Chaos Detection and Mitigation in Swarm of Drones using Machine Learning Techniques and Chaotic Attractors

Emmanuel NEBE[1], Mistura Laide SANNI[2], Rasheed Ayodeji ADETONA[3]
Bodunde Odunola AKINYEMI[4], Sururah Apinke BELLO[5], Ganiyu Adesola ADEROUNMU[6]
Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria[1, 2, 4, 5, 6]
Department of Mathematics, Obafemi Awolowo University, Ile-Ife, Nigeria[3]

***Abstract*—Most existing identification and tackling of chaos in swarm drone missions focus on single drone scenarios. There is a need to assess the status of a system with multiple drones, hence, this research presents an on-the-fly chaotic behavior detection model for large numbers of flying drones using machine learning techniques. A succession of three Artificial Intelligence knowledge discovery procedures, Logistic Regression (LR), Convolutional Neural Network (CNN), Gaussian Mixture Models (GMMs) and Expectation–Maximization (EM) were employed to reduce the dimension of the actual data of the swarm of drone's flight and classify it as non-chaotic and chaotic. A one-dimensional, multi-layer perceptive, deep neural network-based classification system was also used to collect the related characteristics and distinguish between chaotic and non-chaotic conditions. The Rössler system was then employed to deal with such chaotic conditions. Validation of the proposed chaotic detection and mitigation technique was performed using real-world flight test data, demonstrating its viability for real-time implementation. The results demonstrated that swarm mobility horizon-based monitoring is a viable solution for real-time monitoring of a system's chaos with a significantly reduced commotion effect. The proposed technique has been tested to improve the performance of fully autonomous drone swarm flights.**

***Keywords*—Chaos detection; swarm of drones; machine learning; autoencoder; Rössler system**

I. INTRODUCTION

A swarm of drones is a group of two or more drones that exchange data and work as a single cooperative unit to accomplish a specific mission objective. Drone coordination has been extensively researched in the fields of surveillance systems, precision agriculture, transportation, disaster management, and entertainment [1] [2].

A swarm of small aircraft allows for a larger mission area, more flexible mission capabilities, greater resilience against single-point failure, and lower costs. Swarm drone research has covered a wide range of topics, including collision avoidance [3], [4], [5], [6], mission-level planning and control to enable high-level autonomy [7], [8], the human operator's communication with a swarm of drones [9], [10], ad hoc backbone network customized for a swarm operation, and the construction of small-scale airborne vehicles [11], [12]. Previously, technology-oriented research focused on how to improve performance and capacity [13], [14], [15], but more recent research has focused on making such swarm systems more safe, secure, and reliable to operate [7], [16], [17]. Studies in the development of new coordination algorithms that combine biological processes are based on self-regulation [18], [19], and environmental adaptability to allow a swarm of drones to work with greater sophistication, reliability, scalability, and flexibility [20].

A number of practical issues that might disrupt the successful completion of the swarm mission could arise during the operation of a swarm of drones. For example, the energy consumption limitation of drones, which limits drones in their ability to handle long-term flight, may cause one or more drones in the swarm to experience failure, necessitating the development of an intelligent, efficient power failure mechanism. Alternatives for aerial path loss should also be considered while maintaining drone security and safety. When considering a multi-drone environment, where a small or large group should operate together or act in the same aerial environment, various flight problems and obstacles, in addition to the aforementioned chaotic difficulties, may arise, such as weather conditions and signal loss. These chaotic issues, however, impose a number of constraints on the use of swarms of drones and must be addressed in real-time.

The ability to monitor or manage the disorder or instability of the drones in a swarm, and take predictive and critical steps as needed for the safe and dependable operation of swarm drones, is referred to as chaos handling. Many factors can have an impact on the system's health, such as issues with the drone system's actuators and sensors, communication connection flaws, and possibly hostile cyberattacks. Determining such causes requires a thorough understanding of the system, mission, and surroundings. Therefore, detecting chaos in swarm system behavior is the first step toward managing the system's health. This is crucial even when the source or type of chaos is unknown.

Despite extensive research into traditional model-based approaches for fault handling, identification, and isolation, particularly in aircraft safety systems, there are no known existing solutions for dealing with chaos among drones [21] [22]. Failure Detection, Identification, and Recovery (FDIR) has recently been extended to swarms of drone systems in terms of operations [23], and resistance against cyberattacks

[23], [24], [25], but there is still a need to address chaos among the swarm of drones, regardless of whether the causes of such chaos are known or unknown. In [26], evaluation of drones that use onboard sensor data for Failure Detection, Identification, and Recovery (FDIR) using a cooperative virtual sensor system for the design and experimental verification of techniques and procedures for handling chaos in swarm drone systems was done. Furthermore, compared to large aircraft, drone swarms are a new market entry; thus, failure mechanisms and chaos are not widely implemented or in use. Therefore, statistical methods which do not rely solely on the physical-based model of the drones may be a more viable option for detecting and mitigating chaos in the current swarm of drone systems. Thus, in this study, an attempt was made to develop a machine learning-based model for chaos detection in a swarm of drones; and a mitigation technique to deal with such chaotic conditions is also proposed.

The remaining sections of the paper are organized as follows: Section II discusses related works, while Section III describes the machine learning methods used in detecting and mitigating chaos in real-time swarm dataset. Section IV describes the chaos detection modeling process, while Section V presents the results of the model evaluations to demonstrate the effectiveness of the chaos detection and mitigation strategies for swarms of drones. Finally, the conclusion was presented in Section VI.

II. Related Work

Several significant studies have been conducted in order to apply data-driven machine-learning algorithms for detecting faults and anomalies in aerial vehicles. Some researchers combined chaotic dynamics with powerful swarm-based algorithms used in mobility models such as Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO) [27], [28], [29]. The Lorenz and Rössler attractors are time-discretized, and the three-dimensional chaotic maps are addressed via a three-dimensional solvable chaos graph built from general chaos solutions.

A three-dimensional path planning for Unmanned Aerial Vehicles (UAVs) based on chaos particle swarm optimization, which addresses the shortcomings of particle swarm optimization (PSO) was proposed in [30]. However, the solution quickly falls into a local optimum and gradually converges with poor precision in a motion phase. The concept of the Chaos Optimization (CO) algorithm was incorporated into the PSO algorithm through in-depth analysis based on the conventional update operations on the velocity and location of the mobile nodes in the swarm. As a result, track preparation searches are eliminated and rapidity followed by convergence precision is enhanced.

In [31], a basic two-dimensional solvable chaos map was used to mathematically analyze chaotic modeling and simulation on a dynamic coordinate. Also in [32], a chaotic-based approach was used to maintain coordinated flight formation of swarm unmanned aerial vehicles at a low input cost. A study in [33] focuses on a discrete dynamic map (logistic map) to generate a chaotic sequence of bits [33]. The bits are then translated into locations that allow the robots to construct a deterministic route plan. Meanwhile the R-UAVs are fractional three-dimensional when using the Qi system [34]. As a result, a three-dimensional chaotic dynamic solution should be used to model the mobility model of swarm UAVs.

The 3D chaotic-based-approach is used in ASIMUT to implement dynamic system mobility models for UAVs in an unpredictable regime. This mobility model is supplemented by a hybrid mobility model called Chaotic Ant Colony Optimization for Coverage (CACOC), which combines the ACO with the system's three ordinary differential equations. The mobility model is data-centric in a multi-level swarm perception networked on the multi-layer FANET architecture [35], [36]. Also, a collision avoidance technique was incorporated into a predictive mobility model, based on the assumption that all UAVs were flown at different altitudes to avoid collisions [37]. However, this may not be realistic in some UAV swarm applications.

For real-time detection and monitoring of aviation system abnormalities, a Multivariant Gaussian Mixture Model (MGMM) was proposed [38]. Also proposed was a Recurring Neural Network (RNN) method for events and trends which can reduce the security margins of a system using a dataset from a Flight Data Recorder (FDR) [39].

A K-nearest neighbor (KNN) methodology was introduced in [40] to identify the reasons and factors for drone failures and potential deteriorations in drone performance on the ground in order to assess the causes of failure and potential deterioration in drone performance during flight. An actual flight dataset was used in [41] to validate the developed Anomaly Detection (AD) model, which shows if there is any abnormality in the swarm drone flight. For the generation model, the AD model created a training model using a Deep Neural Network.

Most of the work done has focused only on the health management of a single drone; no methodological approach for the reliable detection of chaos in swarm flights has been proposed, with the goal of overcoming the aforementioned swarm drone behavior constraint. For example, an end to end fault analysis framework for a single micro aerial vehicle that only considers anomalies with obstacle detections [54]. Some studies used artificial neural network for sensor-based fault detection [55]. As a result, this current study proposes a machine learning-based, data-driven methodology for detecting chaotic anomalies in swarm flights. The proposed method aimed to address both the lack of marked recorded information in swarm flights and the disparity between non-chaotic and pathological data. This study investigates the use of moving average-based monitoring with a limited time frame to reduce noise in continuous monitoring while also allowing for responsive chaos detection.

In general, supervised learning approaches outperform unsupervised learning methods in classification problems. However, in the case of chaos detection, a relatively new type of self-supervised knowledge extraction that outperforms the fully supervised technique has been reported. As a result, this method can be used to detect chaos in in-flight data. A highly sophisticated self-supervised framework that outperforms all other unsupervised methods, and demonstrated that the

completely controlled approach ranks better and outperforms the other methods was developed in [42].

This paper backs up this claim by demonstrating that supervised learning is still a viable core framework when a significant amount of labeled data is available, and an adequate labeling process can be produced. This study makes three significant contributions: It (a) proposes a systematic process for data-driven chaos identification in swarm flights, (b) generates a real-time solution to such chaos, and (c) validates the method using real flight test data.

III. Methodology

The following methods were used in this study to detect chaos in a swarm of drone flights: First, a set of unsupervised learning methods were employed: An Autoencoder (AE) was used to reconstruct and reduce the time series dimensionality of flight data, and then an Expectation–Maximization (EM) clustering by Gaussian Mixture Model (GMM) separated the flight data time series dimension into four categories: true chaos, unsure chaos, doubtful normal, and truly normal. Then, a Deep Neural Network was trained to extract features and detect chaos, which is a 1D-CNN concatenation of a single, multi-layer logistic regression perceptron neural network [43].

To properly identify chaos and handle it, three decisions must be made: (a) detecting chaos symptoms in the swarm, (b) identifying which drone is in chaos, and (c) providing a solution to such drones in a state of chaos. The chaotic detection technique makes these decisions by observing the kinematic characteristics of the drones, such as a drone's location and velocity. This chaos detection scheme is located at the ground station or controller, which monitors the health and the flying status of all swarm vehicles; thus, it transfers drone data to the ground station, where it is processed by the chaos detection scheme to show flight normality.

In this study, the swarm system employed a real-time Kinematic Velocity (KV) GPS-based precision navigation method as described by [44]. The dataset used as learning data for recognizing chaotic behavior during swarm mission execution was sourced from a series of swarm drone flight tests conducted by the Korean Aerospace Research Institute, in which up to 30 quadcopter drones were deployed [41].

A total of 50 tests were conducted, with individual and multiple groups of test drones. The output data from each drone trajectory consists of 248 parameters presented as time series, some of which include numerous observations of various parameters. As critical characteristics in detecting chaos in motion, three sites were chosen: the drone location and set point values, three-vehicle speed components, and vehicle status. The KV-GPS data, in which the accuracy was validated in [44] was used to calculate the location coordinates (x_t, y_t, z_t) and velocity components in three dimensions for the drones. The following three parameters in Equation 1 are also considered chaotic, as they can be associated with errors between intended and actual behavior during drone movement coming from mission control.

$$x = x_t - x_s$$

$$y = y_t - y_s$$

$$z = z_t - z_s \quad (1)$$

Also, inertial navigation system readings labeled GA from accelerometers and gyro sensors may also cause mechanical faults in drone systems because they can cause unexpected acceleration and angular rate behavior. Thus, the drone status indicator is used to verify the data consistency and calibration. Since this research aimed to develop a chaos detection and mitigation method that is independent of drone type and features, datasets are not tagged with drone identifiers as a result. Fig. 1 depicts an illustrative view of the swarm system configurations.

It should be noted that these topological indicators are not always the whole set of characteristics required to identify all potential chaos in swarm flights. These are, nevertheless, crucial indicators for sensing chaos produced by specific kinds of errors and failures. As a result, this research detects and identifies chaotic behavior using topological indicators and produces a solution to such a problem using a chaotic attractor. Also, the flight trajectory dataset is only partially labeled. Some of the incorrect events discovered during the flight test are classified as chaos. Table I lists the features of the indicators utilized in this paper.

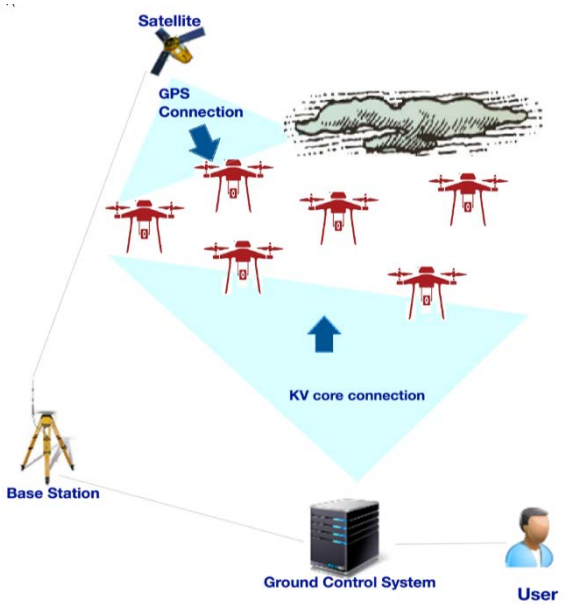

Fig. 1. Configuration of the Swarm of Drone System.

TABLE I. Parameters Extracted for Drone Swarm Flight Analysis

Drone Parts	Parameter Description	Attribute
Drone state	Hovering Navigating	Discrete
Trajectory	Position (x_t, y_t, z_t)	Continuous
GA	Gyro (Rate and integrals of drone body frame) Accelerometer (Axis (x,y,z) values and integrals of accelerometer)	Continuous
KV-GPS	Inertial Velocity Vectors and Positions	Continuous

A. Machine Learning Algorithms

This section discusses four machine learning methods that were used in this study to develop a data-driven chaos detection strategy. The first two are unsupervised learning algorithms that work with unlabeled data, whereas the last two are supervised classification algorithms that work with labeled data. The details are as follows:

1) Autoencoder and Chaos Detection (AECD): Autoencoder-based chaos detection (AECD) is a semi-supervised learning-based chaos identification technique. As part of this study, an autoencoder was used to compress the raw flight input data. Autoencoders (AE) are neural networks that are statistically based on a given probability distribution [45]. The autoencoder consists of two sub-models: encoder and decoder. The encoder compresses the input, while the decoder attempts to reconstruct the input from the encoder's compressed form. After training, the encoder model is saved, whereas the decoder is destroyed. For machine learning training, the encoder is then used as a data preparation tool, to extract features from raw data.

A single-layered neural network has an encoder and a decoder, as shown in Equations (2) and (3). This is the nonlinear transformation function of the autoencoder. Equation 2 depicts how an affine mapping uses nonlinearity to convert an input vector d to a hidden vector h. In Equation 3, the decoder uses the same transformation as the encoder to rebuild the cached representation h back to the initial input space. As shown in Equation 4, the reconstruction error is defined as the difference between the original input vector d and the reconstruction z. The reconstruction error is minimized via the autoencoder.

$$h = \sigma(W_{dh}d + b_{dh}) \quad (2)$$

$$z = \sigma(W_{hd}d + b_{hd}) \quad (3)$$

$$Reconstruction\ error\ =/\ d - z\ / \quad (4)$$

where, w and b are the weights and biases of the neural network.

By adding noise to the original input vector d, the autoencoders use a noisy input vector d' as the input vector. In other words, a noisy input d' was fed into the autoencoder in order to recreate the original input d. In this way, the autoencoder is protected from white noise in the data and collects only significant patterns in swarm flight data [46]. The reconstruction error is then calculated by measuring the difference between the final output and the noisy input reconstruction. The reconstruction error is then used to determine the chaos score.

Data points with a high degree of reconstruction are described as chaos. To train the autoencoder, only data with normal occurrences is utilized. The autoencoder will successfully reconstruct normal data after training but will fail to reconstruct chaotic data that the autoencoder has never seen. Algorithm 1 depicts the chaos detection technique based on autoencoder reconstruction errors.

Algorithm 1 Autoencoder and Chaos Detection Algorithm

INPUT: Normal dataset $D\ d^{(1)},\ldots,\ d^{(n)}$, Chaos dataset $d'^{(i)}$ $i=1,\ldots,n$, threshold α
OUTPUT: *reconstruction error* $||d - d'||$
θ,φ ← *Initialize parameter*
repeat
$E = \sum_{i=1}^{n} || d^{(i)} - g_\theta(f_\varphi(d^{(i)})) ||$ Calculate the total amount of reconstruction error, where g_θ and f_φ are the autoencoder's multilayered neural networks.
θ,φ ← update parameters using Stochastic Gradient Descent
until *parameters* θ,φ *convergence*
then
θ,φ ←Using the normal dataset *D*, train the autoencoder
for *all values i*=1 to *n* **do**
reconstruction error(i) = $|| d^{(i)} - g_\theta(f_\varphi(d^{(i)}))||$
if α < *the reconstruction error(i)* **then**
$d^{(i)}$ is in chaos
else
$d^{(i)}$ is not in chaos
end if
end for

2) Clustering using Gaussian Mixture Models (GMMs) and Expectation–Maximization (EM): The flight data points were assumed to be distributed randomly in a Gaussian manner. Gaussian Mixture Models (GMMs) were used to simulate the data. Each flight data cluster's Gaussian parameters were determined using two parameters derived from an optimization technique called Expectation-Maximization (EM): the mean and standard deviation. Therefore, drone clusters can have any elliptical shape since the standard deviations in the x and y axes are obtained. As a result, each Gaussian distribution has exactly one cluster. The hidden variables were used to find the Maximum Likelihood Estimators (MLEs). Since the AECD model contains latent variables, maximum likelihood estimates of the model were sought using the EM method. As a result, the log-likelihood is as shown in Equation 5.

$$\log\big(P\ (D|\theta)\big) = \log(\textstyle\sum_z P((D, Z|\theta)) \quad (5)$$

Where, D represent the total number of observable variables

Z represent the total number of latent variables, marginalized from the joint distribution.

Assuming datasets D and Z, were selected at the same time, the entire dataset is referred to as $\{D, Z\}$ and the incomplete dataset is referred to as D. From the original dataset, Z, is unknown, but the posterior $P(Z|D,\Theta)$ contains the information about Z. Therefore, the log-likelihood expectation was analyzed by using the M-step process to evaluate the posterior probabilities. The expectation of the entire data log-likelihood was maximized to get a new estimated parameter via the E-step process. The current value of the parameters θ^0 was used in the E-step to obtain the posterior distribution of the latent variables, which is provided by $P\ (Z\ |D,\ \theta^0)$. This expectation, represented by $Q\ (\theta,\ \theta^0)$ is shown in Equation 6. The new parameter θ' is then determined in the M-step by maximizing Q as shown in Equation 7.

$$Q(\theta,\theta^0) = E_{(Z|D,\theta^0)}[\log((D,Z|\theta))] = P\sum_Z P\ (Z\ |D,\theta 0)\log(P\ (D,Z\ |\theta\ ))\ . \quad (6)$$

$$\theta' = argmax_\theta Q(\theta,\theta^0) \quad (7)$$

where

$\theta = \mu_1,\cdots,\mu_k,\sigma_1,\cdots\sigma_k,\pi_1,\cdots,\pi_k$ are the unknown parameters used in deriving the MLE Gaussian Mixture Model. The relevant quantities for GMM are then derived to form the complete likelihood as shown in Equations 8 and 9.

$$\log(P(D,Z|\mu,\sigma,\pi)) = \sum_{i=1}^{n}\sum_{k=1}^{m}\ )I(Z_i = k)(\log(\pi_k) + \log(N(D_i|\mu_k,\sigma_k)) \quad (8)$$

$$E_{(Z|D)}\log\big(P(D,Z|\mu,\sigma,\pi)\big) = \sum_{i=1}^{n}\sum_{k=1}^{m} Y_Z(k)\ (log\ \pi_k + log(N(D_i|\mu_k,\sigma_k))\big)\ . \quad (9)$$

The posterior probabilities $\gamma Z_i(k)$ was evaluated from the current values of μ_k and σ_k. Since $E_{Z|D}[I(Z_i=k)] = P(Z_i=k|D)$, marginal probability distribution can thus be replaced with the posterior probability $\gamma_{Zi}(k)$.

Hence, Expectation-Maximization is derived as follows: first, initial values were chosen for the parameters μ σ, and π, these parameters were employed in the E-step to evaluate the posterior probability $\gamma_{Zi}(k)$. With fixed $\gamma_{Zi}(k)$, the expected complete log-likelihood is maximized as shown in Equation 9 with respect to μ_k, σ_k, and π_k.

3) Convolutional Neural Network (CNN): Convolutional Neural Network concept was used to extract features from drones' trajectories. It was noted that CNN has made great strides in handling two-dimensional image data and thus for one-dimensional time series data, locality/dispersion may also be used [43]. The CNN network structure employed in this study for cataloguing and classification comprises a stack of one-dimensional convolutional and max-pooling layers, including an additional global-pooling or flattening layer connected to the max-pooling layers. A Multi-layer Perceptron (MLP) is linked to the output layer of the one-dimensional CNN and then stochastic gradient descent is used to maximize the weights of both the MLP and CNN simultaneously [47]. A nonlinear activation function is used to transform a perception y into a linear combination that uses its weighted inputs to generate a single output which is dependent on many real-valued inputs (from the GMM and EM outputs). The perceptron is expressed in Equation 10.

$$\boldsymbol{y} = \varphi(\sum_{i=1}^{n}\omega_i D_i + b) = \varphi(w^T x + b) \quad (10)$$

4) Logistic Regression (LR): Logistic Regression (LR) was used for two-class classification in this study. LR is one of the easiest and most widely used methods of machine learning that can be used for two-class classification. It is a statistical method for predicting binary classes with intrinsically dichotomous values or target variables like one and zero. Equation 11 can be used to express the logistic regression hypothesis.

$$y = (1|1 + \exp(-(\beta_0 + \beta_1 D_1 + \beta_2 D_2 +,\cdots,\beta_m D_m) \quad (11)$$

where, Dm denotes the explanatory features of the flight data and; βm denotes the related coefficients that will be optimized via a learning process.

Because y has a value between 0 and 1, the result can be interpreted as the likelihood of fitting to class 1. As shown in Equation 12, the most common loss function for optimizing coefficients is maximization of output probability.

$$loss(z,y) = -\sum_{i=1}^{n}(z_i\ log\ y_i\ +\ (1\ -\ z_i)log(1\ -\ y_i)) \quad (12)$$

where, *n* represents the amount of data points and;

z represents the desired outcome.

Equation 12 illustrates how the loss function can also be viewed as the cross-entropy between z and y. Instead of using a neural network, this sigmoidal activation function is commonly used for categorization. In this scenario, the network is trained to minimize the cross-entropy loss.

B. Rössler System (RS)

The Rössler System (RS) was used to handle any drone's trajectory in the swarm that remains in a state of chaos. As shown in Equation 13, the numerical solution for a drone's three-dimensional trajectories is given as a fractional order of the Rössler System [34]. The data for the a drone's three coordinates in the (*x,y,z*) axis is given by a time step t.

$$\begin{cases} D_t x = y - z \\ D_t y = x + ay \\ D_t z = b + z(x - c) \end{cases} \quad (13)$$

The RS is composed of three Ordinary Differential Equations (ODEs) with only one non-linear term, with constant values *a = 0.2, b = 0.2, and c = 9.0;* thus, each ODE can represent a dimension of a drone's flight trajectory [36], [48]. The synchronization of the RS with machine learning algorithm for trajectories in chaos was achieved after eliminating the leading drone's trajectory flight data observations. In order to remove the transient states, a numerical solution was constructed for this system using the fourth-order Runge-Kutta approach and record time points with each time step being *t = 0.1*.

IV. Chaos Detection and Mitigation Model Description

The details of the model description are as follows:

A. Model Architecture

Fig. 2 depicts the general architecture of the model for detecting and mitigating chaos in a swarm of drones. The flight test data used for this study were collected in advance but when swarm flight data is provided in real time, it is not automatically labeled. This is normal in chaos detection instances since the data can only be plainly labeled in the presence of a functioning chaos detector or if human investigators have thoroughly examined the data. As a result, the data was initially grouped and labeled into various relevant

groups using a clustering method. Four criteria are addressed in this study's labeling method because of the ambiguity in chaos judgments: Normal (N), chaos (C), tentative normal (N˜), and tentative chaos (C˜). The last two categories were included because there are times when it's unclear whether the data provided is sufficient to determine whether a flight's behavior is chaotic or normal. After the labeled data has been secured, a binary classifier based on CNN was trained and validated on a set of training data to understand the critical characteristics in identifying chaos in swarm flight data. The trained model may then be utilized for both post-flight analysis and real-time chaos monitoring as a chaos detection technique. Meanwhile, as a chaos handling technique, the Rössler system was used to generate new flight paths for drones in the swarm that are in a chaotic state.

B. Data Preparation

A total of 73,749-time series dataset were generated as a result of the data preprocessing analysis, which includes data cleaning, integration, and transformation.

1) Data cleaning: The data cleaning process includes the completion of missing values, the filtering of excessive noise, the elimination of outlines, and the resolution of solution discrepancies. The difference between the real and reference positions of the drones in the swarm, as well as the actual velocity, are used as input data. To correspond to the timestamps of the two distinct data sources, a linear interpolation approach was used. In order to get a different value to real data at the same time, a linear interpolation of reference data with low noise relative to actual signals was performed. Excessively high values, possibly due to the training dataset, remove noise aberrations.

2) Data integration: Data from multiple drones were combined to create a dataset containing drone identification numbers. The drone identification numbers are not explicitly used in the learning process, but they are necessary for evaluating the performance of the learned model. One label, a one-time stamp, and six kinematic variables are also included in the input data.

3) Data normalization: The goal of data normalization is to keep the array of values for the specified parameters within a certain range. The well-known standardizing approach was used in this study. As shown in Equation 14, the standardization approach normalized each data sequence by computing the mean ($D_{mean)}$ and standard deviation (D_{std}) values.

$$D_{normalized} = (D - D_{mean}/D_{std} \quad (14)$$

C. Data Labelling and Clustering

Sensors mounted on drones produces multidimensional data which is characterized by a complex correlation, making it difficult to define the system's status. An autoencoder and chaos detection (AECD) algorithm was used to reduce the dimensionality of the data by identifying the primary correlation pattern between the variables. AECD is based on the assumption that the majority of system states can be adequately described by the characteristics of a few key components, and it has proven to be a successful feature extraction technique in a variety of situations [49]. The encoded kinematic information is represented by a six-dimensional AECD algorithm to produce the rate of cumulative dispersion to encode the component axes of a swarm of drones.

The clustering approach was used to facilitate labeling of unlabeled data in its raw form. Labeling data aids supervised learning processes in chaos classification by grouping data based on some kind of similarity or distance measure, making it easier for human specialists to classify the data.

This study employed clustering on the smaller space created by the AECD algorithm using Gaussian Mixture Models (GMMs) and Expectation-Maximization (EM). The time-series data for AECD was based on six variables for drone trajectories in the (x, y, z) axes and the velocity vectors (vx, vy, vz), with one drone label presented after AECD. Because the principal component axis was chosen to have a cumulative dispersion rate of 90% or higher, the number of dimensions reduced as a result of AECD may vary depending on the available flight data.

Specifically, clustering was performed on data that had been dimension-reduced, and the EM technique was used to maximize the number of clusters [50]. Based on the clustered findings, the state of each drone in the swarm was classified as Normal (N) or Chaos (C). The associated variables were obtained by categorizing them into two groups.

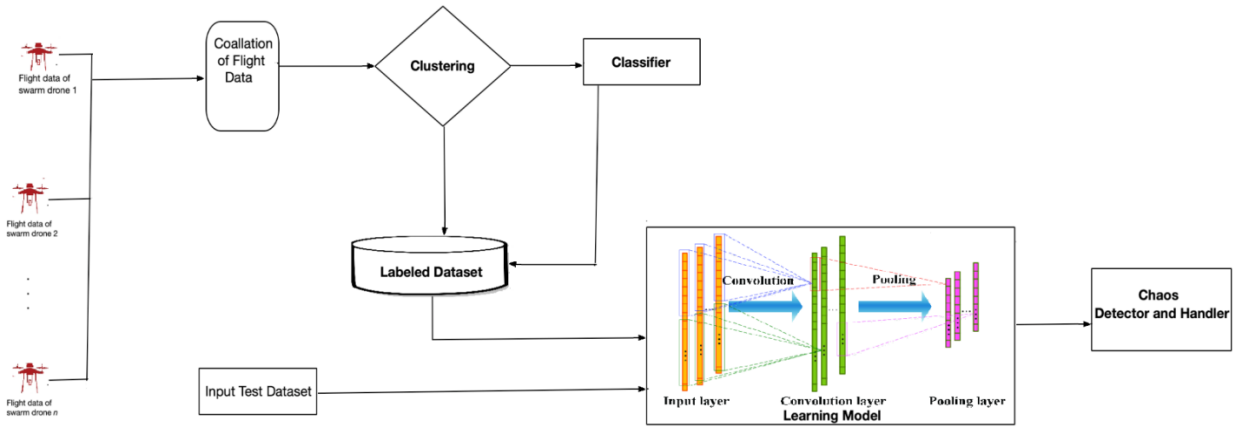

Fig. 2. Chaos Detection and Mitigation Model Architecture for Swarm of Drones.

It first examined the outcome of a swarm of drones in completing a specific scenario as a mission using KV–GPS data and trajectory location data. It finally checked the sensor signals to detect any chaos in the drone using the GA data. AECD, GMMs, and EM clustering are the three techniques that label unlabeled datasets into two categories: Normal (N) with a data sample of 23,501 and Chaotic (C) with a data sample of 20,266.

D. Optimization and Classification of Data Sample

The data sample was classified and optimized in three stages. First, the data sample was standardized, then the data was classified using a CNN classifier, and finally, the classified data was optimized using Stochastic Gradient Descent with the AdaDelta optimization technique.

1) Standardization of data samples: After categorizing the normal and chaotic states, the drone-label information was deleted, leaving a total of six variables. To standardize the range of input variable values, a data standardization approach similar to pre-processing for clustering was used. In the labeled data, the Logistic regression technique was used for the binary categorized variables. The two-potential dependent-variable values of 0 and 1 represent the "normal" and "abnormal" results. Binary logistic models were used to assess the likelihood of a binary answer based on one or more predictor (or independent) functions.

2) CNN Classifier: To normalize the time sequence data of the kinematic variables specified in this study, the Gaussian Mixture Models and Expectation-Maximization technique were used. The data was first transferred through a one-dimensional Convolutional Neural Network (CNN) with six hidden layers, which was then linked to a dense multilevel perceptron with sigmoid activation at the output end. The sigmoid activation function result was compared to the target label value by reversing this error and its cross-entropy error (Equation 5), and was used to learn the overall neural network. A Stochastic Gradient Descent technique combined with minibatch was used for learning. Table II shows the details about the neural network layers and the parameters used to train the network, while Table III shows more details about the design, it was created by first building a sufficiently large network and then controlling it with batch normalization [51].

3) Stochastic and mini batch gradient descent in adadelta optimization: Most neural network methods are designed to improve accuracy; they work best when each class studies the same (or comparable) amount of data. However, when the number of normal and chaotic drones is significantly different, a varied binary classification, such as in defect classification or chaotic detection, does not produce excellent results [52]. This method generates additional samples in order to achieve a one-to-one relationship between normal and abnormal data for batches in order to optimize AdaDelta to compensate for the imbalance used in the network's training [53].

The AdaDelta optimization method is a stochastic optimization methodology for Stochastic Gradient Descent with a per-dimension learning rate method. It aims to slow down the monotonously fast rate of learning. Rather than gathering all past squared gradients, AdaDelta limits the window of accumulated past gradients to a specific size. Only one example was analyzed at a time to perform a single step in Stochastic Gradient Descent (SGD). Using the SGD, the following steps were taken for each epoch:

a) Take the sample data,

b) Insert it to Convolutional Neural Network,

c) Find its gradient,

d) Use the computed gradient in step 3 for weight updates,

e) Repeat steps (a)–(d) for all of the items in the data sample.

Because only one example was considered at a time, the cost will fluctuate rather than decrease over the training examples. Considering, *mN* and *mC* denoting the number of data points labeled as normal (N) and chaotic (C), respectively. If the sampling required to produce a minibatch in the stores' gradient is carried out consistently, the predicted data quantity ratio for the two classes is *mN/mC*.

TABLE II. ONE-DIMENSIONAL CONVOLUTIONAL NEURAL NETWORK PARAMETER

Layer	Input Variables (*n*, timestamp, 5)	Output (*n*, 160, 5)	Operation Standard
1	(n, 160, 5)	(*n*, 80, 10)	One-dimensional Convolutional, Rectified Liner Unit (ReLU), one-dimensional batch normalization
2	(n, 80, 10)	(*n*, 40, 20)	One-dimensional Convolutional, ReLU, one-dimensional batch normalization
3	(*n*, 40, 20)	(*n*, 20, 40)	One-dimensional Convolutional, ReLU, one-dimensional batch normalization
4	(*n*, 20, 40)	(*n*, 10, 80)	One-dimensional Convolutional, ReLU, one-dimensional batch normalization
5	(*n*, 10, 80)	(*n*, 5, 160)	One-dimensional Convolutional, ReLU, one-dimensional batch normalization
6	(*n*, 5, 160)	(*n*, 5, 128)	One-dimensional Convolutional, ReLU, one-dimensional batch normalization
7	(*n*, 5, 160)	(*n*, 160)	Global pooling average
8	(*n*, 160)	(*n*, 64)	Batch standardization, Dense, ReLU
9	(*n*, 80)	(*n*, 2)	Dense, Sigmoid

TABLE III. LEARNING MODEL HYPER-PARAMETERS

Size of Batch	Length of Batch in Seconds	Number of Epoch	Learning rate
128	160	50	0.00005

E. Handling Drones in Chaos

The optimized classified chaos based on the trajectories of the drones in the swarm were addressed in real time using the Poincare map from the Rössler system, with the data sample classified and labeled as chaos, mC. The Poincaré map was created by charting the function's value each time it crosses a specified plane in a specific direction.

Plotting the x, y, and z coordinates every time it passes through the x=0 plane, where x changes from negative to positive. As a result, the Poincare map converts the solutions of the three Ordinary Differential Equations of the Rössler system into the coordinates that best remove such drones from the chaotic state.

V. RESULT AND DISCUSSION

The detailed results are as follows:

A. Clustering and AECD

The flight data from a swarm of six drones was evaluated using the Autoencoder and Chaos Detection algorithms, as well as Gaussian Mixture Models and Expectation-Minimization for clustering. Fig. 3 and Fig. 4 depict the clustering results for a swarm of drones on a specific illustrative flight test day using AECD and Gaussian Mixture Models. The first and second autoencoder components generated the distribution of data points in the reduced space; Fig. 3 is based on KV-GPS data, while Fig. 4 is based on GA data. The dimension was reduced to auto-decoded axes with a cumulative dispersion rate of at least 93% using the AECD process, and the clusters were discovered using Gaussian Mixture Models clustering with Expectation-Maximization. Fig. 4 depicts the distribution of data points in the reduced space formed by the autoencoder encoder and decoder components. A drone's data is disseminated in a very different way than data from other drones. As a result, it's reasonable to assume that data from the fourth drone in the swarm (Drn 4) will contain the chaotic time series. However, it is unclear how the clustering result is related to chaos, given that all data from a potentially problematic drone is unlikely to belong to a single cluster. The scatter plot for each cluster is shown in Fig. 5. This depicts the swarm's data point cluster. The plot aids in identifying the points of the root causes of chaos in a swarm, as well as the dependability of such points in relation to the rest of the swarm. Fig. 6 shows the percentage of data from each drone that belongs to a specific cluster. This depicts the effectiveness of such clusters of data points in relation to each drone in the swarm in order to identify the chaotic drone.

The majority of flight data from all drones falls into Clusters 4 and 9, but the distributions for the swarm's fourth drone (Drn 4) and the other drones differ. Drn 4 differs from the others in the ratio of data belonging to Clusters 4 and 9, with significantly more data belonging to Cluster 4 than Cluster 9 when compared to the other drones in the swarm. Another intriguing discovery is that data from Drn 4, a potentially chaotic drone, does not belong in Clusters 1, 2, 3, or 8. As a result, while the clustering does not indicate which drone may have exhibited chaotic behavior, the distribution of data among the clusters may indicate chaos in the flight data. Clusters 5, 6, and 7 may indicate chaos flight data, whereas Clusters 1, 2, 3, 8, and 9 indicate normal data. For the remainder of the clustering method, data from each drone in Clusters 4-6 was labeled "C," while data from Clusters 1-4, 8, and 9 was labeled "N." In cases of indistinct data, such as Cluster 3, and questionable data, such as the first drone (Drn 1) in Cluster 6, a human expert may examine the flight data to use as labeled data.

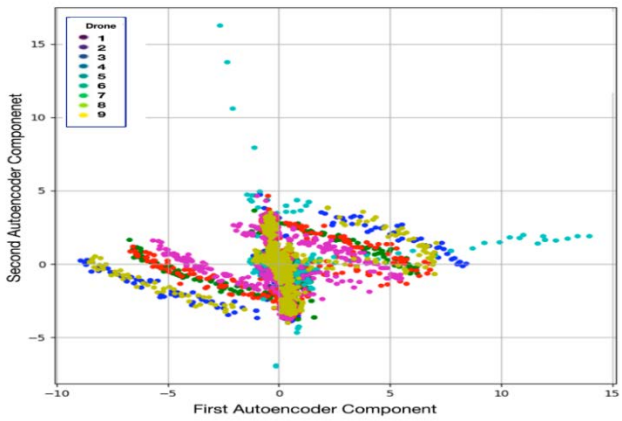

Fig. 3. Autoencoder KV-GPS Set Points Data Results.

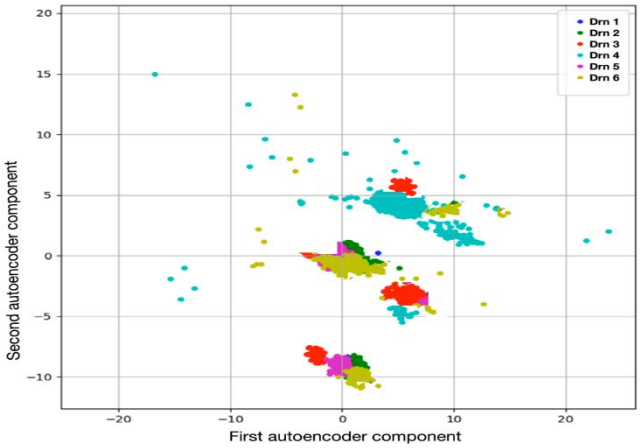

Fig. 4. Autoencoder GA Data Results.

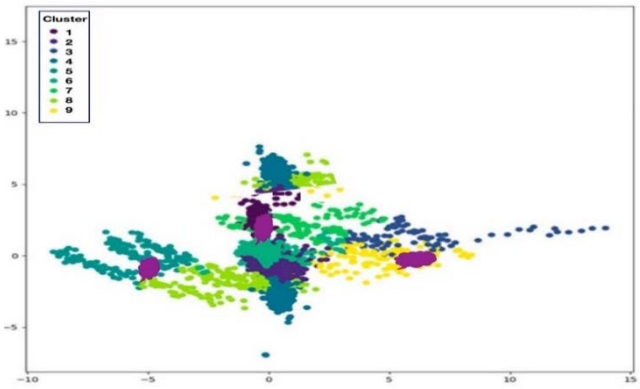

Fig. 5. Cluster Scatter Plot.

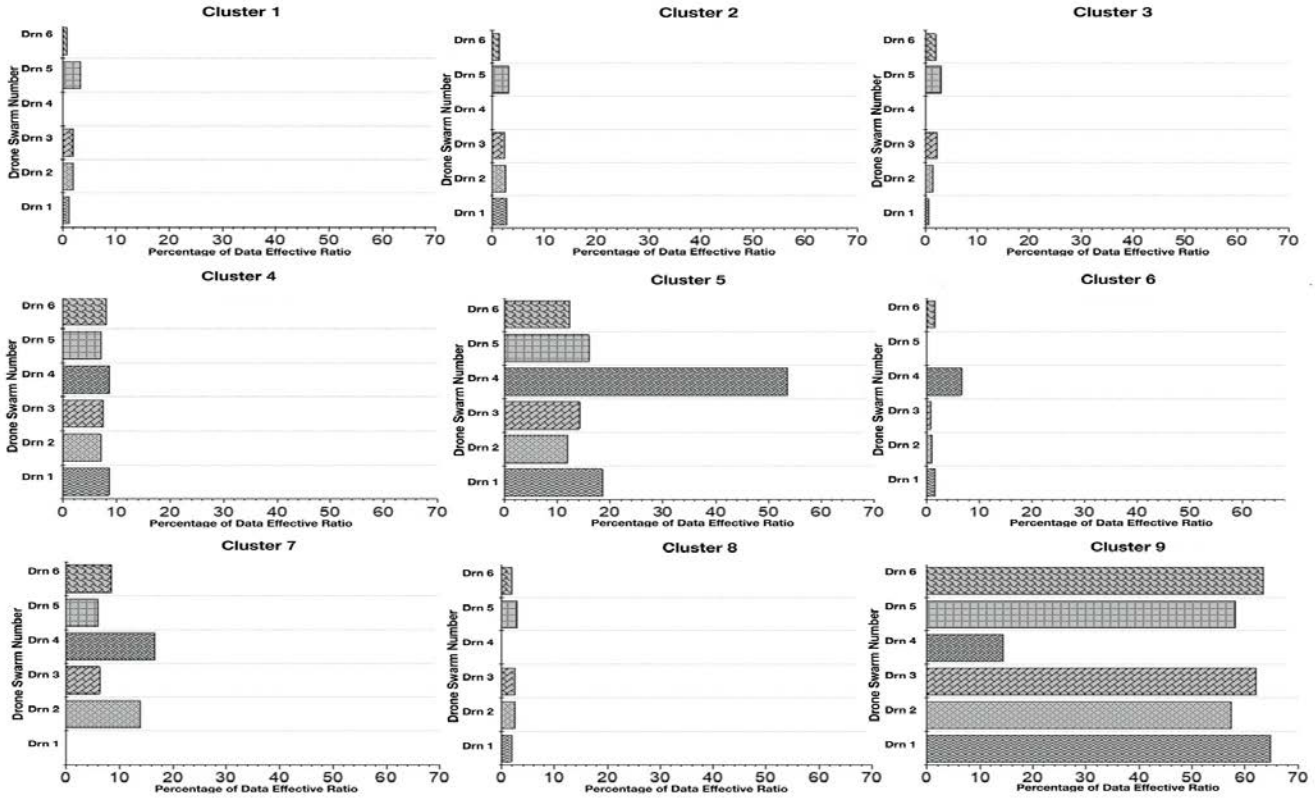

Fig. 6. Clustering Results for Swarm Flight Data using Gaussian Mixture Models.

B. *Classification Results*

The developed Convolutional Neural Network (CNN) classifier was trained until its cross-entropy and loss value converged on the learning parameters listed in Table II. The data from the test flight obtained from the swarm of drones was used as the training set of size 43,767; 70% of this dataset was used for training the neural network and 30% for model validation.

The dataset contains all of the drone's flight data, which was used to generate a test set with a size of 8,753. To validate the applicability of the sampling method, the test accuracy was equated with the variable rate of imbalance, resulting in the chaos data ratio versus the non-chaotic data in the initial dataset. If the recommended sampling is not used, chaotic swarm flight data is used in the learning process with the rate of imbalance relative to non-chaotic data. The chaos swarm flight dataset was replicated by the factor of the rate of inverse imbalance when samplings are used. The classification data accuracy was compared by the accurate classification out of all cases, which is dependent on the use of the sampling scheme to detect the differential imbalance of the original swarm flight data set. Fig. 7 compares the chaos findings of the case for the original rate of imbalance. This is the scenario when the swarm's sixth drone (Drn 6) exhibits chaotic flying behavior; only the sampling result clearly identifies chaos in the Drn 6

flight data. It is clear that the suggested sampling strategy significantly improves classification accuracy, achieving zero classification error. It should also be noted that when the sampling technique is not used, the classification accuracy is approximately 50%. One thing to keep in mind is that even with a higher imbalance rate of 0.5, there was no improvement as shown in Fig. 8.

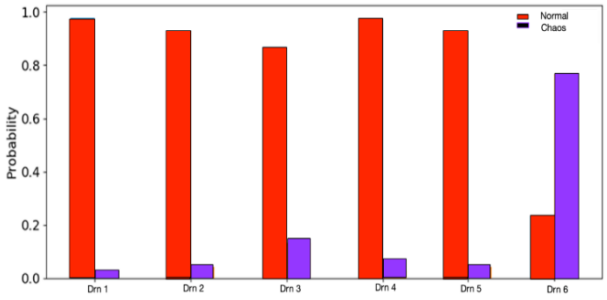

Fig. 7. Drone Imbalance Percentage (with Sampling).

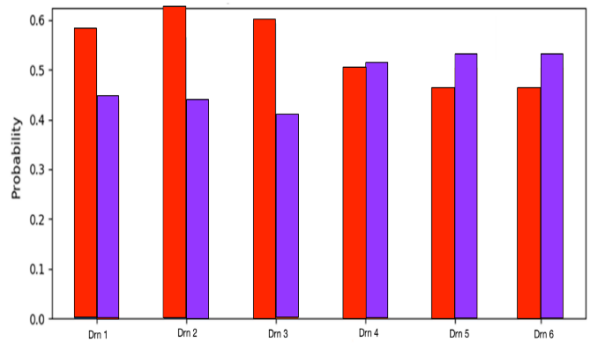

Fig. 8. Drone Imbalance Percentage (without Sampling).

C. Discussion

The proposed chaos detection methods were tested to see if the trained neural network classifier was useful for time series entry using real flight test data. If the input time series did not belong to the training set, the results for a sample drone flight test data and probability were computed [43]. The swarm flight test data was used, with clustering findings shown in Fig. 5 and Fig. 6. When the time series of the entry was constantly fed into the network, the neural network's output value corresponds to the sixth drone (Drn 6). The first output comes in response to the Kinematic variables [44] in the first 16 seconds segment, as the Convolutional Neural Network (CNN) takes a length of 160 input at a data rate of 10 Hz. The neural network calculates the output in real-time in response to the most recent 160-times segment points. The possibility of being normal has been determined to fluctuate over time until it reaches zero. When the average was achieved over the entire duration of the swarm flight, the average probability value of the six drones flown in that swarm flight test was compared. There's a much higher chance that this troublesome drone will be in a state of chaos than usual. According to Drn 6, the time history of the film input variables revealed that the drone does not respond effectively to changes in the x- and z-direction in the 90s and 75s, respectively; thus, the chaos detection schema revealed that the system cannot be normal at 85 seconds from the original data point, based on the behavior. Drn 6 crashed during the flight test, according to the original flight log. As a result of the application of the Rössler system's algorithm, the drone was sent on alternate routes assuming the chaos is in the trajectory flight data. This prevents drone 6 (Drn 6) from crashing in the test flight log. In terms of online chaos monitoring, the results showed that there were two extreme approaches to monitoring. Constant monitoring of the non-chaotic or chaotic probability allows for a high frequency assessment of system chaos. This method enables effective responsiveness in chaos detection because it constantly delivers updated information about the system's normalcy.

The output signal, on the other hand, was noticeably noisy during the transient period, as shown in Fig. 8. As shown, the average probability over the entire time duration provides a cooperative judgment on the system's chaos. Despite the fact that the decision frequency may be too low, this method may be the least noisy susceptible method. The non-chaotic/chaotic probabilities are averaged over a set period of time, and the dataset was then updated with new values.

The adjacent averaging window's time frame may overlap. The average non-chaotic probability was computed over 30 seconds and updated every 20 seconds, implying that one-third of the data is overlapped between time frames. The chaotic probability of 70 is around 40% in the first time segment window, increases to around 55% in the second time frame, and finally exceeds 90% in the third time window. This enables the chaos to be observed at a relatively high frequency while avoiding data noise.

VI. Conclusion

This study proposed a machine learning-based chaos detection and a mitigation technique for the flight paths of a swarm of drones. The proposed techniques consist of three major steps: a labeling phase that uses lower-dimensional features to label unlabeled data, a binary/dual classification step that employs a 1-D CNN with a cross-entropy-based loss function, and a mitigation step that employs the Rössler system to generate new non-chaotic trajectories for drones in chaos. The deep neural network was trained using real flight test data. In this paper, Swarm mobility horizon-based monitoring was demonstrated to be a viable solution for real-time monitoring of a system's chaos with significantly reduced commotion effect. This technique can be used to mitigate the effects of drone crashes and failures in a fully autonomous swarm of drones.

A future focus will be the integration of network intrusion detection, monitoring, and mitigation into the overall health-management architecture of swarm drones. This could result in a more reliable fully autonomous swarm of drones by mitigating the effects of network intrusions, which can cause chaos and anomaly in a swarm of drones.

Acknowledgment

This Research was funded by the TETFund Research Fund and Africa Centre of Excellence OAK-Park, Obafemi Awolowo University, Ile-Ife, Nigeria.

References

[1] A. Otto, N. Agatz, J. Campbell, B. Golden, E. Pesch, “Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey”. Networks, vol. 72, no. 4, pp. 411-458, 2018. doi: 10.1002/net.21818.

[2] Y. Li, C. Liu, “Applications of multirotor drone technologies in construction management”. International Journal of Construction Management, vol. 19, no. 5, pp. 401-412, 2019. doi: 10.1080/15623599.2018.1452101.

[3] K. Loayza, P. Lucas, E. Peláez, “A centralized control of movements using a collision avoidance algorithm for a swarm of autonomous agents”. In proceedings of the 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM), pp. 1-6, 2017. doi: 10.1109/ETCM.2017.8247496.

[4] J. N. Yasin, S. A. Mohamed, M. H. Haghbayan, J. Heikkonen, H. Tenhunen, J. Plosila, “Navigation of autonomous swarm of drones using translational coordinates”. In proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems, pp. 353-362, 2020, doi: 10.1007/978-3-030-49778-1_28.

[5] G. Ahmed, T. Sheltami, A. Mahmoud, A. Yasar, “IoD swarms collision avoidance via improved particle swarm optimization”. Transportation Research Part A: Policy and Practice, vol. 142, pp. 260-278, 2020. doi: 10.1016/j.tra.2020.09.005.

[6] R. Ourari, K. Cui, H. Koeppl, “Decentralized Swarm Collision Avoidance for Quadrotors via End-to-End Reinforcement Learning”. arXiv preprint, arXiv:2104.14912 , 2021.

[7] A. Majd, M. Loni, G. Sahebi, M. Daneshtalab, “Improving motion safety and efficiency of intelligent autonomous swarm of drones”. Drones, vol. 4, no. 3, pp. 48, 2020. doi: 10.3390/drones4030048.

[8] A. Atyabi, Z. S. Mahmoud, S. Nefti-Meziani, “Current advancements on autonomous mission planning and management systems: An AUV and UAV perspective”. Annual Reviews in Control, vol. 46, pp. 196-215, 2018. doi: 10.1016/j.arcontrol.2018.07.002.

[9] J. Cleland-Huang, A. Agrawal, “Human-drone interactions with semi-autonomous cohorts of collaborating drones”. arXiv preprint , arXiv:2010.04101, 2020.

[10] J. R. Cauchard, M. Khamis, J. Garcia, M. Kljun, A. M. Brock, “Toward a roadmap for human-drone interaction”. Interactions, vol. 28, no. 2, pp. 76-81, 2021. doi: 10.1145/3447889.

[11] M. A. Khan, A. Safi, I. M. Qureshi, I. U. Khan, Flying ad-hoc networks (FANETs): A review of communication architectures, and routing protocols. In 2017 First International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT), pp. 1-9, 2017, doi: 10.1109/INTELLECT.2017.8277614.

[12] Z. Chu, F. Hu, E. S. Bentley, S. Kumar, “ Model and simulations of multipath bridge routing for inter-swarm UAV communications in EMANE/CORE”, International Journal of Modelling and Simulation, 2021. doi: 10.1080/02286203.2021.1931789.

[13] A. A. Akintola, G. A. Aderounmu, L. A. Akanbi, M.O Adigun, “Modeling and performance Analysis of Dynamic Random Early Detection (DRED) Gateway for Congestion Avoidance”. Journal of Issues in Informing Science and Information Technology, vol. 2, pp. 623-636, 2005, doi: 10.28945/2920.

[14] A. A. Akintola, G. A. Aderounmu, A. U. Osakwe, M.O. Adigun, “Performance Modelling of an Enhanced Optimistic Locking Architecture for Concurrency Control in a Distributed Database”. Journal of Research and Practice in Information Technology, vol. 37, no. 4, pp. 365-380, 2005.

[15] O. A. Oluwatope, D. Iyanda, G. A. Aderounmu, E. R. Adagunodo, “Computational Modelling of Collaborative Resources Sharing in Grid System”. The Journal of Computer Science and Its Application, vol. 19, no. 1, pp. 74-83, 2012.

[16] Z. Yuan, J. Jin, L. Sun, K. W. Chin, G. M. Muntean, “Ultra-reliable IoT communications with UAVs: A swarm use case”. IEEE Communications Magazine, vol. 56, no. 12, pp. 90-96, 2018. doi: 10.1109/MCOM.2018.1800161

[17] S. A. Bello, G. A. Aderounmu, M. L. Sanni, K. A. AbdulSalam. S.I. Eludiora., A. D. Akinde, “Estimation of Optimal Frequency and Antennae Length between Resource Interface and Mobile Terminal in Wireless ATM”, Ife Journal of Technology, vol. 19, no. 2, pp. 55-58, 2010.

[18] S.U. Otor, B.O. Akinyemi, T.A. Aladesanmi, G.A. Aderounmu, B.H. Kamagaté , “An Adaptive Bio-Inspired Optimisation Model Based on the Foraging Behaviour of a Social Spider”. Cogent Engineering, vol. 6, no. 1, 2019. doi: 10.1080/23311916.2019.1588681.

[19] S.U. Otor, B.O. Akinyemi, T.A. Aladesanmi, G.A. Aderounmu, B.H. Kamagaté, “An Improved Bio-inspired based Intrusion Detection Model for a Cyberspace”. Cogent Engineering, vol. 8, no. 1, 2021. doi: 10.1080/23311916.2020.1859667.

[20] P. Petráček, V. Walter, T. Báča, M. Saska, “Bio-inspired compact swarms of unmanned aerial vehicles without communication and external localization”. Bioinspiration & Biomimetics, vol. 16, no. 2, 2020.

[21] Y. Albayram, T. Jensen, M. M. H. Khan, R. Buck, E. Coman, “Investigating the Effect of System Reliability, Risk, and Role on Users' Emotions and Attitudes toward a Safety-Critical Drone System”. International Journal of Human–Computer Interaction, vol. 35, no. 9, pp.761-772, 2019. doi: 10.1080/10447318.2018.1491665.

[22] V. Kkarchenko, “Big data and internet of things for safety critical applications: challenges, methodology and industry cases”. International Journal on Information Technologies and Security, vol.10, no.4, 2018.

[23] B. Hu, P. J. Seiler, “Certification analysis for a model-based UAV fault detection system”. In proceedings of the AIAA Guidance, Navigation, and Control Conference, pp. 0610, 2014. doi:10.2514/6.2014-0610.

[24] E. Ordoukhanian, A. M. Madni, “System Trade-offs in Multi-UAV Networks”. In proceedings of the AIAA SPACE 2015 Conference and Exposition, pp. 4542, 2015. doi: 10.2514/6.2015-4542.

[25] L. Negash, S. H. Kim, H. L. Choi, “Distributed observes for cyberattack detection and isolation in formation-flying unmanned aerial vehicles”. Journal of Aerospace Information Systems, vol. 14, no. 10, pp. 551-565, 2017. doi: 10.2514/1.I010531.

[26] A. Suarez, G. Heredia, A. Ollero, “Cooperative sensor fault recovery in multi-UAV systems”. In proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1188-1193, 2016. doi: 10.1109/ICRA.2016.7487249.

[27] M. C. Boutalbi, M. A. Riahla, A. Ahriche. "Enhanced UAVs Mobility Models for Surveillance and Intruders Detection Missions." *Arabian* Journal for Science and Engineering, pp. 1-17, 2022. Doi: 10.1007/s13369-021-06541-3.

[28] R. A. Saeed, M. Omri, S. Abdel-Khalek, E.S. Ali, M. F. Alotaibi. "Optimal path planning for drones based on swarm intelligence algorithm." Neural Computing and Applications vol. 34, no. 12, pp. 10133-10155, 2022. Doi: 10.1007/s00521-022-06998-9.

[29] W. He, X.Q. Wenjian, L. Lifang. "A novel hybrid particle swarm optimization for multi-UAV cooperate path planning." Applied Intelligence, vol. 51, no. 10, pp. 7350-7364, 2021. Doi: 10.1007/s10489-020-02082-8.

[30] Z. Cheng, Y. X. Tang, Y. L. Liu, “3-D Path Planning for UAV Based on Chaos Particle Swarm Optimization”. Applied Mechanics and Materials, vol. 232, no. 1, pp. 625-630, 2012. doi: 10.4028/www.scientific.net/AMM.232.625.

[31] S. Kawamoto, “2-D and 3-D solvable chaos maps “.Chaotic Modeling and Simulation (CMSIM), vol. 1, pp. 107-118, 2017.

[32] W. Zhu, H. Duan, “Chaotic biogeography-based optimization approach to receding horizon control for multiple UAVs formation flight”. IFAC-Papers OnLine, vol.48, no.5, pp.35-40, 2015. doi: 10.1016/j.ifacol.2015.06.460.

[33] C. K. Volos, I. M. Kyprianidis, I. N. Stouboulos, “A chaotic path planning generator for autonomous mobile robots”. Robotics and Autonomous Systems, vol. 60, no.4, pp. 651-656. 2012. doi: 10.1016/j.robot.2012.01.001.

[34] H. Wang, S. He, K. Sun, “Complex dynamics of the fractional-order rössler system and its tracking synchronization control”. Complexity, 2018. doi: 10.1155/2018/4019749.

[35] L. Ouvry, G. Masson, F. Hameau, B. G. Gaillard, B. Caillat, “A CMOS duty-cycled coherent RF front-end IC for IR-UWB systems”. In proceedings of the 2015 IEEE International Conference on Ubiquitous

Wireless Broadband (ICUWB), pp. 1-5, 2015. doi: 10.1109/ICUWB.2015.7324396.

[36] M. Rosalie, E. Kieffer, M. R. Brust, G. Danoy, P. Bouvry, “Bayesian optimisation to select Rössler system parameters used in Chaotic Ant Colony Optimisation for Coverage”. Journal of computational science, vol. 41, no.2, 2020. doi: 10.1016/j.jocs.2019.101047.

[37] J. Dentler, M. Rosalie, G. Danoy, P. Bouvry, S. Kannan, M. Olivares-Mendez, H. Voos, “Collision avoidance effects on the mobility of a UAV swarm using chaotic ant colony with model predictive control”. Journal of intelligent and robotic systems, vol. 93, pp. 227-243, 2019. doi: 10.1007/s10846-018-0822-8.

[38] G. Li, A. Rai, H. Lee, A. Chattopadhyay, “Operational anomaly detection in flight data using a multivariate gaussian mixture model”. In proceedings of the 10th Annual Conference of the Prognostics and Health Management Society, 2018. doi: 10.36001/phmconf.2018.v10i1.474.

[39] A. Nanduri, L. Sherry, “Anomaly detection in aircraft data using Recurrent Neural Networks (RNN)”. In proceedings of the 2016 Integrated Communications Navigation and Surveillance (ICNS), pp. 5C2-1, 2016. doi: 10.1109/ICNSURV.2016.7486356.

[40] A. Manukyan, M. A. Olivares-Mendez, H. Voos, M. Geist, “Real time degradation identification of UAV using machine learning techniques”. In proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1223-1230, 2017. doi: 10.1109/ICUAS.2017.7991445.

[41] H. Ahn, “Deep Learning based Anomaly Detection for a Vehicle in Swarm Drone System”. In proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 557-561, 2020. doi: 10.1109/ICUAS48674.2020.9213880.

[42] D. Hendrycks, M. Mantas, K. Saurav, S. Dawn. "Using self-supervised learning can improve model robustness and uncertainty." arXiv preprint: arXiv:1906.12340, 2019.

[43] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D. J. nman, “1D convolutionalal neural networks and applications: A survey”. Mechanical systems and signal processing, vol. 151, 2021. doi: 10.1016/j.ymssp.2020.107398.

[44] H. Moon, C. Kim, W. Lee, “A UAV based 3-D positioning framework for detecting locations of buried persons in collapsed disaster area”. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, vol. 41, 2016. doi: 10.5194/isprs-archives-XLI-B8-121-2016.

[45] J. An, S. Cho, “Variational autoencoder based anomaly detection using reconstruction probability”. Special Lecture on IE, vol. 2, no. 1, pp. 1-18, 2015.

[46] K. Kashima, “Nonlinear model reduction by deep autoencoder of noise response data”. In proceedings of the 2016 IEEE 55th conference on decision and control (CDC), pp. 5750-5755, 2016. doi: 10.1109/CDC.2016.7799153.

[47] F. Chollet, Deep learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek. MITP-Verlags GmbH & Co. KG., 2018).

[48] T. F. Weng, X. X. Cao, H. J. Yang, “Complex network perspective on modelling chaotic systems via machine learning”. Chinese Physics B, vol. 30, no. 6, 2021. doi: 10.1088/1674-1056/abd9b3.

[49] M. Ramamurthy, Y. H. Robinson, S. Vimal, A. Suresh, “Auto encoder based dimensionality reduction and classification using convolutional neural networks for hyperspectral images”. Microprocessors and Microsystems, vol. 79, 2020. doi: 10.1016/j.micpro.2020.103280.

[50] Z. Tirandaz, G. Akbarizadeh, H. Kaabi, “PolSAR image segmentation based on feature extraction and data compression using weighted neighborhood filter bank and hidden Markov random field-expectation maximization”. Measurement, vol. 153, 2020. doi: 10.1016/j.measurement.2019.107432.

[51] S. Loffe, C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In proceedings of the International conference on machine learning, pp. 448-456, 2015. doi: 10.5555/3045118.3045167.

[52] M. Buda, A. Maki, M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks”. Neural Networks, vol. 106, pp. 249-259, 2018. doi: 10.1016/j.neunet.2018.07.011.

[53] E. Yazan, M. F. Talu, “Comparison of the stochastic gradient descent based optimization techniques”. In proceedings of the 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), pp. 1-5, 2017. doi: 10.1109/IDAP.2017.8090299.

[54] Y. S. Hsiao, Z. Wan, T. Jia, R. Ghosal, A. Raychowdhury, D. Brooks, G. Y. Wei, V. J. Reddi "Mavfi: An end-to-end fault analysis framework with anomaly detection and recovery for micro aerial vehicles." arXiv preprint arXiv:2105.12882, 2021.

[55] M. Ullah, C. Zhao, H. Maqsood, M. U. Hassan, M. Humayun "Improved neural network-based sensor fault detection and estimation strategy for an autonomous aerial vehicle." International Journal of Intelligent Unmanned Systems, 2021. Doi: 10.1108/IJIUS-09-2021-0109.