AI Algorithm for Predicting and Optimizing Trajectory of UAV Swarm

MS(Research) Thesis

By

Amit Raj

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

June 2024

AI Algorithm for Predicting and Optimizing Trajectory of UAV Swarm

A THESIS

submitted to the

INDIAN INSTITUTE OF TECHNOLOGY INDORE

in partial fulfillment of the requirements for

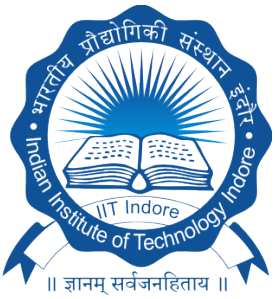
the award of the degree

of

MS(Research)

By

Amit Raj



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

June 2024

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **AI Algorithm for Predicting and Optimizing Trajectory of UAV Swarm** in the partial fulfillment of the requirements for the award of the degree of **MS (Research)** and submitted in the **Department of Computer Science and Engineering, Indian Institute of Technology Indore,** is an authentic record of my own work carried out during the time period from August 2022 to June 2024.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Date

Signature of the Student with

(Amit Raj)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of Thesis Supervisor with Date

(Prof. Kapil Ahuja)

Amit Raj has successfully given his MS(Research) Oral Examination held on _____

Signature of Chairperson, OEB Date:	Signature of External Examiner Date:	Signature of Thesis Supervisor Date:
Signature of PSPC Member #1 Date:	Signature of PSPC Member #2 Date:	Signature of Convener, DPGC Date:
Signature of Head of Discipline Date:		

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my heartfelt gratitude to a number of persons who in one or the other way contributed by making this time as learnable, enjoyable, and bearable. First, I would like to thank my supervisor **Prof. Kapil Ahuja**, a constant source of inspiration during my work. Without his constant guidance and research directions, this research work could not be completed. His continuous support and encouragement have motivated me to remain streamlined in my research work.

I am thankful to **Prof. Aruna Tiwari** and **Dr. Ayan Mondal**, my PSPC members for taking out some valuable time to evaluate my progress throughout the course. Their good comments and suggestions helped me to improve my work at various stages. I am also grateful to **DPGC** and **HOD** of Computer Science and Engineering for their help and support.

My sincere acknowledgment and respect to **Prof. Suhas Joshi** , Director, Indian Institute of Technology Indore for providing me the opportunity to explore my research capabilities at Indian Institute of Technology Indore.

I would like to express my heartfelt respect to my parents for the love, care, and support they have provided to me throughout my life.

Finally, I am thankful to all who directly or indirectly contributed, helped, and supported me.

Amit Raj

Abstract

This thesis explores the application of Artificial Intelligence (AI) techniques for generating the trajectories of fleets of Unmanned Aerial Vehicles (UAVs). The two main challenges addressed include accurately predicting the paths of UAVs and efficiently avoiding collisions between them, which we discuss in the two paragraphs below, respectively.

In all the previous studies that predicted the path, a Feedforward Neural Network (FFNN) with a single hidden layer and standard activation functions like Sigmoid, Tanh, and ReLU was used. These activation functions resulted in high errors (Mean Squared Error or MSE and Root MSE or RMSE) in the predicted path. In this work, we apply a non-standard set of activation functions, including Swish and Elliott, and also propose our new activation function, AdaptoSwelliGauss. AdaptoSwelliGauss is a sophisticated fusion of Swish and Elliott activations, seamlessly integrated with a scaled and shifted Gaussian component. This dynamic combination is specifically designed to excel in capturing the complexities of UAV trajectory prediction. The accuracy obtained with our new activation function is better by three to four orders of magnitude as compared to the standard activation functions.

UAV detection can be achieved in many ways. One approach involves changing the UAV trajectories, while another involves altering the starting time, a method also known as batching. Both techniques have their drawbacks. When applying a standard trajectory-changing technique to our set of UAVs, the algorithm ran into an infinite loop. On the other hand, when we used batching, the batch size was too large for our data. Therefore, in this thesis, we propose a novel integrated collision detection and avoidance by path and start time changes (ICDAPS) algorithm strategy that combines two complementary UAV collision avoidance techniques. This approach results in a finite number of trajectory changes in the first technique and a substantial reduction in batch size in the second.

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

AI Artificial Intelligence

UAV Unmanned Aerial Vehicle

FFNN Feedforward Neural Network

ICDAPS Integrated Collision Detection and Avoidance by Path and Start time changes

ReLU Rectified Linear Unit

CTGA Circular Arc Trajectory Geometric Method

WLOG Without Loss of Generality

MSE Mean Squared Error

Tanh Hyperbolic Tangent Function

Chapter 1

Introduction

UAVs have become increasingly popular in recent years due to their versatility and potential for a wide range of applications, from surveillance and monitoring to delivery and transportation. However, the safe and efficient operation of UAVs in complex environments remains a significant challenge, particularly when multiple UAVs are involved. A key issue is the need to optimize the trajectories of the UAVs to achieve various objectives, such as minimizing travel time, avoiding collisions, and maximizing coverage (1). Traditional methods for trajectory planning and control are often limited in their ability to handle the complexity and uncertainty of real-world scenarios and may not be scalable to large fleets of UAVs.

Prior research, exemplified by (2), (3), (4), (5), (6) and (7) has demonstrated the efficacy of leveraging non-linear optimization techniques. When quick trajectory changes are required, the optimization routine is too slow and not adaptive. AI techniques, particularly those based on machine learning and neural networks, have shown great promise in addressing these challenges. There are two kinds of popular AI techniques; and some prominent work in this area include (6), which has used Feedforward Neural Network (FFNN) with a single hidden layer, and (8), which has used deep networks.

Since deep learning methods don't provide the level of accuracy (discussed later in the thesis), we focus on simple FFNN with a single hidden layer. Most previous works in this domain have utilized standard activation functions such as Sigmoid, Hyperbolic Tangent (Tanh), and Rectified Linear Unit (ReLU), which don't predict paths with much accuracy (6), (7), (9), (10), (11). So we focus on improving this

aspect. Our contributions in this area are as below.

- We systematically apply both standard activation functions and application-oriented activation functions to the FFNN. The application-oriented activation functions used include Swish and Elliot, known for their resilience to noisy data, which is common in UAV path prediction.

- Additionally, we combine these activation functions with a Gaussian function to propose our new AdaptoSwelliGauss, which better captures the application's behavior.

- We compare our best application-oriented activation function, AdaptoSwelli-Gauss, with the best standard activation function, ReLU, and observe three to four orders magnitude reduction in error.

In the UAV context, detection of collisions between UAVs and their avoidance is of the utmost importance. Hence, we look at this aspect.

Collision detection between UAVs is most commonly done by building a small radius sphere around each UAVs and detecting whether those sphere intersect or not, we follow this approach. There are two ways to avoid collisions, one technique is by changing their trajectories. We follow the approach of (1), where a small perturbation is added to the one of the X,Y and Z component. This algorithm has a drawback. Any alteration in the trajectory of one UAV may inadvertently create collision candidates with other UAVs, leading to a challenging situation. Additionally, frequent manipulations in a UAV's trajectory can result in a convoluted flight path, compromising the overall efficiency of the UAV swarm.

Another complementary technique to avoid UAV collisions is to change their starting times. (12) and (13) propose such a popular approach. They employ a batching mechanism, creating groups of UAVs with non-colliding trajectories to facilitate safe flight. However, the creation of multiple batches introduces a time-consuming process, which delays the overall launch of the UAV swarm.

In this thesis, we introduce an advanced collision detection and avoidance algorithm, referred to as the integrated collision detection and avoidance by path and start time changes (ICDAPS) algorithm . Our contributions are as below.

- Here, we first improve the collision avoidance from (1) by introducing tracking array to avoid infinite alteration.
- Second, we integrate this avoidance algorithm with the batching mechanism, leading to our improved algorithm.
- This leads to a finite number of path adjustments and a reduction by half in the number of batches in which the number of UAVs are dispatched.

The remainder of the thesis is organized as follows: Chapter 2 reviews the literature, Chapter 3 describes our proposed algorithms and methodology, Chapter 4 presents the results, and Chapter 5 concludes the thesis and suggests directions for future work.

Chapter 2

Literature Review

In this chapter, we analyze the previous works on applying AI algorithms in optimizing UAV trajectories, and their collision detection and avoidance.

Firstly, we highlight the earlier studies on AI for UAVs, as summarized in Table 2.1. As evident, these papers address different kinds of problems. Papers (7), (11) and (6) are closest to our problem. The activation functions used by them are standard. It is important to note that the data used to train and test the neural network is different for each technique. When we apply standard activation functions to our data, we observe a high errors. However, when we use more sophisticated activation functions, including our new activation function, the error decreases by three to four orders of magnitude compared to the standard activation functions.

Second, we review the literature on collision detection and avoidance when there is a change in the path of the trajectory. These are summarized in Table 2.2. For detection, most of the techniques build a sphere around the UAV under consideration and obs erve which UAVs are coming close to it at a particular point in time. For example, we use the techniques of (14) as described in Table 2.2. For avoidance, most works change trajectory by either changing by velocity or adding force to it. We follow change in trajectory by (1).

Table 2.1: Summary on AI for UAVs.

Studies	Focus	AI Architecture	Activation Function	Order of MSE (any unit of distance)
(7)	Trajectory Modelling	FFNN	Tanh	Best of X,Y and Z: 10^{-3}
(11)	Predict UAV's Position	Recurrent Neural Network	Tanh	Best of X,Y and Z: 10^{-3}
(6)	Trajectory Generation	FFNN	Tanh	X: 10^{-2} Y: 10^{-2} Z: 10^{-2}
(10)	Flight Time Prediction	FFNN	Tanh	Flight time: 10^{-4}
(9)	Wind-induced Trajectory Deviation	Deep Neural Network (DNN12)	ReLU	X: 10^{-6} Y: 10^{-6} Z: 10^{-6}
(15)	UAV Control	FFNN	Sigmoid	X: 10^{-2} Y: 10^{-2} Z: 10^{-2}

Table 2.2: Collision avoidance for UAVs via path changes

Studies	Detection Technique	Avoidance Technique
(16)	They measure the distance between quadcopters and assess their velocities to identify potential collisions	Uses repulsive force field around each quadcopter to adjust their trajectories and prevent collision
(17)	They use current position and velocities between UAVs to identify potential collisions	Adjusts UAV trajectories to maintain safe distances
(18)	They use sphere around UAVs and Euclidean distance between them to identify collision	Uses coordinated trajectory adjustments among UAVs to ensure they change their flight paths
(14)	They use UAVs current position and velocities to identify possible collisions in real-time	Adjusts UAV velocities in real-time to maintain safe distance
(1)	They calculate future distance between them by using positions and velocities.	Adjusts UAV trajectories in real-time to maintain safe distances

Table 2.3: Collision avoidance for UAV via start time changes (Batching)

Studies	Detection Technique	Avoidance Technique
(12)	Line-based trajectory collision analysis	Changing take-off time by creating Batches
(12)	Position and line based collision detection analysis	Changing take-off time by creating Batches

Third we review literature on collision detection and avoidance based on start time change (batching). These are summarized in Table 2.3. Both papers here are from the same author. We use the detection technique discussed above and combine this avoidance technique of start time change with above avoidance technique.

and apply similar techniques. We adopt this with the collision avoidance approach discussed above.

Chapter 3

Methodology

One of the basic architectures is the FFNN, which works well for us. However, there are many advanced neural networks available that, in the very basic form, do not perform well. This aspect is discussed in Appendix A.

Our research employs a FFNN architecture, as illustrated in Figure 3.1, comprising an input layer, a hidden layer, and an output layer. The input layer of the FFNN is structured to incorporate the initial, intermediate, and final states of the UAV across all three spatial coordinates: X, Y, and Z. These states collectively form a multidimensional input vector that encapsulates the complete trajectory information of the UAV.

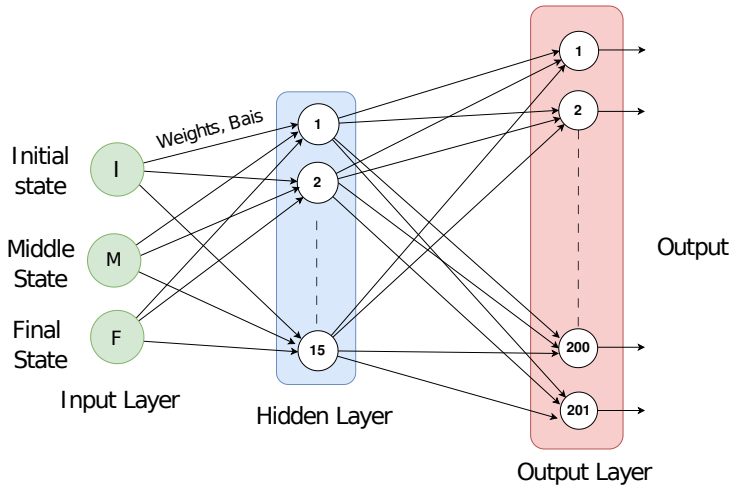


Figure 3.1: FFNN Architecture.

The hidden layer plays a pivotal role in capturing and representing complex patterns within the input data. Comprising 15 neurons, this layer employs various

activation functions to nonlinearly transform the input, facilitating the extraction of relevant features and patterns essential for accurate prediction. This is the layer where we experiment with different types of activation functions and propose a new activation function as well, which is discussed below.

Responsible for producing the final predictions, the output layer is meticulously designed with 201 output neurons which comprise of linear polynomials. The deliberate choice of linear activation functions in this layer aligns with the nature of regression tasks, where the objective is to predict continuous numerical values. We don't change these activation functions in this layer.

To effectively train our neural network architecture, we employ the Levenberg-Marquardt (LM) backpropagation algorithm from (19). This optimization technique blends the advantages of both gradient descent and the Gauss-Newton method and updates old network's weights w_{old} to new weights w_{new} by solving:

$$w_{\text{new}} = w_{\text{old}} - (J^T J + \lambda I)^{-1} J^T e \tag{3.1}$$

where J is the Jacobian matrix of the network's error function, e represents the error vector, λ is the damping parameter that controls the transition between the gradient descent and Gauss-Newton approaches, and I is the identity matrix. When λ is large, the update rule resembles gradient descent, while for smaller values of λ, it approaches the Gauss-Newton method.

The combination of a diverse set of activation functions in the hidden layer and the linear output layer forms a comprehensive framework for our research, allowing us to address the specific complexities and intricacies of our dataset. The rest of this chapter has three parts. In chapter 3.1, we describe an array of activation functions, including the newest ones. In chapter 3.2, we describe our novel AdaptoSwelliGauss activation function, in chapter 3.3, we discuss our ICDAPS technique.

3.1 Activation Functions

In a neural network, an activation function plays a crucial role by applying a mathematical operation to the weighted sum of inputs. This introduces non-linearity,

which is essential for the network to comprehend intricate patterns and connections within the data. Essentially, the activation function decides whether a neuron should fire or remain inactive, thereby impacting the flow of information throughout the network. In this sub-chapter, we delve into the various standard activation functions commonly employed in neural networks.

Here, x represents the input to each activation function.

1. **Sigmoid:** The Sigmoid function maps input values into a probability range between 0 and 1, which is ideal for binary classification outputs (20). The formula for this is

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{3.2}$$

Its major drawback is the vanishing gradient problem.

2. **Tanh:** Tanh extends the Sigmoid activation function shape, mapping inputs to a range between -1 and 1, which is zero-centered. It typically results in faster convergence than Sigmoid (21). The formula for this is

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{3.3}$$

Like Sigmoid, it suffers from vanishing gradients, affecting its utility in deep networks.

3. **ReLU:** ReLU addresses some of the critical issues of earlier activation functions as it is computationally cheaper and prevents vanishing gradient issues (22). It is given by

$$f(x) = \max(0, x). \tag{3.4}$$

Nonetheless, ReLU is susceptible to the "dying ReLU" problem.

4. **Leaky ReLU:** Leaky ReLU modifies ReLU to allow a small slope for negative input values. This modification ensures that all neurons have the opportunity to update during training, thereby avoiding the dying ReLU problem (23). It is given by

$$f(x) = \max(\alpha x, x). \tag{3.5}$$

where α is a small coefficient. This function requires careful tuning of α, adding complexity to the network's training process.

5. **Swish:** Swish is a self-gated activation function that blends input and Sigmoid output. It is smooth, non-monotonic, and also deals with the vanishing Gradient problem (24). It is given by

$$f(x) = x \cdot \sigma(\beta x), \tag{3.6}$$

where σ is the Sigmoid function and β is a trainable parameter. Like Leaky ReLU, Swish also requires careful tuning of β, which is computationally expensive (25).

6. **Maxout:** Maxout is a piecewise linear activation function. It selects the maximum value among the outputs of k linear functions, which allows it to capture more complex decision boundaries (26). It is given by

$$f(x) = \max(w_1 \cdot x + b_1, w_2 \cdot x + b_2, \ldots). \tag{3.7}$$

where w_i and b_i are the weights and biases associated with each of the k linear functions, respectively. To leverage multiple linear transformations, Maxout adds additional parameters, which lead to increased model complexity.

7. **Elliot:** Elliot is designed to provide a simpler and computationally efficient alternative to the Tanh activation function with the same property (27) and differentiable everywhere. It is given by

$$f(x) = \frac{x}{1 + |x|}. \tag{3.8}$$

3.2 Novel Activation Function

The UAV data contains a lot of noise due to the inherent jerky nature of the automatic control of UAVs (28). Path of UAVs are also non-linear (29). The activation function which behaves well under both these condition (noisy data as well as non-linearity) is the Swish (30). However, sensitivity of the Swish activation function for large value of input is high, hence we combined it with Elliot (31), which is also considered good for noisy data. Since Elliot does not have non-linear component, we multiply it with scaled and shifted Gaussian (32). Hence our new activation function has the following formula.

$$\text{AdaptoSwelliGauss}(x) = \begin{cases} \text{Swish}(x), & \text{if } x \leq \alpha \\ \text{Elliott}(x) \times \text{Scaled Shifted Gaussian}(x), & \text{otherwise} \end{cases} \tag{3.9}$$

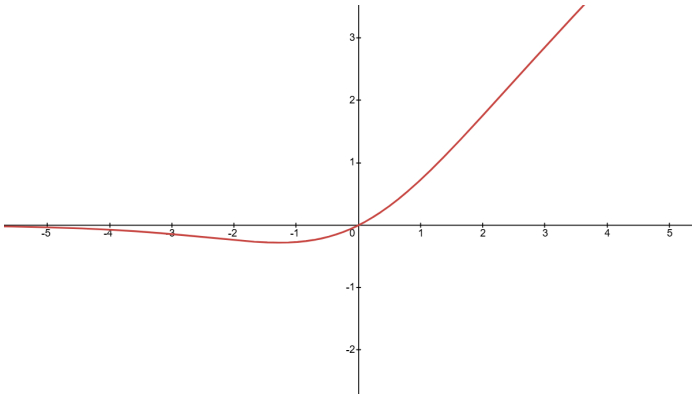
In this formulation,

- x represents the input to the activation function,
- Swish(x) is the output when the Swish activation is applied to input x,
- Elliott(x) is the output when the Elliott activation is applied to input x,
- Scaled Shifted Gaussian(x) is the output of the function

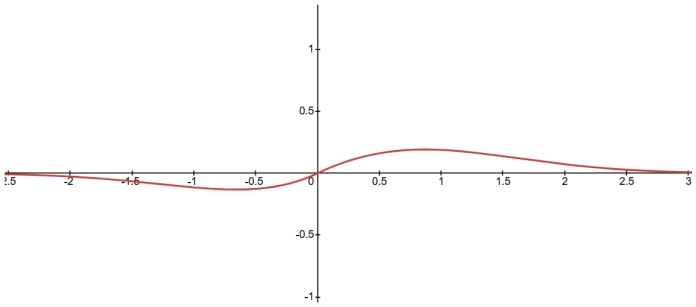
$$Scale \cdot e^{-\frac{(x-Shift)^2}{2}}$$

- α and β (of Swish) are the hyperparameter adjusted based on experimentation and learning.

The plot of this activation function based on α, which is dependent on the input to the activation function, is given in Figure 3.2.



(a) Behave like Swish(x) when $x \leq \alpha$



(b) Elliot(x) $\times$ Scaled Shifted Gaussian(x) when $x > \alpha$

Figure 3.2: Behavior of AdaptoSwelliGauss

Some properties that we can read from the figure are that the function is bounded below, it is non-monotonic, and it is also not differentiable everywhere because of Elliot.

3.3 Integrated Collision Detection and Avoidance by Path and Start Time Changes (ICDAPS)

As discussed in the introduction and literature review, this section presents our system designed for efficient detection and avoidance of collisions among UAVs. The detection process is straightforward, but there are two methods for collision avoidance: one by altering the UAV's path and the other by adjusting its start time. The novelty of our approach lies in integrating these two strategies. We define the system as efficient when it minimizes the need for infinite path changes to altering the UAV's path and reduces the batch size of UAVs dispatched using start time adjustments. Each component has been designed to function synergistically, enhancing the overall efficacy of the UAV management system.

1. **Collision Detection:** A geometric approach, as described in the works of (14), is employed for collision detection. We use the preliminary technique of shallow detection there. More detailed techniques can also be integrated, which will not change the overall execution of our algorithm.

2. **Collision Avoidance by path changes:** As mentioned in the literature review section Table-2, there are multiple paper who have done trajectory adjustment (16), (17), (18), (14) and (1). They have achieved this by changing the velocity or by adding extra forces on UAVs to update path, etc. In our context, we have been provided the location of the UAV at every time instance (for example, every 1 sec, 0.1 sec, or 0.2 sec), and the velocity and acceleration are automatically adjusted based on the data. Hence we don't need to change the velocity of UAV or any other properties. If we change the position of the UAV at that colliding time instant, then we can achieve our goal.

   To facilitate integration with the subsequent component, a novel `tracking array` is introduced, which monitors the number of trajectory manipulations for each UAV. We also introduce a `batching list`, which stores the UAVs for whom this collision avoidance strategy fails.

3. **Collision Avoidance by start time change (Batching):** The batching

technique, as outlined in (12), is adopted here. We synchronize this with the above collision avoidance component.

Next, we discuss the above three components in detail.

3.3.1 Collision Detection

As practically common, we assume that all UAVs take off and fly simultaneously, following predefined trajectories between their respective pickup and delivery points. In our collision detection methodology, we employ a three-dimensional virtual sphere around each UAV referred to as the "collision sphere" at every time instance. This sphere is defined by a parameter "R" that denotes its radius (See Figure 3.3). On a broader level, a collision is defined as when any UAV collision sphere intersects with the collision sphere of another UAV at that particular time instance. As mentioned earlier in the thesis, this strategy closely follows one of the approaches given in (14). Next, we make this statement precise.

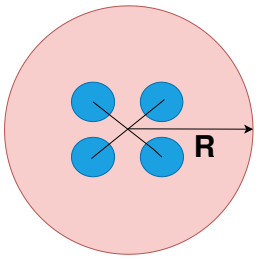


Figure 3.3: Collision sphere around the UAV.

Let us be given a list of UAVs and their respective trajectories in terms of X, Y, and Z coordinates, also called waypoints at different time instances. The granularity of the time instance can be adapted based on user need, for this we use at every 1 second, which is common(17). Without loss of generality for a UAV pair, the standard algorithm calculates the Euclidean distance between their position at each waypoint along their paths and checks if this distance falls below a user-defined distance threshold (again, waypoints correspond to every time instant).

The position vector of the i^{th} UAV at waypoint w is given by $\mathbf{p}_i(w)$. The Euclidean

distance between the i^{th} and j^{th} UAVs at this waypoint is

$$d_{ij}(w) = \|\mathbf{p}_i(w) - \mathbf{p}_j(w)\| \tag{3.10}$$

A collision is detected, if the distance condition is satisfied, i.e., if $d_{ij}(w) < 2R + \delta$ where δ is the user-defined distance for collision detection.

If a collision is detected, then instead of the standard procedure of building a list of colliding pairs of UAVs, we apply the collision avoidance strategy discussed below. Finally, this procedure is repeated for all pairs.

3.3.2 Collision Avoidance

The primary objective here is to dynamically adjust the trajectory of two UAVs on a collision course. Without loss of generality, we assume UAV1 and UAV2 are two UAVs that would collide. Without loss of generality again, we change the trajectory of UAV1.

Assuming that two UAVs are colliding at waypoint w_c, which is a three-dimensional array. We have the option of changing UAV1 in and of the X,Y, and Z directions. Without loss of generality we change location of UAV1 in X direction. i.e. We change w_c (X) to w_c (X to 2R+`safe distance`). Here R is the collision sphere, discussed in the detection section and `safe distance` is the extra distance between the "collision sphere" of two UAVs. This approach moves the UAV1 to a different location, there might be a jerk by the sudden change in the location of UAV1, to make this process smoother we start changing the location of UAV1, from K waypoint before and K waypoint after collision waypoint w_c (Which is equivalent to saying K time instances before the colliding time instance and K time after the colliding time instance). This process is mathematically demonstrated below.

$$\begin{pmatrix} w_{c-k} \\ \vdots \\ w_{c-1} \\ w_c \\ w_{c+1} \\ \vdots \\ w_{c+K} \end{pmatrix} + \begin{pmatrix} \left(\frac{2R+\text{safe distance}}{K}\right) \times 1 \\ \vdots \\ \left(\frac{2R+\text{safe distance}}{K}\right) \times k-1 \\ 2R+\text{safe distance} \\ \left(\frac{2R+\text{safe distance}}{K}\right) \times k-1 \\ \vdots \\ \left(\frac{2R+\text{safe distance}}{K}\right) \times 1 \end{pmatrix} = \begin{pmatrix} w_{c-k} + \left(\frac{2R+\text{safe distance}}{K}\right) \times 1 \\ \vdots \\ w_{c-1} + \left(\frac{2R+\text{safe distance}}{K}\right) \times k-1 \\ w_c + 2R+\text{safe distance} \\ w_{c+1} + \left(\frac{2R+\text{safe distance}}{K}\right) \times k-1 \\ \vdots \\ w_{c+K} + \left(\frac{2R+\text{safe distance}}{K}\right) \times 1 \end{pmatrix}$$

We now introduce our contribution involving integration with a batching mechanism. The trajectory adjustments are recorded in a `tracking array` for each UAV, ensuring that adjustments do not exceed a predefined limit, preventing infinite adjustment loops (as seen in standard approaches). If this limit is reached for a UAV, the trajectory of the other UAV in the colliding pair is examined. Should the adjustment for the second UAV also exceed the limit, both UAVs are moved to a `batching list` as separate entries.

As shown above, we discuss the collision detection and avoidance between different UAVs when simulating a fleet of UAVs. However, a more practical scenario involves obstacles appearing in the path of UAVs, whether they are static or dynamic. In this case, the VFH+ (Vector Field Histogram Plus) algorithm is more industry-standard and uses LiDAR sensors. For the sake of the reader's understanding, we have mentioned this in Appendix B.

3.3.3 Batching Mechanism

The batch generation algorithm is responsible for organizing UAVs into batches to facilitate coordinated flight. Its goal is to form collision-free batches that can be managed as a single unit. In the standard approach, the list of colliding pairs of UAVs is used as a input here.

Next, we describe our approach, which integrates with the above collision avoidance strategy. Here, the input to the batching algorithm is the `batching list`. The UAVs that are not on the `batching list` are free from any collision and outputted as one batch.

Next, the 1^{st} UAV from the batching list is picked for the next batch. This 1^{st} UAV is checked for a collision with all the subsequent UAVs in the list (2^{nd} UAV onwards).

WLOG, let the i^{th} UAV does not collide with the 1^{st} UAV, then 1^{st} and i^{th} UAVs are added to this next batch, and both are further checked for collision from $(i+1)^{th}$ UAV in the batching list. WLOG let j^{th} UAV does not collide with both the 1^{st} and i^{th} UAV. Then, j^{th} UAV is also added to the next batch. Further, all three are checked for collision from $(j+1)^{th}$ UAV in the batching list.

This process is recursively done to determine this final next batch as well as all subsequent batches.

Chapter 4

Results

Each UAV's output trajectory is generated with a total of 201 waypoints, providing a rich and detailed dataset conducive to robust algorithm development. This data is generated using a UAV simulator implemented with the UAV toolbox of Simulink in MATLAB. The simulation is run on a PC equipped with an Intel Core i7 12th generation processor, a 64-bit operating system, and 8GB of RAM.

For the input data, the configuration of initial and destination coordinates for each UAV is taken from (6), which provides guidance on setting realistic ranges. The X and Y coordinates for the initial position range from [10-50], with a fixed initial Z-coordinate at 0. Destination coordinates are constrained within X, Y [200-300] with the Z-coordinate again fixed at zero. There is a third point, which is the middle point for X and Y coordinates, is taken as the average of the initial and destination coordinates, while the Z middle point is between [30-40].

The 500 UAV dataset is partitioned into three distinct segments: 70% is allocated for training, 15% for validation, and the remaining 15% for testing. The computational process iterates over 1000 epochs to optimize outcomes. A pictorial representation of the paths of all the UAVs is shown in Figure 4.1.

The choice of a loss function is crucial for training neural networks, as it measures the difference between predicted and actual values. For regression tasks, including UAV trajectory prediction, Mean Squared Error (MSE) is widely used in the literature (33), (7), (10) and more. Given its effectiveness in minimizing prediction errors by averaging the squared differences, we also adopt MSE in this work. While other loss functions exist, MSE remains the standard for its simplicity and reliability in

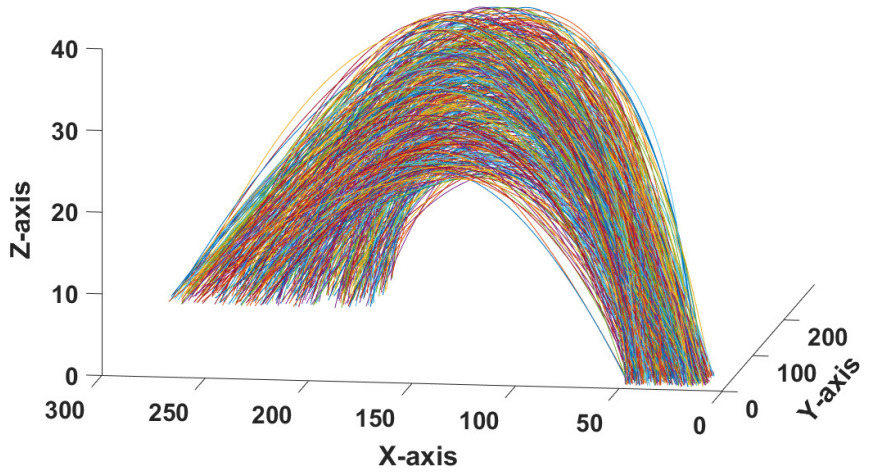


Figure 4.1: Dataset of 500 UAVs.

regression problems.

The rest of this Section has two parts. In Section 4.1, we present appropriate loss function values on the standard activation functions and our novel AdaptoSwelli-Gauss activation function (from Sections 3.1 and 3.2). In Section 4.2, we give experimental results using ICDAPS techniques (from Section 3.3)

4.1 Relevant Loss Functions Values Using Different Activations

For these experiments, as mentioned earlier, the value of hyperparameter α is taken as the median of the input to the activation function. This ensures that Swish and Elliot times Gaussian are used an equal number of times. The value of other hyperparameter β, $scale$, and $shift$ are taken as 0.14, 0.5 and 0.25, respectively.

The errors we report for the X, Y, and Z coordinates below are computed as follows: We calculate the average error across 201 waypoints, derived from 15% of a dataset containing 500 UAVs (i.e., test data consisting of 75 UAVs). The results for the optimal epoch are listed below.

The best performance on the standard activation function is Relu, which gives the order of 10^{-10}, 10^{-8}, and 10^{-9} for X,Y, and Z, respectively. While we look at

Table 4.1: Error Metrics for the X-coordinate.

No.	**Category**	**Activation Function**	**MSE**
1-3	Standard	Sigmoid	9.200×10^{-5}
		Tanh	1.406×10^{-7}
		ReLU	1.733×10^{-10}
4-6	Application-Oriented	Swish	1.638×10^{-9}
		Elliot	3.605×10^{-11}
		AdaptoSwelliGauss	3.059×10^{-14}

Table 4.2: Error Metrics for the Y-coordinate.

No.	**Category**	**Activation Function**	**MSE**
1-3	Standard	Sigmoid	1.822×10^{-8}
		Tanh	8.664×10^{-7}
		ReLU	1.793×10^{-8}
4-6	Application-Oriented	Swish	6.805×10^{-10}
		Elliot	3.780×10^{-9}
		AdaptoSwelliGauss	5.127×10^{-11}

Table 4.3: Error Metrics for the Z-coordinate.

No.	**Category**	**Activation Function**	**MSE**
1-3	Standard	Sigmoid	4.588×10^{-11}
		Tanh	8.031×10^{-7}
		ReLU	5.851×10^{-9}
4-6	Application-Oriented	Swish	2.182×10^{-13}
		Elliot	4.851×10^{-8}
		AdaptoSwelliGauss	1.739×10^{-13}

our purposed activation function, it gives an error of the order of 10^{-14}, 10^{-11}, and 10^{-13} in X, Y, and Z, respectively. Thus we see a reduction of 10^{-3} to 10^{-4} order of magnitude in error.

AdaptoSwelliGauss *consistently demonstrates substantially superior performance compared to all of these*, highlighting its robustness and effectiveness in enhancing the accuracy and predictive capabilities of the neural network across different dimensions.

4.2 Results On ICDAPS

When we apply collision detection as described in Section 3.3.1 we detect 262 collisions. Here, the value of "R" is taken as 0.5. Next, we apply the original (or standard) collision avoidance as initially described in Section 3.3.2 (i.e., one where infinite trajectory manipulation is allowed), and the algorithm gets stuck in a loop.

Thus, further, we apply our modified collision avoidance algorithm (as given at the end of Section 3.3.2) along with batching (from Section 3.3.3). This integration leads to substantial improvement in both components.

First, the application of the modified collision avoidance (i.e., up to ten trajectory manipulations allowed) leads to convergence of the algorithm, eventually resulting in only 41 possible collisions. Batching brings these collisions down to zero. Here, the safe radius taken is 0.5.

Second, results for batching integrated with the modified collision avoidance are given in Table 4.4. As evident from this table, the number of batches required is reduced from 7 to 5, while the maximum number of UAVs per batch increased from 38 to 315.

Table 4.4: Result before and after applying ICDAPS.

Metric	**Before Collision Avoidance**	**After Collision Avoidance**
Number of Batches	7	5
Maximum number of UAVs per Batch	38	315

Next, we show the importance of carefully calibrating the safe radius in our IC-

DAPS algorithm. As shown in Table 4.5, increasing the safe radius initially results in a decrease in the number of colliding UAVs, decrease in number of batches, and increase in the maximum number of UAVs per batch, achieving optimal results at a safe radius of 2.3. At this radius, the number of colliding UAVs is reduced to 33, the number of batches is minimized to 4, and the maximum number of UAVs per batch increases to 396. However, beyond this optimal point, further increases in the safe radius lead to a reverse trend, where both the number of colliding UAVs and the number of batches begin to increase again, while the maximum number of UAVs per batch decreases. These results are subject to the characteristics and dynamics of the specific dataset under consideration, and different datasets may yield different optimal safe radius values. However, there is a scientific reason for this observed trend.

Table 4.5: ICDAPS outcomes with different safe radius values.

Safe Radius	Number of Collision After Avoidance	Number of Batches	Max Number of UAVs per Batch
0.5	47	5	315
0.7	46	6	321
1.9	39	7	326
2.1	35	4	378
2.3	33	4	396
2.5	42	6	335
2.7	61	8	305
2.9	65	10	278

This is because of unintended consequences of over-adjusting UAV trajectories at larger safe radius (see Figure 4.2). As in the given figure, when the trajectory of UAV1 is significantly altered to avoid UAV2, it may inadvertently collide with UAV3, which was previously not on a collision course with UAV1. This has a cascading effect on all the UAVs.

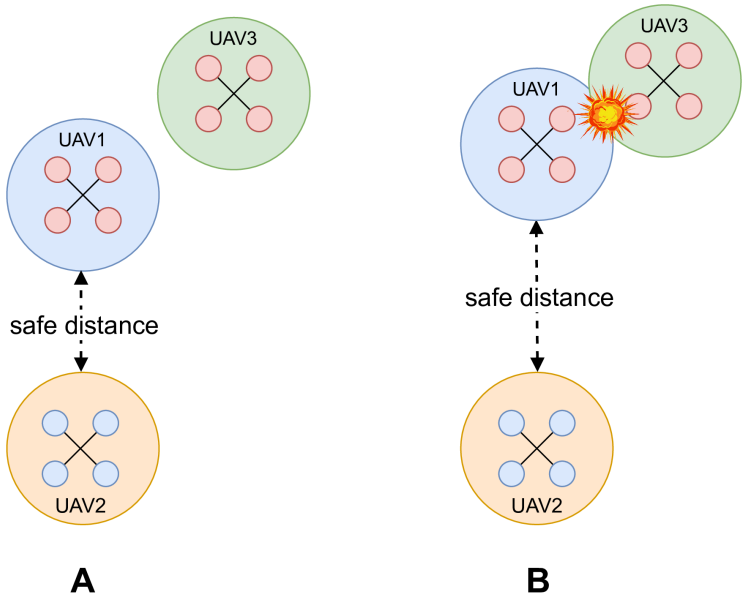
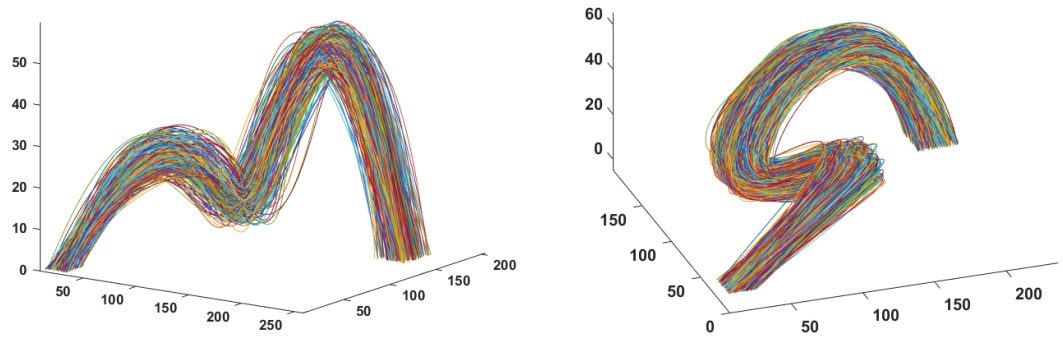


Figure 4.2: **A:** ICDAPS with small safe radius, **B:** ICDAPS with large safe radius.

4.2.1 Sensitivity Analysis of ICDAPS

To demonstrate the robustness and adaptability of the proposed ICDAPS algorithm, we evaluate its performance on datasets of varying UAV fleet sizes. In addition to the 500 UAVs considered in the main experiments (Figure 4.1), we also apply the ICDAPS approach to smaller and larger fleets comprising 250 and 750 UAVs, respectively, as depicted in Figure 4.3. The results, presented in Table 4.6, indicate that the proposed method maintains consistent performance across different scenarios, highlighting its scalability and effectiveness for UAV collision detection and avoidance in diverse operational contexts.



(a) 250 UAVs dataset (b) 750 UAVs dataset

Figure 4.3: Different datasets of UAVs

Table 4.6: ICDAPS analysis with different datasets

	Number of UAVs	**Number of Collisions**	**Batch size**	**Maximum number of UAVs per batch**
Before ICDAPS	250	195	7	56
After ICDAPS		32	4	180
Before ICDAPS	750	695	31	55
After ICDAPS		274	14	426

Chapter 5

Conclusion

In this thesis, we focus on the development of an intelligent framework for predicting and optimizing the trajectory of a fleet of UAVs. Traditional methods have limitations in accurately predicting paths as well as efficiently avoiding collisions for large fleets of UAVs. We address these challenges here in two paragraphs below.

In the past, people have used FFNN with a single hidden layer with a standard activation function, which gave a high error. We use the application-oriented activation function and also propose a new activation function, AdaptoSwelliGauss, which reduces the error with three to four order.

We improve existing methods for collision detection and avoidance. Detection is straightforward while avoidance is challenging. We can avoid collision between UAVs by either changing their trajectories or sending them in batches. Each of these has its drawbacks. First has the drawback of throwing the result in an infinite loop, and second has the drawback of a large number of batch sizes. We uniquely combine those techniques, leading to finite trajectory changes in the first approach and about half of the reduction in the number of batches in which UAVs are dispatched.

Future work involves improving the underlying mathematical optimization (34); exploring Convolutional Neural Networks (CNNs) for enhanced trajectory accuracy (35); preliminary investigations of Deep Neural Networks (1D Convolutional Neural Networks (1D CNN) have been presented in Appendix A. However, automating model selection and hyperparameter tuning will be a key focus in future work (36); using approximate computing in neural networks (37; 38); leveraging multi-agent reinforcement learning for collaborative decision-making (39); exploiting implicit re-

lation between different drone trajectories (40).

Bibliography

[1] J. Tang, L. Fan, and S. Lao, "Collision avoidance for multi-uav based on geometric optimization model in 3d airspace," *Arabian Journal for Science and Engineering*, vol. 39, pp. 8409–8416, 2014.

[2] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari, "Path planning algorithms in the autonomous driving system: A comprehensive review," *Robotics and Autonomous Systems*, vol. 174, p. 104630, 2024.

[3] X. Xu, C. Xie, Z. Luo, C. Zhang, and T. Zhang, "A multi-objective evolutionary algorithm based on dimension exploration and discrepancy evolution for UAV path planning problem," *Information Sciences*, vol. 657, p. 119977, 2024.

[4] A. F. Hasan, A. J. Humaidi, A. S. M. Al-Obaidi, A. T. Azar, I. K. Ibraheem, A. Q. Al-Dujaili, A. K. Al-Mhdawi, and F. A. Abdulmajeed, *Fractional Order Extended State Observer Enhances the Performance of Controlled Tri-copter UAV Based on Active Disturbance Rejection Control*. Cham: Springer International Publishing, 2023, pp. 439–487.

[5] H. Qiu and H. Duan, "A multi-objective pigeon-inspired optimization approach to UAV distributed flocking among obstacles," *Information Sciences*, vol. 509, pp. 515–529, 2020.

[6] R. Lai, "A machine learning approach to trajectory planning for UAV," Master's thesis, 2020.

[7] M. Xue, "UAV trajectory modeling using neural networks," in *17th AIAA Aviation Technology, Integration, and Operations Conference*. ARC, 2017, p. 3072.

[8] H. Al-Khazraji, A. R. Nasser, A. M. Hasan, A. K. Al Mhdawi, H. Al-Raweshidy, and A. J. Humaidi, "Aircraft engines remaining useful life prediction based on a hybrid model of autoencoder and deep belief network," *IEEE Access*, vol. 10, pp. 82 156–82 163, 2022.

[9] S. Jeong, K. You, and D. Seok, "Hazardous flight region prediction for a small UAV operated in an urban area using a deep neural network," *Aerospace Science and Technology*, vol. 118, p. 107060, 2021.

[10] S. Sarkar, M. W. Totaro, and A. Kumar, "An intelligent framework for prediction of a UAV's flight time," in *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2020, pp. 328–332.

[11] K. Xiao, J. Zhao, Y. He, and S. Yu, "Trajectory prediction of UAV in smart city using recurrent neural networks," in *ICC 2019-2019 IEEE International Conference on Communications*. IEEE, 2019, pp. 1–6.

[12] C. Sastre, J. Wubben, C. T. Calafate, J. C. Cano, and P. Manzoni, "Collision-free swarm take-off based on trajectory analysis and UAV grouping," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2022, pp. 477–482.

[13] C. Sastre, J. Wubben, C. T. Calafate, J.-C. Cano, and P. Manzoni, "Safe and efficient take-off of VTOL UAV swarms," *Electronics*, vol. 11, no. 7, p. 1128, 2022.

[14] F. Ho, R. Geraldes, A. Gonçalves, M. Cavazza, and H. Prendinger, "Improved conflict detection and resolution for service UAVs in shared airspace," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1231–1242, 2019.

[15] B. Jiang, B. Li, W. Zhou, L.-Y. Lo, C.-K. Chen, and C.-Y. Wen, "Neural network based model predictive control for a quadrotor UAV," *Aerospace*, vol. 9, no. 8, p. 460, 2022.

[16] J.-W. Park, H.-D. Oh, and M.-J. Tahk, "UAV collision avoidance based on geometric approach," in *2008 SICE Annual Conference*, 2008, pp. 2122–2126.

[17] G. Elmkaiel and V. V. Serebrenny, "Collision avoidance algorithm for a quadcopters swarm," *AIP Conference Proceedings*, vol. 2171, no. 1, p. 190006, 11 2019.

[18] Y. Wan, J. Tang, and S. Lao, "Distributed conflict-detection and resolution algorithm for UAV swarms based on consensus algorithm and strategy coordination," *IEEE Access*, vol. 7, pp. 100 552–100 566, 2019.

[19] M. Hagan and M. Menhaj, "Training feedforward networks with the marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[20] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," in *2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 1836–1841.

[21] G. Orr and K. Müller, *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003.

[22] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions," *Neural Networks*, vol. 138, pp. 14–32, 2021.

[23] O. Yenigün, "Choosing the right activation function in deep learning: A practical overview and comparison," *Medium*, 2023. [Online]. Available: https://tinyurl.com/53ub978e

[24] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv:1710.05941*, 2017.

[25] A. B. Dash, "Top 10 activation function's advantages and disadvantages," *Linkedin*, 2021. [Online]. Available: https://www.linkedin.com/pulse/top-10-activation-functions-advantages-disadvantages-dash/

[26] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," 2013.

[27] C. Dennis, A. Engelbrecht, and B. Ombuki-Berman, "An analysis of activation function saturation in particle swarm optimization trained neural networks," *Neural Processing Letters*, vol. 52, 10 2020.

[28] Z. Shi, J. Zhang, G. Shi, L. Ji, D. Wang, and Y. Wu, "Design of a UAV trajectory prediction system based on multi-flight modes," *Drones*, vol. 8, no. 6, 2024.

[29] S. K. Singh, A. Sinha, and S. R. Kumar, "Nonlinear control design for an unmanned aerial vehicle for path following," *IFAC-PapersOnLine*, vol. 55, no. 1, pp. 592–597, 2022, 7th International Conference on Advances in Control and Optimization of Dynamical Systems ACODS 2022.

[30] A. Nikhade, "Swish activation function," *Medium*, 2023. [Online]. Available: https://amitnikhade.medium.com/swish-activation-function-d106fe13930e

[31] J. L. Salmeron and A. Ruiz-Celma, "Elliot and symmetric elliot extreme learning machines for gaussian noisy industrial thermal modelling," *Energies*, vol. 12, no. 1, 2019.

[32] A. S., "Gaussian activation function," *codecademy*, 2024. [Online]. Available: https://www.codecademy.com/resources/docs/ai/neural-networks/gaussian-activation-function#

[33] B. Jiang, B. Li, W. Zhou, L.-Y. Lo, C.-K. Chen, and C.-Y. Wen, "Neural network based model predictive control for a quadrotor uav," *Aerospace*, vol. 9, no. 8, 2022.

[34] K. Ahuja, L. T. Watson, and S. C. Billups, "Probability-one homotopy maps for mixed complementarity problems," *Computational Optimization and Applications*, vol. 41, pp. 363 – 375, 2008.

[35] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[36] R. H. Hadi, H. N. Hady, A. M. Hasan, A. Al-Jodah, and A. J. Humaidi, "Improved fault classification for predictive maintenance in industrial iot based on automl: A case study of ball-bearing faults," *Processes*, vol. 11, no. 5, 2023.

[37] S. Gupta, S. Ullah, K. Ahuja, A. Tiwari, and A. Kumar, "ALigN: A highly accurate adaptive layerwise Log_2_Lead quantization of pretrained neural networks," *IEEE Access*, vol. 8, p. 118899, 2020.

[38] S. Ullah, S. Gupta, K. Ahuja, A. Tiwari, and A. Kumar, "L2L: A highly accurate Log_2_Lead quantization of pretrained neural networks," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 979–982.

[39] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Multi-agent reinforcement learning: A review of challenges and applications," *Applied Sciences*, vol. 11, no. 11, p. 4948, 2021.

[40] S. Kim, U. Murthy, K. Ahuja, S. Vasile, and E. A. Fox, "Effectiveness of implicit rating data on characterizing users in complex information systems," in *Research and Advanced Technology for Digital Libraries (ECDL 2005), Lecture Notes in Computer Science*, A. Rauber, S. Christodoulakis, and A. M. e. Tjoa, Eds. Springer, 2005, vol. 3652, pp. 186 – 194.