Simulator Design and Implementation

The simulator is designed to model the movement of a herd of animals by incorporating various behavioral states and interactions within the environment. The simulation is driven by parameters provided in JSON format, which include the number of agents, state definitions, and specifications for herd behavior.

3.1 State Definition and Transition

Each agent's behavior in the simulation is modeled using a Markov process with four distinct states: Grazing, Resting, Foraging, and Fleeing. These states determine the movement characteristics, such as speed and turning angle, of the agents.

State Transition Matrix: The transitions between states are governed by a state transition matrix P, where each element P_{ij} represents the probability of transitioning from state i to state j. The transition matrix for this simulation is:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \tag{1}$$

For example, the element P_{13} represents the probability of transitioning from the state "Grazing" (state 1) to the state "Foraging" (state 3) in the next time step.

Turning Angle and Speed: The turning angle θ_s for each state s is defined as follows:

$$\theta_s = \begin{cases} \text{Uniform}(\theta_{\text{min}}, \theta_{\text{max}}) & \text{if } s \in \{\text{Grazing, Resting}\}, \\ \mathcal{N}(\mu_\theta, \sigma_\theta^2) & \text{if } s \in \{\text{Foraging, Fleeing}\}. \end{cases} \tag{2}$$

The speed v_s of an agent in a given state s is drawn from a uniform distribution based on the state's defined range.

3.2 Modeling Speed, Direction, and Herd Behavior

Neighbor Influence: Herd behavior is simulated by adjusting each agent's speed and direction according to its neighbors. For each agent i, the set of neighbors $\mathcal{N}_i$ is defined as the N agents that are closest to agent i in terms of the Euclidean distance (L2 norm). Mathematically, this is expressed as:

$$\mathcal{N}_i = \left\{ j \mid \text{dist}(i, j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2,\ j \in \text{Top-}N \right\}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ represent the positions of agents i and j, respectively.

The mean speed $\bar{v}_{\text{neigh}}$ and mean direction $\bar{\theta}_{\text{neigh}}$ of the neighbors are calculated as follows:

$$\bar{v}_{\text{neigh}} = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} v_j$$

$$\bar{\theta}_{\text{neigh}} = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \theta_j$$

Final Speed and Direction: The speed v_i and direction θ_i for agent i are then computed as follows:

$$v_i = \alpha \cdot \bar{v}_{\text{neigh}} + (1 - \alpha) \cdot v_{\text{s}} \tag{3}$$

$$\theta_i = \alpha \cdot \bar{\theta}_{\text{neigh}} + (1 - \alpha) \cdot \theta_{\text{s}} \tag{4}$$

where α is the weight factor, $\bar{v}_{\text{neigh}}$ and $\bar{\theta}_{\text{neigh}}$ are the mean speed and direction of the neighbors, and v_{s} and θ_{s} are the agent's random speed and direction.

Special Cases - Fleeing State: A special case is implemented for the Fleeing state: if an agent enters the Fleeing state, all of its neighbors will also transition to the Fleeing state, simulating the natural response of animals to a threat.

3.3 Position Update and GPS Error Simulation

Each agent's position is updated at each time step based on its speed and direction. The position (x_t, y_t) at time t is computed using the following equations:

$$x_{t+1} = x_t + v_t \cdot \cos(\theta_t) \tag{5}$$

$$y_{t+1} = y_t + v_t \cdot \sin(\theta_t) \tag{6}$$

GPS Error Simulation: To enhance realism, GPS errors are introduced by adding Gaussian noise to the agents' positions. The GPS error is modeled as:

$$\text{GPS error} \sim \mathcal{N}(0, \sigma_{\text{x}}^2) + \mathcal{N}(0, \sigma_{\text{y}}^2) \tag{7}$$

The updated position with GPS error is then given by:

$$\hat{x}_{t+1} = x_{t+1} + n_x \tag{8}$$

$$\hat{y}_{t+1} = y_{t+1} + n_y \tag{9}$$

where $n_x \sim \mathcal{N}(0, \sigma_{\text{x}}^2)$ and $n_y \sim \mathcal{N}(0, \sigma_{\text{y}}^2)$ represent the GPS errors in the x and y coordinates, respectively.

3.4 Parametrization

The simulator's behavior is controlled by parameters provided in a JSON file. These parameters include the frequency of updates, the number of agents, state transition probabilities, speed and direction ranges, and herd behavior settings. This flexibility allows the simulation to be adapted to a wide range of scenarios and research needs.

4 Inference Design and Implementation

The inference component plays a critical role in identifying anomalous behavior within the herd in real-time. This system operates asynchronously, continuously processing data for each agent in the simulation. At its core, the inference system employs an autoencoder—a neural network model specifically trained to learn the normal behavior patterns of the agents, such as Grazing, Foraging, and Resting. The autoencoder is key to detecting deviations from these normal behaviors, thus flagging potential anomalies.

4.1 Inference Design

Input Vector Definition: For each agent in the herd, a rolling window of observations is maintained with a predefined length m. This window stores the agent's longitude (lon), latitude (lat), and corresponding timestamps, encapsulating recent movement history. The input vector to the autoencoder, $\mathbf{x}_i$, is a concatenation of these observations, structured as $\mathbf{x}_i \in \mathbb{R}^{2m\times 1}$. The first m elements of this vector represent the longitude values, and the next m elements represent the latitude values. Formally, the input vector for agent i at time t is given by:

$$\mathbf{x}_i(t) = \left[\text{lon}_{t-(m-1)\cdot\Delta t} \quad \cdots \quad \text{lon}_t, \quad \text{lat}_{t-(m-1)\cdot\Delta t} \quad \cdots \quad \text{lat}_t\right]^\top$$

In this expression, $\text{lon}_{t-(m-1)\cdot\Delta t}$ represents the longitude of the agent at the time $t-(m-1)\cdot\Delta t$, which is the earliest observation in the rolling window, while lon_t represents the longitude at the current time t. Similarly, $\text{lat}_{t-(m-1)\cdot\Delta t}$ denotes the latitude at the time $t-(m-1)\cdot\Delta t$, and lat_t denotes the latitude at the current time t. The time interval between consecutive observations is denoted by Δt.

Data Handling: As new data points are received, they are appended to the rolling window for the respective agent. To maintain a fixed window size, if the number of data points exceeds m, the oldest data points are discarded.

4.2 Model Training and Deployment

Autoencoder Structure: The autoencoder model is composed of an encoder that reduces the dimensionality of the input vector $\mathbf{x}_i(t)$ to a latent representation and a decoder that attempts to reconstruct the original input from this latent space. The reconstructed output is denoted as $\hat{\mathbf{x}}_i(t)$. The autoencoder is trained on data representing simulated normal herd behaviors, allowing it to capture the underlying patterns of these behaviors.

Anomaly Detection: The detection of anomalies is based on the reconstruction error, calculated as the Mean Squared Error (MSE) between the input vector $\mathbf{x}_i(t)$ and its reconstruction $\hat{\mathbf{x}}_i(t)$. The MSE for agent i is computed as follows:

$$\text{MSE}_i(t) = \frac{1}{2m}\sum_{k=1}^{2m}\left(x_{i,k}(t) - \hat{x}_{i,k}(t)\right)^2$$

where $x_{i,k}(t)$ represents the k-th element of the input vector, and $\hat{x}_{i,k}(t)$ represents the corresponding reconstructed value. If the MSE exceeds a predefined threshold, the agent is flagged as exhibiting anomalous behavior, which could indicate issues such as illness, predator presence, or other environmental stressors.

4.3 Model Testing and Evaluation

To evaluate the effectiveness of the autoencoder in detecting anomalies, the model was tested by simulating different herd behaviors, including the Fleeing state, which introduces anomalous behavior. During the anomalous state simulation, the state of the agents was set to Fleeing, and the MSE was compared across normal and anomalous states, allowing us to assess how well the model could differentiate between normal states (Grazing, Foraging, Resting) and the anomalous Fleeing state in unsupervised manner where model does not see any anomalous data.

AUC Metric: The Area Under the Curve (AUC) metric was employed to further quantify the model's performance in distinguishing between normal and anomalous behaviors. The AUC is calculated by comparing the anomaly score (i.e., MSE) for each anomalous sample against every normal sample. For each comparison, if the anomaly score for the anomalous sample is greater than the score for the normal sample, a value of 1 is assigned; otherwise, a value of 0 is assigned. The AUC is then computed as the ratio of successful comparisons over the total number of comparisons, mathematically expressed as:

$$\text{AUC} = \frac{1}{|\mathcal{A}| \times |\mathcal{N}|}\sum_{a\in\mathcal{A}}\sum_{n\in\mathcal{N}}\mathbb{I}\left(\text{MSE}_a > \text{MSE}_n\right)$$

where $\mathcal{A}$ is the set of anomalous samples, $\mathcal{N}$ is the set of normal samples, and $\mathbb{I}(\cdot)$ is the indicator function, returning 1 if the condition is true and 0 otherwise.

5 Simulation and Results

5.1 System Deployment

The simulation system was deployed using two separate Docker containers: one for the simulator and another for the inference module. These components were connected to allow real-time data exchange. The simulation was initiated by posting the necessary parameters via Postman to the local network.

5.2 Simulation Parameters

The simulation modeled a herd of 50 agents, each interacting with 5 nearest neighbors. Key parameters include:

- **Weight Factor:** Movement was influenced by herd with a weight factor of 0.8, balancing random and herd behavior.
- **Behavioral States:** The agents could be in one of four states: Grazing, Resting, Foraging, or Fleeing. Initial probabilities were set to 50% for Grazing and Resting, and 0% for Foraging and Fleeing.
- **State Transition Matrix:** Reflecting realistic behavior changes, the matrix was designed to maintain stability in common states while allowing rare transitions, particularly into the Fleeing state:
  - `Grazing: [0.89995, 0.05, 0.05, 0.00005]`
  - `Resting: [0.05, 0.89995, 0.05, 0.00005]`
  - `Foraging: [0.05, 0.05, 0.89995, 0.00005]`
  - `Fleeing: [0.0003, 0.0003, 0.0003, 0.9991]`
- **State Parameters:** Each state's movement characteristics were chosen to reflect typical behavior, with Grazing and

Resting involving slow, random movement, and Fleeing involving fast, directed movement:

- **Grazing:** Speed [0.1, 0.5], Turning Angle [-180, 180]
- **Resting:** Speed [0, 0.5], Turning Angle [-180, 180]
- **Foraging:** Speed [0.5, 1.5], Turning Angle [0, 20]
- **Fleeing:** Speed [1.5, 3], Turning Angle [0, 5]

5.3 Model Training

The anomaly detection model used a fully connected autoencoder neural network. The autoencoder consisted of an encoder that reduced the input vector's dimensionality and a decoder that attempted to reconstruct the original input. The model was trained on normal herd behavior data, such as Grazing, Resting and Foraging, to learn the expected patterns of movement.

The training data was normalized using the mean and standard deviation from the dataset. The autoencoder was trained using the Mean Squared Error (MSE) loss function and the Adam optimizer, over 100 epochs. The dataset was split into 60% for training, 10% for validation, and 30% for testing. During training, the model parameters were saved when the validation loss reached its minimum, ensuring the best performance on unseen data.

5.4 Results

Figure 1 presents the distribution of Mean Squared Error (MSE) values for both normal and anomalous samples. As depicted in the figure, the MSE for normal samples is generally lower, indicating that the model can accurately reconstruct normal behavior patterns. Conversely, the MSE for anomalous samples is significantly higher, reflecting the model's difficulty in reconstructing behaviors that deviate from the learned normal patterns.

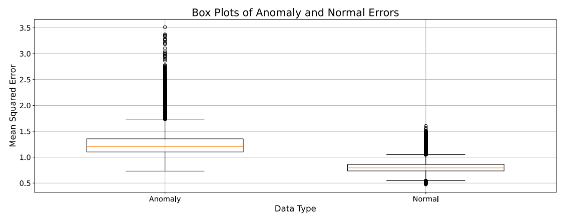

Figure 1: MSE plot for normal and anomalous behavior

To further illustrate the effectiveness of MSE as an indicator for anomaly detection, Figure 2 shows two examples: the left-hand side depicts an anomalous sample with an MSE of 1.40, while the right-hand side depicts a normal sample with an MSE of 0.74. These examples highlight how the model assigns higher MSE values to anomalous data, making MSE a reliable metric for identifying anomalies.

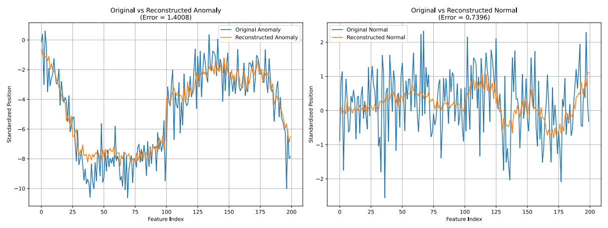

Figure 2: Reconstruction of normal and anomalous data. The left image shows an anomalous sample with MSE = 1.40, while the right image shows a normal sample with MSE = 0.74.

Additionally, the model's performance was evaluated using the Area Under the Curve (AUC) metric. The model achieved an AUC of 0.98, demonstrating its high accuracy in detecting anomalies.

6 Conclusion

In this work, we developed a comprehensive system for simulating animal herd behavior and detecting anomalies using a combination of a custom-built simulator and a deep learning-based inferencer. The simulator effectively models the movement of animals under various states, including normal behaviors such as Grazing, Resting and Foraging, as well as anomalous behaviors like Fleeing. The inferencer, powered by an autoencoder, processes the simulated data in real-time and successfully identifies deviations from expected behavior patterns through the use of Mean Squared Error (MSE) as an anomaly detection metric.

Our results demonstrate the effectiveness of the proposed system, with the model achieving an Area Under the Curve (AUC) of 0.98, indicating a high level of accuracy in distinguishing between normal and anomalous behaviors.

Further work will focus on deploying the model on a Kubernetes cluster, which will enable the simulation of multiple herds simultaneously and the inference of their behavior in a scalable manner. This deployment will likely require the integration of a database to store historical data and improve the system's ability to detect anomalies by leveraging past observations. The planned enhancements will extend the system's applicability to more complex scenarios, providing valuable insights for both wildlife management and agricultural practices.

References

[1] F. Cagnacci, L. Boitani, R. A. Powell, and M. S. Boyce. 2010. Animal ecology meets GPS-based radiotelemetry: A perfect storm of opportunities and challenges. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1550):2157–2162.

[2] P. Laube. 2011. How fast is a cow? Cross-Scale Analysis of Movement Data. *Transactions in GIS*, 15(3):401–418.

[3] C. W. Reynolds. 1987. Flocks, Herds, and Schools: A Distributed Behavioral Model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34.

[4] J. Nanajkar, S. Thorat, G. Borse, U. Naravade, and V. Pratale. 2024. A survey of Anomaly Detection in IoT Networks. *International Journal of Creative Research Thoughts (IJCRT)*, 12(2):e382–e389.