Faculty of Computers and Artificial Intelligence
Cairo University

Movement Fault Detection in Legged Robots with AI

Graduation Project

Academic Year 2023-2024

Final Documentation

Supervised by :

Dr. Mustafa Shiple
TA. Mohamed Halal

PROJECT LEAD TIME: 1 year

REPORT PREPARED AND WRITTEN BY:

Joeshwoa George Abo Hager Melek	**20200131**
Mostafa Mohamed Nabil Mahmoud	**20200548**
Osama Ibrahim Marzok Eid	**20200069**
Sameh Raouf Helmy Hanna	**20200218**

TABLE OF CONTENTS

A. INTRODUCTION

n today's rapidly evolving world, the quest for solutions that are more efficient, faster, and resource-saving has never been more critical. The demand for systems that operate with minimal human intervention is driving significant advancements in technology. This trend is particularly evident in the fields of robotics and artificial intelligence (AI), where companies are investing heavily in research and development. The ultimate aim is to create systems that not only perform tasks more effectively but also offer unprecedented levels of autonomy and self-reliance.

Artificial intelligence has undergone remarkable transformations over the past few years. Breakthroughs in areas such as deep learning, machine learning, and self-learning algorithms have revolutionized the way AI systems are developed and deployed. These advancements have enabled AI to permeate almost every aspect of our daily lives, from personal assistants like Siri and Alexa to complex decision-making systems in healthcare, finance, and beyond.

AI's integration into various sectors has led to significant improvements in efficiency and effectiveness. For example, in healthcare, AI is used for diagnosing diseases, predicting patient outcomes, and personalizing treatment plans. In finance, AI algorithms analyze vast amounts of data to detect fraud, assess risks, and optimize investment strategies. The potential applications are limitless, and the benefits substantial.

Robotics, another critical area of technological advancement, has seen widespread adoption across multiple domains. Robots are now commonplace in environments ranging from hospitals and disaster-stricken areas to restaurants and warfare. In hospitals, robots assist in surgeries, transport medications, and provide companionship to patients. During natural disasters, they are deployed for search and rescue missions, navigating hazardous terrains where human intervention would be perilous.

In the culinary world, robots are revolutionizing the restaurant industry by performing tasks such as food preparation, cooking, and even customer service. These applications not only enhance efficiency but also ensure consistency and quality in service delivery. However, the primary objective of these technological advancements remains to benefit humanity. The focus is on creating solutions that improve quality of life, increase safety, and provide assistance where it is most needed.

Despite the vast potential and diverse applications of robots and AI, ethical considerations are paramount. The deployment of robots and AI systems must align with humanitarian goals, ensuring that they are used to support and enhance human life rather than cause harm. The vision is to leverage these technologies for positive purposes, such as improving healthcare outcomes, enhancing disaster response, and providing better services in various industries.
It is crucial to steer the development and use of robots away from applications that could lead to harm, oppression, or unethical practices. The ethical framework guiding AI and robotics development emphasizes transparency, accountability, and fairness. This includes addressing concerns about privacy, security, and the potential biases in AI algorithms.

One of the significant challenges in the deployment of robots, particularly legged robots, is the detection and management of movement faults. Legged robots, designed to navigate complex terrains

and environments, are susceptible to various mechanical and operational faults. These faults can range from joint malfunctions and sensor failures to more complex issues like imbalance and slippage. Detecting these faults in real-time is critical to ensuring the reliability and safety of legged robots. Traditional fault detection methods, often based on predefined models and manual inspections, are increasingly being supplemented by AI-driven approaches. AI-based fault detection systems offer several advantages, including the ability to learn from data, adapt to new fault types, and provide real-time monitoring and diagnostics.

The integration of AI into fault detection systems for legged robots represents a significant advancement. AI algorithms, particularly those involving machine learning and deep learning, can analyze vast amounts of sensor data to identify patterns and anomalies indicative of faults. These systems can predict potential failures before they occur, allowing for proactive maintenance and reducing downtime.
Moreover, AI-driven fault detection systems can continuously improve through self-learning mechanisms. By analyzing historical data and learning from past faults, these systems can enhance their accuracy and reliability over time. This capability is particularly valuable in dynamic and unpredictable environments, where the ability to adapt and learn is crucial.

This project focuses on developing an AI-based movement fault detection system for legged robots. The primary goal is to leverage advanced AI techniques to enhance the accuracy, reliability, and efficiency of fault detection processes. The project involves collecting and analyzing sensor data from a quadruped robot, developing machine learning models to detect faults, and integrating these models into the robot's control system.

The expected outcomes include improved fault detection performance, reduced maintenance costs, and enhanced operational safety for legged robots. By addressing the challenges associated with traditional fault detection methods, this project aims to contribute to the broader field of robotics and AI, offering practical solutions that can be applied in various real-world scenarios.

B. ABOUT THE PROJECT

Hexa robots, a type of legged robot typically featuring six legs, have become indispensable tools in various critical applications. These robots are specifically designed to navigate challenging terrains and perform tasks that would be hazardous or impossible for humans. One of their most significant applications is in search and rescue operations. In the aftermath of natural disasters, such as earthquakes, Hexa robots are deployed to locate injured individuals trapped under rubble. Their ability to maneuver through debris and tight spaces makes them invaluable for these missions.
Additionally, Hexa robots play a crucial role in exploration missions, both on Earth and other planets. Their design allows them to traverse rugged terrains, making them ideal for geological surveys and environmental monitoring in remote areas. In planetary exploration, such as missions to Mars or the Moon, Hexa robots can analyze the terrain, collect samples, and send valuable data back to Earth. These capabilities highlight the importance of ensuring their operational reliability and robustness.

Despite their advanced capabilities, legged robots, including Hexa robots, face significant challenges. One of the most critical issues is the failure of leg motors. Each leg typically comprises multiple motors (two or three), and the failure of even a single motor can compromise the robot's mobility. Given that these robots often have four or more legs, the complexity of maintaining their functionality increases exponentially.

The failure of leg motors can result from various factors, including mechanical wear and tear, environmental conditions, and electrical issues. When a motor fails, the robot may lose its ability to walk correctly, significantly impacting its ability to perform its intended tasks. This malfunction can lead to the loss of expensive robotic units, which is a significant concern given the high cost of these sophisticated machines.

To address motor failures, current methods rely heavily on the use of sensors to monitor the health of the motors. These sensors collect data on various parameters, such as motor temperature, vibration, and electrical current, to determine whether a motor is functioning correctly. While this approach can be effective, it also has several drawbacks.

Firstly, equipping each motor with multiple sensors increases the overall cost of the robot. Given that a single Hexa robot can have more than a dozen motors, the cumulative cost of the sensors can be substantial. Additionally, sensors themselves are not immune to failure. If a sensor malfunctions, it can provide false data, leading to incorrect diagnoses and potentially exacerbating the problem. Thus, while sensor-based fault detection is a step in the right direction, it is not a foolproof solution.

Traditional solutions to motor failure in legged robots often involve changing the gait or movement pattern of the robot. By adjusting the speed of the healthy motors or altering the sequence of their movements, the robot can compensate for the malfunctioning motor to some extent. This approach, however, has its limitations.
While changing the gait can help the robot continue moving, it does not address the root cause of the problem—the broken motor. Moreover, the effectiveness of this method is limited. If multiple motors fail or if the robot encounters particularly challenging terrain, compensatory gait changes may not suffice to maintain functionality. Thus, while gait adjustment can provide a temporary workaround, it is not a long-term solution to the problem of motor failures.

Given the limitations of current methods, there is a clear need for more advanced and reliable fault detection systems for legged robots. This project aims to develop an AI-based system for detecting movement faults in Hexa robots, with a focus on identifying motor failures accurately and in real-time. The primary objective is to leverage advanced machine learning techniques to analyze data from existing sensors more effectively, reducing the reliance on additional hardware and mitigating the risk of sensor failures.

The proposed AI-based fault detection system will utilize machine learning algorithms to analyze sensor data and detect anomalies indicative of motor failures. By training the AI models on historical data, the system can learn to recognize patterns associated with both healthy and malfunctioning motors. This approach offers several advantages over traditional methods.
Firstly, AI models can process and analyze vast amounts of data much faster and more accurately than human operators or simple rule-based systems. This allows for real-time fault detection, enabling immediate corrective actions. Secondly, machine learning algorithms can continuously learn and improve over time. As the system is exposed to more data and diverse scenarios, its accuracy and reliability will increase.

The successful implementation of this project is expected to yield several significant benefits. Improved fault detection will enhance the reliability and operational lifespan of Hexa robots, reducing the likelihood of mission failures and the associated costs. Additionally, by minimizing the need for additional sensors, the overall cost of the robots can be reduced, making them more accessible for various applications.
Furthermore, the insights gained from this project can be applied to other types of legged robots, contributing to the broader field of robotics. By advancing the state of fault detection technology, this project aims to set a new standard for reliability and efficiency in robotic systems.

Importance of Hexapod Robots :
Hexapod robots, characterized by their six-legged structure, are increasingly recognized for their unique capabilities and advantages. Their design provides them with several critical benefits that make them invaluable in various applications. This section delves into the importance of hexapod robots, focusing on their stability, mobility and adaptability, payload capacity, redundancy, and their role in exploration.

1. Stability
One of the most significant advantages of hexapod robots is their inherent stability. The six-legged design offers a robust and balanced platform that can maintain stability even on uneven terrain. Unlike bipedal robots, which require complex algorithms to stay upright, hexapods can distribute their weight more evenly and maintain multiple points of contact with the ground at all times.
Static Stability :
Hexapod robots can achieve static stability, which means they can remain stable even when not in motion. With three legs always on the ground, forming a tripod, they provide a stable base that prevents tipping over. This static stability is crucial in environments where sudden movements or shifts in terrain can occur, such as during search and rescue operations in disaster zones.
Dynamic Stability :

In addition to static stability, hexapods also exhibit dynamic stability, allowing them to adapt to changes in terrain while in motion. Their control systems can adjust the position and movement of each leg in response to sensor feedback, ensuring continuous stability as they navigate obstacles. This capability is vital for maintaining balance while walking on rocky surfaces or inclining planes, where other types of robots might struggle to remain upright.

2. Mobility and Adaptability

Hexapod robots are renowned for their exceptional mobility and adaptability. Their legged design allows them to traverse a wide range of terrains that would be challenging or impossible for wheeled or tracked robots.

Terrain Adaptation :

Hexapods can adapt their gait and posture to different types of surfaces, including soft sand, gravel, snow, and uneven rocky terrain. Their legs can adjust their height and angle, enabling the robot to step over obstacles, climb inclines, and navigate through narrow passages. This adaptability makes them ideal for exploratory missions on rugged landscapes, whether on Earth or on other planets.

Versatile Gaits :

Hexapod robots can employ various gaits to optimize their movement based on the environment and task at hand. For example, they can use a tripod gait, where three legs are always on the ground, for maximum stability, or a wave gait, where legs move in sequence, for smoother and faster movement. This versatility in gait selection enhances their ability to perform diverse tasks, from precise manipulation to rapid traversal.

3. Payload Capacity

The design of hexapod robots allows them to carry significant payloads relative to their size. Their multiple legs distribute the weight evenly, reducing the strain on individual actuators and ensuring stable movement even under heavy loads.

Enhanced Load Distribution :

Hexapods can carry equipment, sensors, and other payloads essential for various missions. In search and rescue operations, they can be equipped with cameras, communication devices, and medical supplies. In scientific explorations, they can transport sampling tools and analytical instruments. The even load distribution across six legs minimizes the risk of mechanical failure and ensures the robot can operate effectively even with substantial payloads.

Customization for Specific Tasks :

The payload capacity of hexapod robots can be customized to suit specific tasks. Modular designs allow for the integration of different payload modules, depending on the mission requirements. This customization capability is beneficial in industrial applications where robots need to carry and manipulate heavy tools or materials, enhancing their utility and efficiency.

4. Redundancy

Hexapod robots offer a high level of redundancy, which is crucial for ensuring reliability and continued operation even in the event of partial system failures.

Motor and Leg Redundancy :

With six legs, a hexapod robot can continue to function effectively even if one or two legs become damaged or fail. The control system can reconfigure the gait to compensate for the loss, allowing the robot to maintain stability and mobility. This redundancy is a significant advantage in hazardous

environments where the likelihood of damage is high, such as during disaster response or in harsh industrial settings.

Sensor Redundancy :

Hexapods can also incorporate multiple sensors for redundancy. If a primary sensor fails, secondary sensors can take over its function, ensuring continuous operation. This redundancy is vital for applications requiring high reliability, such as autonomous navigation and environmental monitoring.

5. Exploration

Hexapod robots are particularly well-suited for exploration missions, both terrestrial and extraterrestrial. Their design and capabilities enable them to navigate and perform tasks in challenging and uncharted environments.

Terrestrial Exploration :

On Earth, hexapod robots are used in various exploration missions, from geological surveys to environmental monitoring. They can traverse difficult terrains, such as dense forests, mountainous regions, and caves, collecting data and samples that are critical for scientific research. Their ability to operate in remote and inaccessible areas makes them invaluable for studying ecosystems, climate change, and natural resources.

Extraterrestrial Exploration :

In space exploration, hexapod robots are designed to explore the surfaces of other planets and moons. Their adaptability and stability are crucial for navigating the unknown and unpredictable terrains of extraterrestrial bodies. For instance, a hexapod robot could be deployed on Mars to explore craters, analyze soil samples, and search for signs of past or present life. Their robust design ensures they can withstand the harsh conditions of space, including extreme temperatures and radiation.

Hexapod robots, with their unique design and capabilities, offer significant advantages in terms of stability, mobility and adaptability, payload capacity, redundancy, and exploration potential. Their ability to navigate challenging terrains, carry substantial payloads, and maintain functionality even in adverse conditions makes them invaluable tools for a wide range of applications. As technology advances, the role of hexapod robots is expected to expand further, driving innovations in fields such as disaster response, scientific exploration, and industrial automation. By leveraging their strengths, we can unlock new possibilities and achieve remarkable feats in various domains.

Some Hexapods Products from NVIDIA

NVIDIA has made significant strides in the field of robotics, providing cutting-edge technologies that enhance the capabilities and applications of hexapod robots. Their platforms, powered by advanced AI, GPU, and deep learning technologies, support the development of highly autonomous and efficient hexapod robots. Below are some notable NVIDIA products and technologies relevant to hexapod robots.

Hexapod Robot Development Using NVIDIA Technologies

Several hexapod robots have been developed using NVIDIA's Jetson AI computing modules and the Isaac platform. These robots showcase the practical applications and enhanced capabilities enabled by NVIDIA's technology.

1. **Reachy by Pollen Robotics**: While Reachy is not a hexapod, it utilizes NVIDIA Jetson for its AI processing. This demonstrates the flexibility and power of Jetson modules in enabling complex robotic behaviors, which can be adapted for hexapod designs to enhance their capabilities in interaction and task execution.
2. **Spot from Boston Dynamics**: Spot, a quadruped robot from Boston Dynamics, leverages NVIDIA Jetson for AI processing. Although Spot is a quadruped, the advanced capabilities in navigation and autonomy provided by Jetson modules highlight the potential for similar applications in hexapod robots.

One of the persistent challenges in the operation of hexapod robots is the failure of leg motors. Typically, solutions involve changing the gait of the robot by adjusting the speed of the healthy motors or altering their movement order. However, the primary issue remains the timely detection and repair of broken motors. Here, NVIDIA's AI technologies play a crucial role.

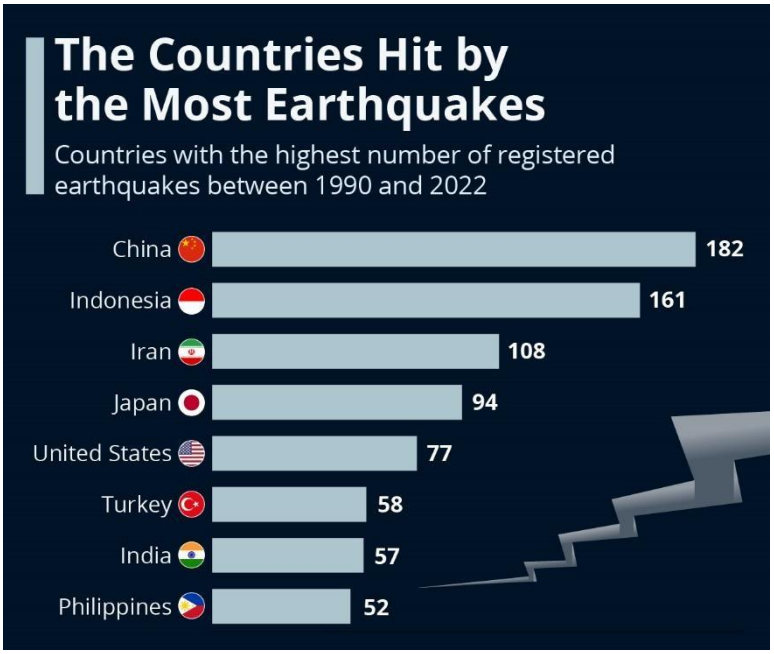
NVIDIA's advanced AI technologies and robotics platforms provide the tools necessary to enhance the capabilities and reliability of hexapod robots. By focusing on AI-driven fault detection, this project aims to address one of the critical challenges in the operation of legged robots. The successful implementation of this system will lead to significant resource savings, increased reliability, and broader adoption of hexapod robots in various industries, ultimately contributing to their effective deployment and utilization in diverse applications.

The problem of broken motors in legged robots is usually solved by changing the gate with which they move, which is by changing the speed of the healthy motors or the order of their movement, and this, but the biggest problem is still discovering the broken motors in order to repair them.

The goal of our project is to try to discover broken motors using artificial intelligence and try to address their steps
The project aims to save resources and the expected cost of building a legged robot in order to help spread legged robots and help prevent the loss of broken robots

C. MOTIVATION

The world we live in has become more challenging and unpredictable than ever before. It is a world teeming with problems, natural disasters, and accidents that demand innovative and robust solutions. As natural resources dwindle and the global population continues to rise, there is an urgent need for technologies that can address these growing challenges efficiently and effectively. This section delves into the motivation behind developing advanced hexapod robots, highlighting their critical role in addressing some of the most pressing issues of our time.



This chart explain the most countries that hit by earthquakes
This shows the extent of the world's exposure to natural disasters such as earthquakes and wars, and this is the importance of robotics in saving people in these disasters and avoiding the loss of human lives.

In recent years, the frequency and severity of natural disasters have surged dramatically. Earthquakes,

hurricanes, floods, and wildfires are becoming more common, causing widespread devastation and loss of life. These disasters often strike with little warning, leaving behind rubble and chaos that complicate rescue and recovery efforts.Hexapod robots have emerged as vital tools in disaster response and recovery operations. Their six-legged design allows them to navigate through debris and uneven terrain, making them ideal for searching for survivors in the aftermath of earthquakes or landslides. Equipped with advanced sensors and AI capabilities, hexapod robots can locate trapped individuals, assess structural damage, and provide real-time data to rescue teams. This ability to operate in hazardous environments significantly enhances the effectiveness of disaster response efforts, potentially saving countless lives.

Human ambition to explore space and understand our universe has reached unprecedented heights. Missions to the Moon, Mars, and beyond are no longer science fiction but tangible goals pursued by space agencies and private companies. These missions require technology that can operate autonomously in extremely harsh and unpredictable conditions. Hexapod robots are uniquely suited for space exploration due to their adaptability and stability. Their multiple legs allow them to traverse rocky and uneven surfaces found on other planets and moons. For instance, on Mars, where the terrain is challenging and varied, hexapod robots can explore craters, collect soil samples, and analyze geological formations. Their ability to carry scientific instruments and perform complex tasks autonomously makes them invaluable assets in the quest to explore and understand space.

The depletion of natural resources is a growing concern that threatens the sustainability of human societies. Overexploitation of resources such as minerals, fossil fuels, and water is leading to environmental degradation and ecological imbalance. There is an urgent need for technologies that can help monitor and manage these resources more effectively. Hexapod robots equipped with environmental sensors can play a crucial role in monitoring natural resources and assessing environmental health. These robots can be deployed in remote and challenging locations, such as forests, mountains, and oceans, to collect data on air and water quality, biodiversity, and soil conditions. By providing real-time information, hexapod robots can help scientists and policymakers make informed decisions about resource management and conservation efforts.

Enhancing Human Capabilities in Industrial Applications
The modern industrial landscape demands high levels of efficiency, precision, and safety. Traditional machinery and human labor alone are often insufficient to meet these demands, especially in environments that are hazardous or require repetitive tasks.
Industrial Applications of Hexapod Robots
Hexapod robots are increasingly being used in various industrial applications to enhance productivity and safety. In manufacturing, they can perform tasks such as welding, assembly, and quality inspection with high precision. In mining and construction, hexapod robots can navigate through unstable terrain to carry out inspections, drilling, and transportation of materials. Their ability to operate continuously in harsh conditions reduces the risk to human workers and increases overall efficiency.
Addressing the Limitations of Wheeled Robots
While wheeled robots have been widely used for numerous applications, they have inherent limitations when it comes to navigating complex and uneven terrains. Wheels are effective on flat and smooth surfaces but struggle in environments with obstacles, steep inclines, and irregular ground.
Advantages of Legged Robots Over Wheeled Robots
Hexapod robots, with their six-legged configuration, offer significant advantages over wheeled robots in challenging environments. They can step over obstacles, maintain stability on uneven ground, and adapt their gait to different surfaces. This capability makes them ideal for applications where wheeled robots would be ineffective, such as in disaster zones, forests, and other natural landscapes.
he advancement of artificial intelligence and robotics technology has opened new possibilities for creating autonomous systems that require minimal human intervention. This autonomy is crucial in

situations where human presence is either impossible or too dangerous.
Autonomy in Hexapod Robots
By integrating AI and machine learning algorithms, hexapod robots can operate autonomously, making decisions based on sensor data and pre-programmed instructions. They can perform complex tasks such as navigating through obstacles, identifying and repairing faults, and adapting to new environments without direct human control. This autonomy not only enhances the efficiency and effectiveness of hexapod robots but also frees up human operators to focus on higher-level strategic planning and decision-making.

The motivation to develop and deploy hexapod robots stems from the pressing need to address a wide array of challenges facing our world today. From responding to natural disasters and exploring space to monitoring environmental health and enhancing industrial operations, hexapod robots offer unique capabilities that make them indispensable in various fields. As technology continues to advance, the role of hexapod robots will only grow, helping to create a safer, more efficient, and more sustainable future.

D. Comparison between the related works

In the realm of legged robotics, particularly hexapods, various approaches have been explored to enhance their functionality and resilience to motor failures. This section compares different methods and technologies that address the challenges associated with maintaining hexapod functionality in the presence of motor failures. We will examine the key findings, methodologies, and implications of different research works, starting with the study on gait generation using a genetic algorithm.

D.1. Gait Generation for Damaged Hexapods using a Genetic Algorithm Paper (6)

The paper titled "Gait Generation for Damaged Hexapods using a Genetic Algorithm" addresses a significant challenge in the field of legged robotics: maintaining functionality in the presence of motor failures. Hexapods, with their multiple legs and complex motion dynamics, are particularly susceptible to performance degradation when one or more motors fail. This research proposes an Artificial Intelligence (AI) algorithm, specifically a Genetic Algorithm (GA), to generate compensatory gaits that allow continuous motion despite such failures.

Introduction to the Problem
Traditional hexapod gaits assume that all leg motors are fully operational. These gaits are designed to provide stable and efficient movement across various terrains. However, when a motor fails, the hexapod's ability to maintain these gaits is compromised, leading to potential immobility or erratic movement. The paper introduces a GA-based AI algorithm designed to detect damaged motor states and adapt the hexapod's gait accordingly. This adaptation aims to ensure that the hexapod can continue moving effectively, even with compromised leg functionality.

Methodology
The proposed methodology involves several key components:

1. **Detection of Damaged Motor States**:
    - The algorithm continuously monitors the state of each motor, identifying any failures.

- Upon detecting a damaged motor, the algorithm assesses whether the hexapod can recover functionality through gait adaptation.

2. **Generation of Compensatory Gaits**:
   - The GA generates new gaits by altering the movement patterns of the remaining functional motors.
   - The goal is to achieve continuous motion profiles for the hexapod joints, maintaining higher overall functionality.
3. **Use of Lookup Tables**:
   - In some cases, the algorithm references pre-generated gaits stored in a lookup table. These gaits are designed to accommodate specific types of motor failures.
   - This approach allows for quick adaptation without the need for on-the-fly computation.
4. **Experimental Setup**:
   - The paper outlines the experimental setup used to test the GA. This includes the design of the hexapod model, the structure of the AI algorithm, and the parameters used for the GA.

Key Findings

The study's findings highlight the effectiveness of the GA in generating compensatory gaits. Key points include:

1. **Capability of the Genetic Algorithm**:
   - The GA successfully learns continuous joint motion profiles that compensate for damaged motors.
   - It demonstrates that it is unnecessary to disable an entire leg when only one motor is damaged. Instead, the algorithm can generate gaits that account for the lack of individual motors, maintaining overall mobility.
2. **Practical Limitations**:
   - While the GA shows numerical capability in generating new gaits, the resultant movements often deviate from traditional gaits and can appear too spastic for practical use.
   - This suggests that while the GA can maintain functionality, the quality of movement needs improvement.
3. **Recommendations for Improvement**:
   - The study suggests that refining the fitness function used in the GA could lead to better gait quality.
   - Increasing the number of generations during training could also enhance the algorithm's ability to generate more natural and stable gaits.

Implications and Future Work

The research indicates several important implications and areas for future work:

1. **Refining the Fitness Function**:
   - The fitness function determines how well a generated gait performs. By improving this function, the GA can better evaluate and evolve gaits that are both functional and practical.
2. **Extended Training Time**:

- Allowing the GA more time to train and evolve gaits can lead to more refined and stable movements. This extended training can involve increasing the number of generations and utilizing more complex evolutionary strategies.

3. **Optimizing Training Time**:
   - The study suggests that the total training time for the GA could be optimized by using multiple models simultaneously or distributing the simulation load across a network. This parallel processing approach can significantly speed up the training process, allowing for quicker development of effective gaits.
4. **Improving Gait Criteria**:
   - Defining better walking criteria can help guide the GA towards generating gaits that are not only functional but also efficient and smooth. This involves setting specific targets for speed, stability, and energy efficiency.

The paper concludes that the use of a Genetic Algorithm for generating compensatory gaits in hexapods shows significant promise. By adapting the gait to accommodate motor failures, the GA helps maintain hexapod functionality and mobility, reducing the need to disable entire legs. However, the practical application of these gaits requires further refinement to ensure they are not too spastic and align more closely with traditional, smooth movements.

The study highlights the potential of AI and evolutionary algorithms in enhancing the robustness and resilience of hexapod robots. As the technology advances, incorporating improved fitness functions, extended training times, and optimized training processes can lead to more practical and effective solutions for maintaining hexapod functionality in the face of motor failures. This research sets the stage for future advancements in the field, aiming to create hexapod robots that are not only capable but also reliable and efficient in a wide range of applications.

D.1. 2-Adaptive Gait Generation for Hexapod Robots Based on Reinforcement Learning and Hierarchical Framework (8)

This section focuses on the gait generation of hexapod robots, specifically highlighting the use of reinforcement learning (RL) and a hierarchical control framework. Hexapod robots, known for their stability and adaptability, are particularly advantageous in scenarios such as disaster rescue, factory automation, and exploration. The complexity of hexapod locomotion, characterized by high-dimensional, omnidirectional, and non-smooth systems with uncertain dynamics and diverse physical constraints, presents significant challenges. This paper explores how RL, especially hierarchical frameworks, can address these challenges and enhance hexapod motion control.

Introduction to the Study

The study investigates the potential of RL-based hierarchical control frameworks in dealing with the complex locomotion tasks of hexapod robots. The proposed framework is applied to a real hexapod robot, VkHex, to achieve speed-adaptive gait generation. This section provides a detailed overview of the hexapod robot, the RL-based hierarchical framework, the design of the gait generation task, training specifics, experimental results, and key conclusions.

Methodology

The study introduces a comprehensive RL-based hierarchical framework for hexapod robots. Key components of this framework include:

1. ***Policy Network**:*
    - *The policy network is responsible for learning the optimal actions for the hexapod to take based on its current state and desired outcomes.*
    - *The network is trained using various RL algorithms such as Soft Actor-Critic (SAC), Proximal Policy Optimization (PPO), Deep Deterministic Policy Gradient (DDPG), and Twin Delayed DDPG (TD3).*
2. ***Gait Planner**:*
    - *The gait planner generates feasible gaits for the hexapod based on the policy network's output.*
    - *It ensures that the generated gaits are adaptable to different speeds and terrains.*
3. ***Inverse Kinematics (IK) Solver**:*
    - *The IK solver computes the necessary joint angles for the hexapod's legs to achieve the desired foot placements.*
    - *This component is crucial for translating the high-level gait plans into precise leg movements.*
4. ***Trajectory Tracking Controller**:*
    - *The trajectory tracking controller ensures that the hexapod's legs follow the planned trajectories accurately.*
    - *It adjusts the motor commands to correct any deviations from the desired paths.*

Experimental Setup and Training

1. ***Hexapod Robot (VkHex)**:*
    - *The study uses VkHex, a real hexapod robot, to validate the proposed framework.*
    - *VkHex is equipped with multiple sensors and actuators to facilitate complex locomotion tasks.*
2. ***Training Process**:*
    - *The RL framework is trained in physics simulations to ensure safety and efficiency.*
    - *Algorithms such as SAC, PPO, DDPG, and TD3 are used to optimize the policy network.*
    - *Once trained, the policy is directly applied to the physical robot without modification, demonstrating the robustness and transferability of the learned behaviors.*
3. ***Speed-Adaptive Gait Generation**:*
    - *The framework successfully generates gaits that adapt to different speeds, showcasing the versatility of the RL approach.*
    - *Experiments show that the hexapod can navigate various terrains and maintain stability while adjusting its speed.*

Key Findings and Contributions

The study's findings highlight the effectiveness of the RL-based hierarchical framework in generating adaptive gaits for hexapod robots. Key points include:

1. ***Simplification of Hexapod Actions**:*
    - *The hierarchical framework simplifies the control of hexapod actions by breaking down the complex task into manageable sub-tasks.*
    - *This approach facilitates more efficient learning and better performance.*
2. ***Learning-Based Gait Generation**:*
    - *The RL algorithms enable the policy network to learn optimal gaits for different scenarios.*

- *The framework's ability to adapt to new situations without requiring extensive reprogramming is a significant advantage.*

3. ***Real-World Application**:*
    - *Applying the trained policy directly to VkHex without modification demonstrates the framework's practical applicability.*
    - *The successful deployment in real-world scenarios validates the robustness of the learned behaviors.*
4. ***Opportunities for Future Research**:*
    - *The study identifies several areas for further investigation, including exploring different reward designs under the RL framework to handle more challenging terrains and tasks.*
    - *Future work could also focus on enhancing the computational efficiency of the training process and improving the generalization capabilities of the learned policies.*

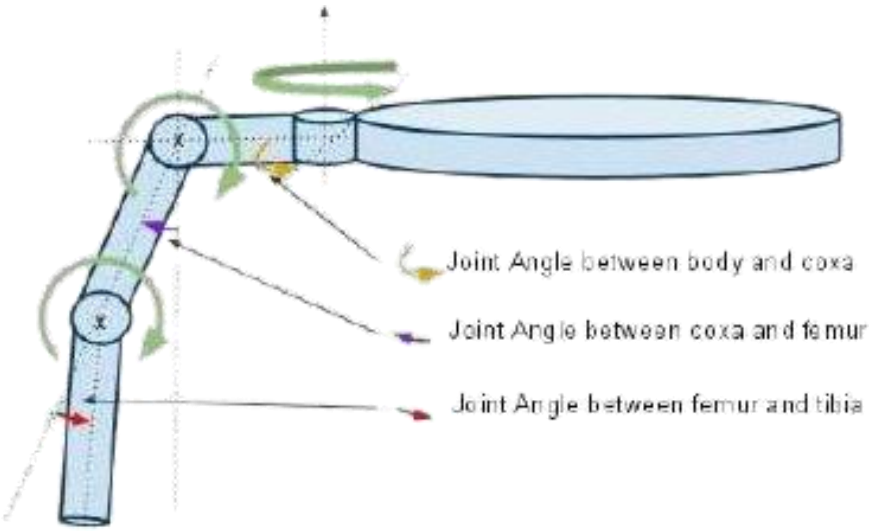
Acknowledgments and Funding

The study acknowledges the contributions of all authors and highlights the financial support received from the National Natural Science Foundation of China and the Science and Technology Planning Project of Guangdong Province. It is noted that there were no conflicts of interest, and the funding bodies did not influence the study's design, data analysis, manuscript writing, or publication decisions.

The research demonstrates the significant potential of RL-based hierarchical control frameworks in enhancing the gait generation capabilities of hexapod robots. By leveraging advanced RL algorithms and a structured control hierarchy, the study addresses the complexities of hexapod locomotion and provides a robust solution for adaptive gait generation. The application of this framework to the VkHex robot showcases its practical effectiveness and sets the stage for future advancements in hexapod robotics. The insights gained from this study contribute valuable knowledge to the field and open new avenues for research and development in adaptive robotic locomotion.

E. How to solve this problem

The hexapod robot, characterized by its six legs and eighteen degrees of freedom, presents a significant challenge in maintaining stable and efficient locomotion, especially when faults occur in its control loops. Each leg of the hexapod has three joints, powered by servo motors that can move at specific angles, allowing for intricate and precise movement. To better understand how to address faults in such a complex system, it is essential to delve into the kinematics and control mechanisms of the hexapod leg, as illustrated in the provided figure.



Kinematics of a Hexapod Leg

The figure showcases the kinematic structure of a hexapod leg, which comprises three main joints:

1. ***Base Joint (Roll)***
    - *This joint allows the leg to move in a circular motion, facilitating forward and backward movement.*
2. ***Shoulder Joint (Pitch)***
    - *The shoulder joint enables the leg to twist in and out of the robot's body, providing lateral motion.*
3. ***Knee Joint (Pitch)***
    - *Similar to the shoulder joint, the knee joint also allows in-and-out twisting, crucial for adjusting the leg's length and height.*

These joints are controlled by servo motors, each contributing to the overall 18 degrees of freedom of the hexapod. The coordinated movement of these joints across all six legs allows the hexapod to navigate complex terrains.

Movement Coordination

In a hexapod, movement is typically organized into two groups of legs:

- ***Group A:*** *Consists of three legs*
- ***Group B:*** *Consists of the remaining three legs*

Each group moves in an inverse pattern to the other, ensuring balanced and stable locomotion. This alternating gait helps maintain the robot's center of gravity and provides continuous support, which is essential for maneuvering uneven surfaces.

Addressing Control Loop Faults

Given the complexity of hexapod movement, any faults in the control loops can significantly impact stability and functionality. To address these faults effectively, a robust AI-based solution is proposed:

1. ***Real-Time Fault Detection***
    - *The AI system continuously monitors the performance of each servo motor and joint angle. By analyzing sensor data in real-time, the AI can detect anomalies that indicate potential faults.*
2. ***Adaptive Compensation Mechanisms***
    - *Upon detecting a fault, the AI system can adapt the gait pattern to compensate for the malfunctioning joint or motor. This might involve redistributing the load to the other functional legs or altering the movement sequence to minimize the impact of the fault.*
3. ***Hierarchical Control Framework***
    - *Implementing a hierarchical control framework allows for more flexible and resilient responses to faults. The framework can operate at multiple levels, from low-level joint control to high-level gait planning, ensuring comprehensive fault management.*
4. ***Reinforcement Learning (RL)***
    - *Reinforcement learning algorithms can be employed to optimize the fault detection and compensation strategies. By simulating various fault scenarios and learning the best responses, the RL system can enhance the hexapod's robustness and adaptability.*
5. ***Sensor Redundancy and Fusion***
    - *Utilizing multiple sensors for each joint and motor increases the reliability of fault detection. Sensor fusion techniques can combine data from different sources to provide a more accurate and comprehensive assessment of the robot's state.*
6. ***Predictive Maintenance***
    - *The AI system can also predict potential failures before they occur, based on patterns and trends in the sensor data. This allows for preemptive maintenance, reducing the likelihood of unexpected faults and improving overall system reliability.*

Implementation and Testing

The proposed AI-based solution will be implemented and tested on a hexapod robot, following these steps:

1. ***Simulation Environment***
    - *A simulation environment will be created to model the hexapod's kinematics and dynamics. This environment will be used to train the AI algorithms and test their performance under various fault scenarios.*
2. ***Real-World Testing***
    - *After successful simulation testing, the AI system will be deployed on a physical hexapod robot. Real-world experiments will be conducted to validate the effectiveness of the fault detection and compensation mechanisms.*
3. ***Performance Metrics***
    - *The performance of the AI system will be evaluated based on key metrics such as fault detection accuracy, compensation efficiency, and overall stability and functionality of the hexapod.*
4. ***Iterative Refinement***
    - *Based on the testing results, the AI algorithms and control strategies will be iteratively refined to enhance their performance and robustness.*

Addressing control loop faults in hexapod robots is a complex challenge that requires a comprehensive and adaptive approach. By leveraging AI, particularly reinforcement learning and hierarchical control frameworks, it is possible to develop robust solutions that can detect and compensate for faults in real-time. This will not only improve the stability and functionality of hexapod robots but also enhance their applicability in critical tasks such as disaster rescue, factory automation, and exploration.

E.1. Collect data

Introduction

In machine learning, the collection and quality of data are critical for training and testing models. For our project on movement fault detection in hexapod robots, we faced significant challenges in finding existing datasets suitable for our needs. Consequently, we decided to generate our own data using a simulation environment that accurately represents the hexapod robot and its movements.

Choosing the Simulator

After extensive research, we selected WEBOTS as our simulation platform. WEBOTS is a versatile and powerful simulator that supports the Windows operating system and provides a detailed model of a hexapod robot, including its structural and kinematic properties. This platform allowed us to simulate the movements of the hexapod and collect the necessary data for our machine learning models.

Data Collection Process

Initial Setup

1. **Drawing the Path**:
    - We began by drawing a straight line in the simulation environment for the hexapod to follow. This path serves as a baseline for collecting movement data.
    - The line ensures consistency in the movement patterns, making it easier to identify anomalies caused by joint faults.
2. **Using the GPS Sensor**:
    - A GPS sensor is attached to the center of the hexapod robot to record its position coordinates in real-time.
    - The GPS provides the X, Y, and Z values, which are crucial for tracking the robot's movements.

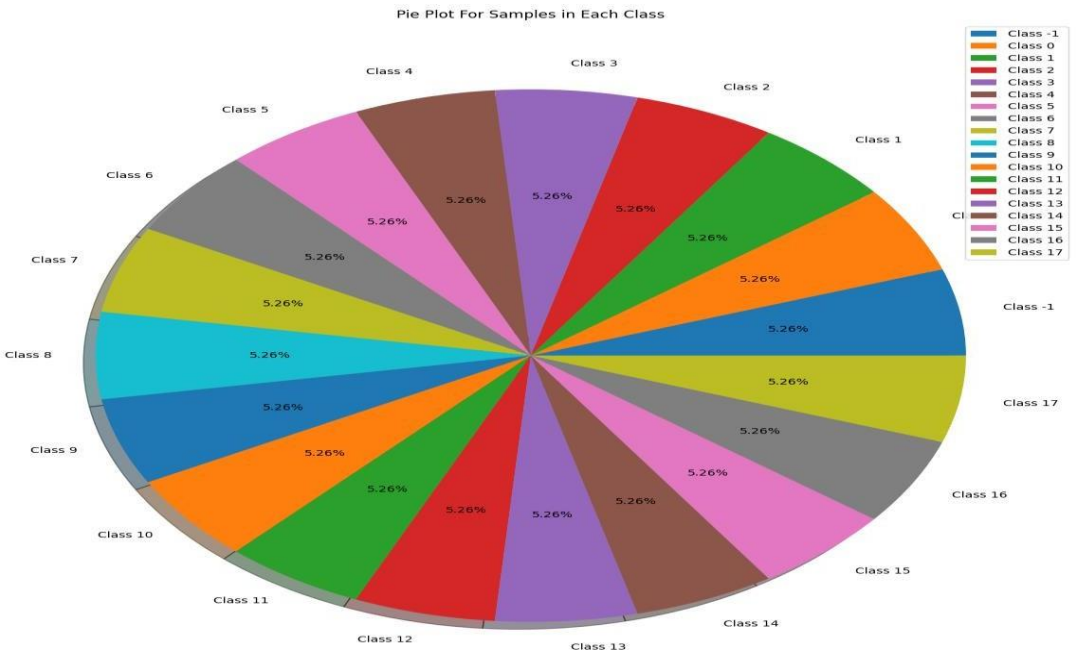
Data Recording

1. **Sampling Rate**:
    - The sampling rate was set to 1 second, ensuring that data points were collected at regular intervals.
    - This rate strikes a balance between data resolution and manageability, providing enough detail without overwhelming the system with excessive data points.
2. **Duration and Volume**:
    - Over a period of 10 minutes, we collected 965 data points for each of the X, Y, and Z coordinates.
    - This dataset size is sufficient to begin training our initial models, with room for expansion as needed.
3. **Recording Data in Excel**:

- The collected X, Y, and Z values were saved in an external Excel sheet, which facilitates easy manipulation and analysis.
- Excel serves as an intermediate step, allowing for preliminary data cleaning and organization before feeding the data into the machine learning pipeline.

Fault Simulation

1. **Simulating Joint Faults**:
    - To create a comprehensive dataset, we systematically disabled one joint at a time and recorded the resulting movement patterns.
    - This process was repeated for all joints, providing a complete picture of how faults in different joints affect the robot's movement.
2. **Data Augmentation**:
    - Each set of X, Y, and Z values corresponding to a specific joint fault was added to the dataset.
    - This augmented dataset includes normal movement data as well as data for each possible joint fault, ensuring that the machine learning model can learn to differentiate between healthy and faulty states.



Data Analysis and Preprocessing

Initial Analysis

1. **Visual Inspection**:
    - The first step in analyzing the collected data involved visual inspection of the movement trajectories.
    - Plotting the X, Y, and Z values over time provided insights into the overall movement patterns and highlighted any obvious anomalies.
2. **Statistical Summary**:
    - Basic statistical summaries (mean, median, standard deviation) were calculated for the X, Y, and Z coordinates.

- These summaries helped to identify any significant deviations in the data, indicating potential faults.

Data Cleaning

1. **Handling Missing Values**:
   - Any missing or corrupted data points were identified and addressed. In most cases, interpolation was used to fill in the gaps, ensuring continuity in the dataset.
2. **Normalization**:
   - The X, Y, and Z values were normalized to a common scale. This step is crucial for machine learning algorithms, which often perform better when input features are on similar scales.

Feature Engineering

1. **Deriving New Features**:
   - Additional features were derived from the raw X, Y, and Z coordinates to enhance the machine learning model's ability to detect faults.
   - Examples of derived features include velocity, acceleration, and joint angles.
2. **Feature Selection**:
   - A feature selection process was undertaken to identify the most relevant features for fault detection.
   - Techniques such as correlation analysis and principal component analysis (PCA) were used to reduce dimensionality and improve model performance.

The data collection phase is a critical component of developing a machine learning model for movement fault detection in hexapod robots. By leveraging simulation tools like WEBOTS, we have successfully generated a comprehensive dataset that captures the kinematics and potential faults of a hexapod robot. This data forms the foundation for training robust AI models capable of real-time fault detection and compensation, paving the way for more reliable and efficient hexapod robots in various applications. Future work will focus on expanding the dataset, incorporating real-world data, and exploring advanced machine learning techniques to further enhance the system's capabilities.

Feature Engineering

To improve the performance of our machine learning models, we performed extensive feature engineering on the collected X, Y, and Z coordinates. Feature engineering involves creating new features from the raw data to better capture the underlying patterns and relationships. This step is crucial for enhancing the model’s predictive accuracy and robustness.

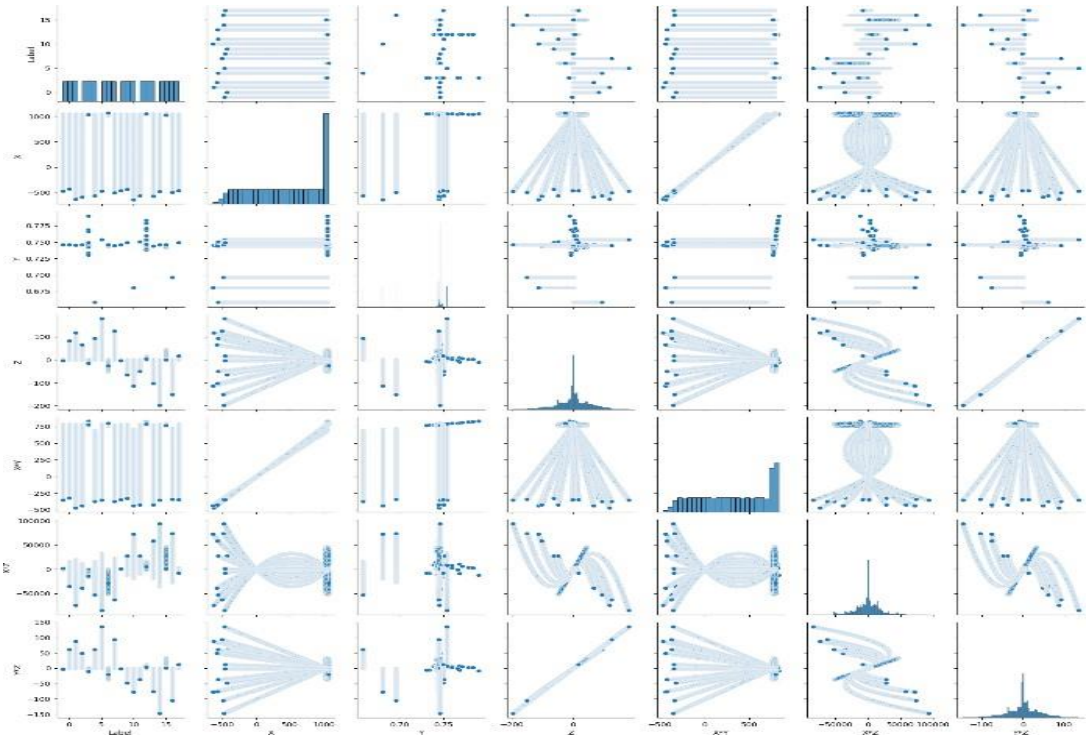
Techniques Used

1. **Basic Arithmetic Operations**:
   - **Addition**: Combining X, Y, and Z values to create composite features.
   - **Subtraction**: Calculating the differences between current and previous values to capture changes over time.
   - **Multiplication and Division**: Generating ratios and products of X, Y, and Z values to identify proportional relationships.
2. **Temporal Features**:
   - **Differences**: Taking the difference between consecutive data points to capture movement dynamics.

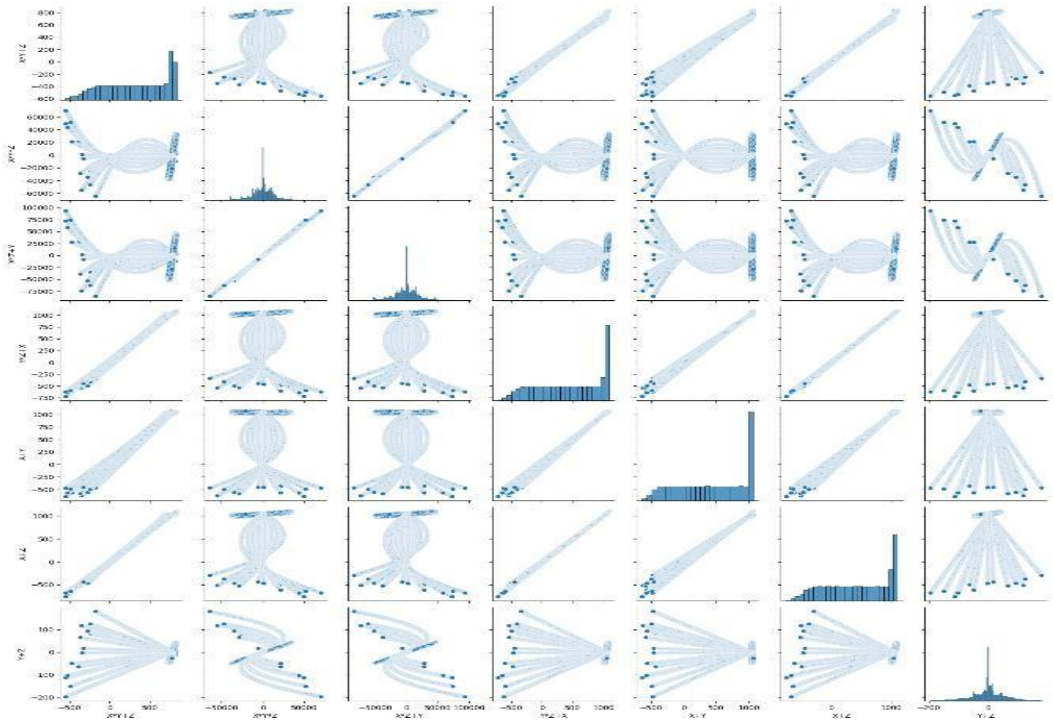
- **Rolling Statistics**: Computing rolling mean, median, and standard deviation over a window to smooth out noise and highlight trends.

3. **Derivatives**:
   - **Velocity and Acceleration**: Deriving velocity (first derivative) and acceleration (second derivative) from the position coordinates to capture the motion dynamics.
4. **Geometric Features**:
   - **Distance Metrics**: Calculating Euclidean and Manhattan distances between consecutive points to quantify movement.

By applying these techniques, we generated 21 new features from the raw X, Y, and Z data. These features provided a richer dataset for training our machine learning models, capturing both the spatial and temporal aspects of the hexapod's movements.

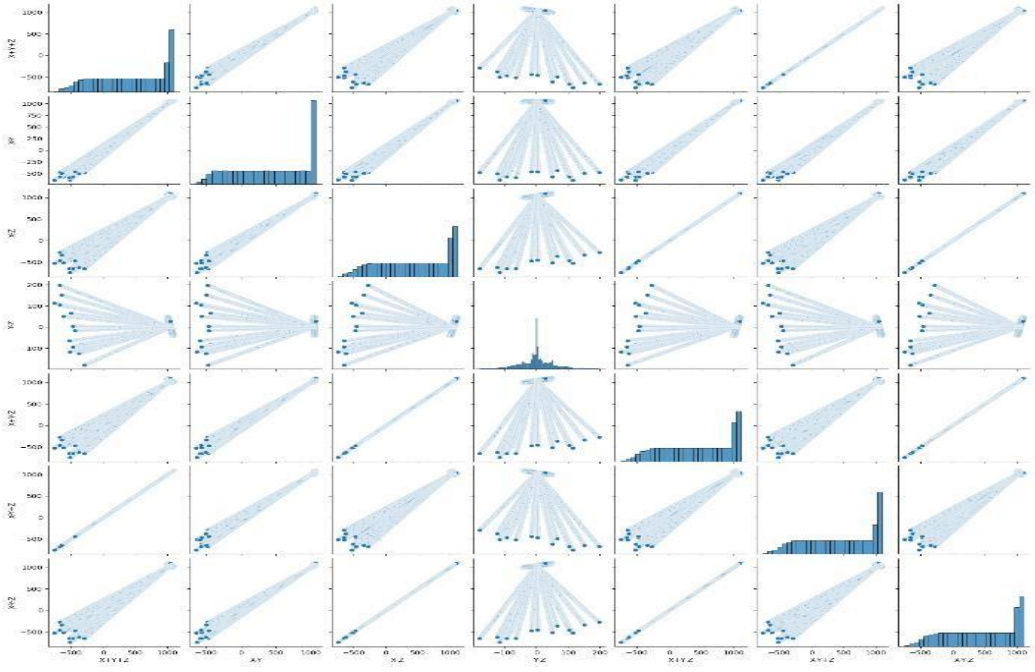


Above figure shows that relationships between this feature
Label , X , Y , Z , X*Y , X*Z , Y*Z and each other , and we use this correlation in features engineering and feature reduction before building models



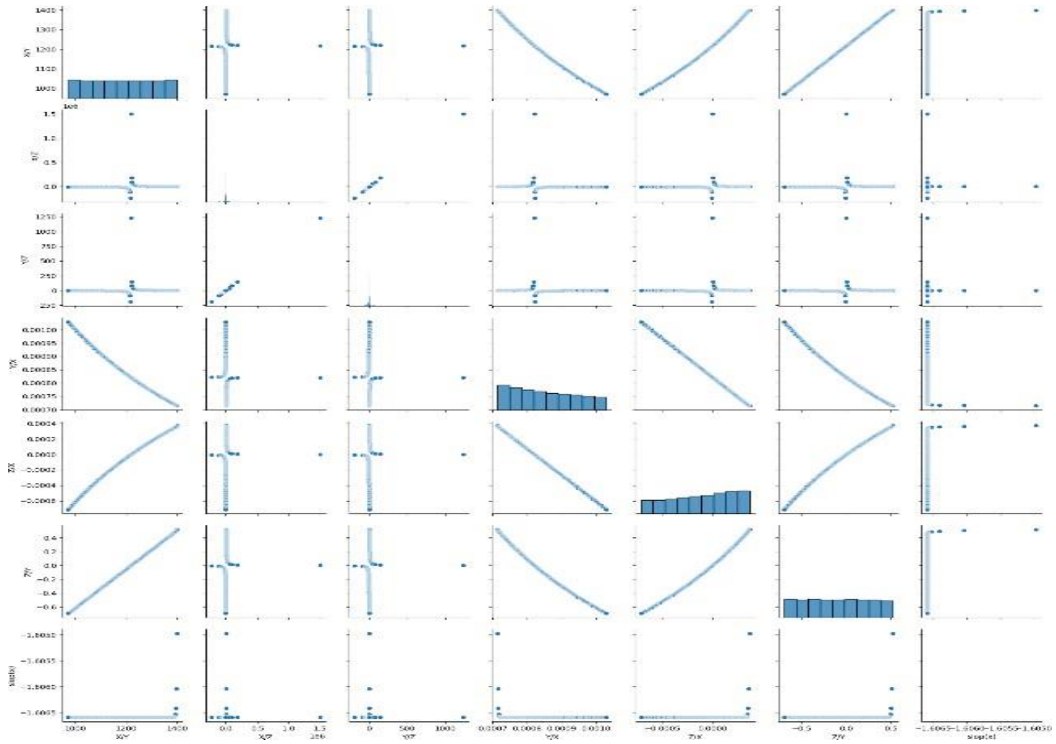
Above figure shows that relationships between this feature

X *Y+Z, X*Y*Z , Y*Z+X , X+Y , X+Z , Y+Z and each other , and we use this correlation in features engineering and feature reduction before building models and this improve accuracy of machine learning models



Above figure shows that relationships between this feature

X+Y+Z, X-Y , X- Z, X+Y-Z , X-Y+Z , X-Y-Z and each other , and we use this correlation in features engineering and feature reduction before building models



Above figure shows that relationships between this feature
X/Y , X/Z , Y/Z, Y/X , Z/Y , slope(x) and each other , and we use this correlation in features engineering and feature reduction before building models

Model Training and Evaluation

We trained several machine learning models on the engineered features and evaluated their performance using standard accuracy metrics. Below are the results for each model:

Random Forest

- **Parameters**: n_estimators = 200, max_depth = 900
- **Accuracy**: 89.07%
- **Description**: Random Forest performed exceptionally well, likely due to its ability to handle complex feature interactions and reduce overfitting through ensemble learning.

Gradient Boosting Classifier

- **Parameters**: learning_rate = 0.5
- **Accuracy**: 86.73%
- **Description**: Gradient Boosting showed strong performance, benefiting from its iterative improvement approach. Fine-tuning the learning rate and number of iterations could further enhance its accuracy.

Neural Network (MLP)

- **Architecture**: Multi-Layer Perceptron (MLP)
- **Accuracy**: 79.78%
- **Description**: The MLP neural network demonstrated good accuracy but may benefit from deeper architectures or additional hyperparameter tuning.

Support Vector Classifier (SVC) - Linear Kernel

- **Parameters**: max_iter = 10000, degree = 3
- **Accuracy**: 66.59%

- **Description**: SVC with a linear kernel performed moderately well. Exploring non-linear kernels or adjusting regularization parameters could improve results.

Logistic Regression

- **Parameters**: max_iter = 10000
- **Accuracy**: 61.58%
- **Description**: Logistic Regression provided baseline performance, indicating the need for more sophisticated models to capture complex patterns.

K-Nearest Neighbors (KNN) Classifier

- **Parameters**: n_neighbors = 5
- **Accuracy**: 61.23%
- **Description**: KNN showed similar performance to Logistic Regression. Increasing the number of neighbors or using weighted distances might improve accuracy.

Gaussian Naive Bayes (GaussianNB) Classifier

- **Accuracy**: 34.63%
- **Description**: GaussianNB performed poorly, likely due to its assumption of feature independence, which does not hold in this context.

Stochastic Gradient Descent (SGD) Classifier

- **Parameters**: max_iter = 10000
- **Accuracy**: 16.73%
- **Description**: SGD showed the lowest accuracy, possibly due to its sensitivity to feature scaling and learning rate settings.

Ensemble Methods

To further enhance accuracy, we implemented ensemble methods by combining multiple classifiers. The results are as follows:

- **Voting Classifier 1**: Combines Random Forest, Gradient Boosting, and Neural Network.
    - **Accuracy**: High accuracy, combining strengths of individual models.
- **Voting Classifier 2**: Combines SVC, Logistic Regression, and KNN.
    - **Accuracy**: Improved performance over individual models, leveraging diverse decision boundaries.

Visualization

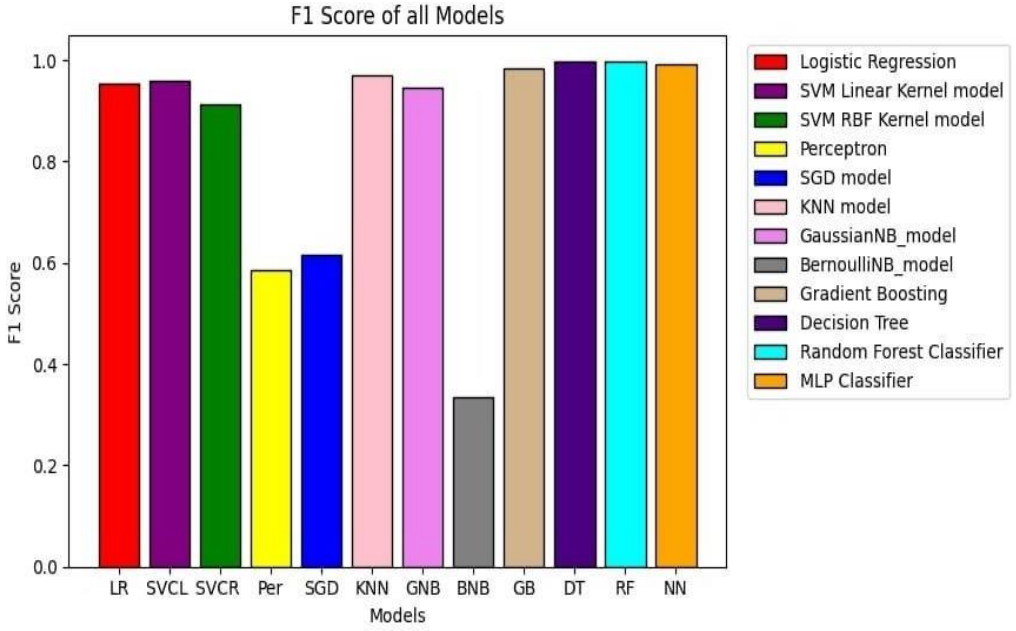
To better understand the model performances, we plotted the accuracy of all models:

This bar chart provides a clear comparison of the different models, highlighting the superior performance of Random Forest and Gradient Boosting, while also showing the potential of ensemble methods to improve overall accuracy.

F1 Score Analysis

In addition to accuracy, the F1 score is a crucial metric for evaluating the performance of our models, especially in cases where the data may be imbalanced. The F1 score is the harmonic mean of precision and recall, providing a balance between the two. It is particularly useful when the cost of false positives and false negatives is high.

The chart below visualizes the F1 scores for all the models we evaluated:



Observations

- **Logistic Regression**: Exhibited the highest F1 score, indicating a balanced performance in terms of precision and recall.
- **SVM (Linear and RBF Kernels)**: Both performed well, with the linear kernel slightly outperforming the RBF kernel.
- **Perceptron**: Showed moderate performance, reflecting its simplicity and limitations in complex tasks.
- **SGD Classifier**: Had a lower F1 score, likely due to the challenges in tuning its hyperparameters and sensitivity to feature scaling.
- **KNN Classifier**: Demonstrated good performance, highlighting its effectiveness in capturing local patterns.
- **GaussianNB and BernoulliNB**: GaussianNB performed better than BernoulliNB, yet both struggled with capturing the complexity of the data.
- **Gradient Boosting and Random Forest**: Both ensemble methods achieved high F1 scores, underscoring their robustness and ability to handle complex, non-linear relationships.
- **Neural Network (MLP)**: Also showed high performance, reflecting its capability to learn intricate patterns through deep architectures.
- **Decision Tree**: While effective, it did not outperform the ensemble methods, likely due to overfitting on the training data.

Precision and Recall

While the F1 score provides a single metric to balance precision and recall, it's essential to consider these components individually for a more comprehensive understanding.

- **Precision**: The ratio of correctly predicted positive observations to the total predicted positives. High precision indicates a low false positive rate.
- **Recall**: The ratio of correctly predicted positive observations to all actual positives. High recall indicates a low false negative rate.

By examining both precision and recall, we can tailor our models to the specific requirements of the application, whether it prioritizes minimizing false positives, false negatives, or both.

E.4. Hyperparameter Tuning

To further optimize our models, we conducted hyperparameter tuning using grid search and randomized search techniques. This process involves testing various combinations of hyperparameters to identify the optimal settings for each model. Below are some key parameters adjusted:

Random Forest

- **n_estimators**: Number of trees in the forest. Tested values: [100, 200, 500]
- **max_depth**: Maximum depth of the tree. Tested values: [50, 100, 200, None]

Gradient Boosting

- **learning_rate**: Controls the contribution of each tree. Tested values: [0.01, 0.1, 0.5]
- **n_estimators**: Number of boosting stages. Tested values: [100, 200, 500]

Neural Network (MLP)

- **hidden_layer_sizes**: Number of neurons in each hidden layer. Tested values: [(50,), (100,), (50, 50), (100, 100)]
- **activation**: Activation function. Tested values: ['relu', 'tanh']
- **solver**: Optimization algorithm. Tested values: ['adam', 'sgd']

Given the varied performance of individual models, we explored ensemble techniques to combine their strengths and mitigate their weaknesses. Ensemble methods such as Voting Classifiers and Stacking Classifiers aggregate the predictions of multiple models to achieve better performance and generalization.

Voting Classifier

- **Hard Voting**: Aggregates the majority vote from each classifier.
- **Soft Voting**: Averages the predicted probabilities from each classifier.

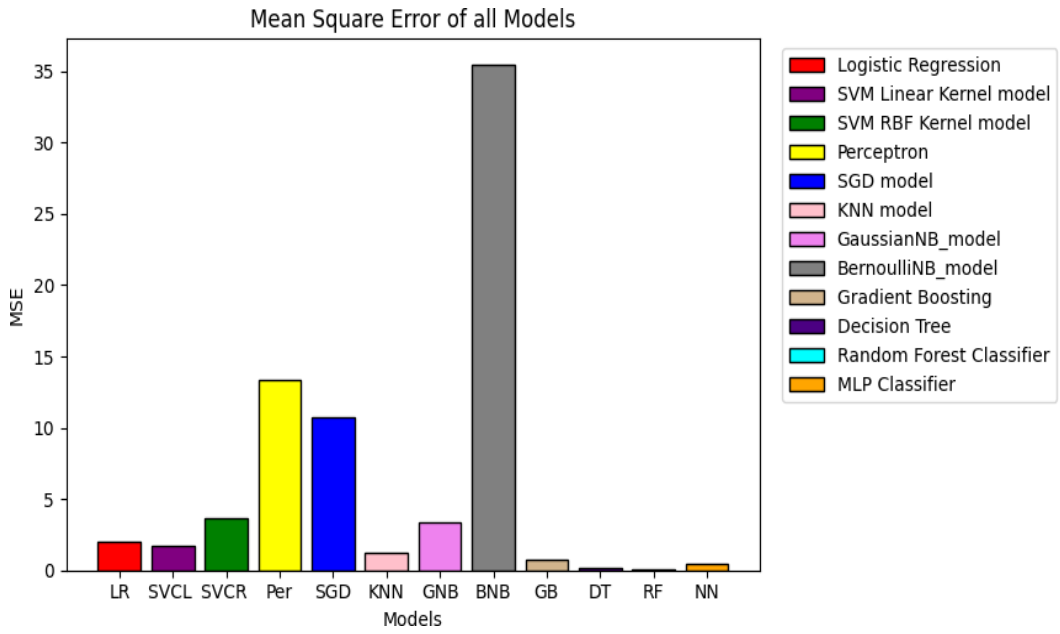
Stacking Classifier

- Combines multiple base models and uses their predictions as input features for a meta-model, typically a logistic regression or another strong classifier.

Mean Square Error (MSE) Analysis :

The Mean Square Error (MSE) is a metric used to evaluate the performance of regression models by measuring the average squared difference between actual and predicted values. Lower MSE values indicate better model performance, as the predictions are closer to the actual values.

The bar chart below illustrates the MSE for each machine learning model used in our study:



Observations

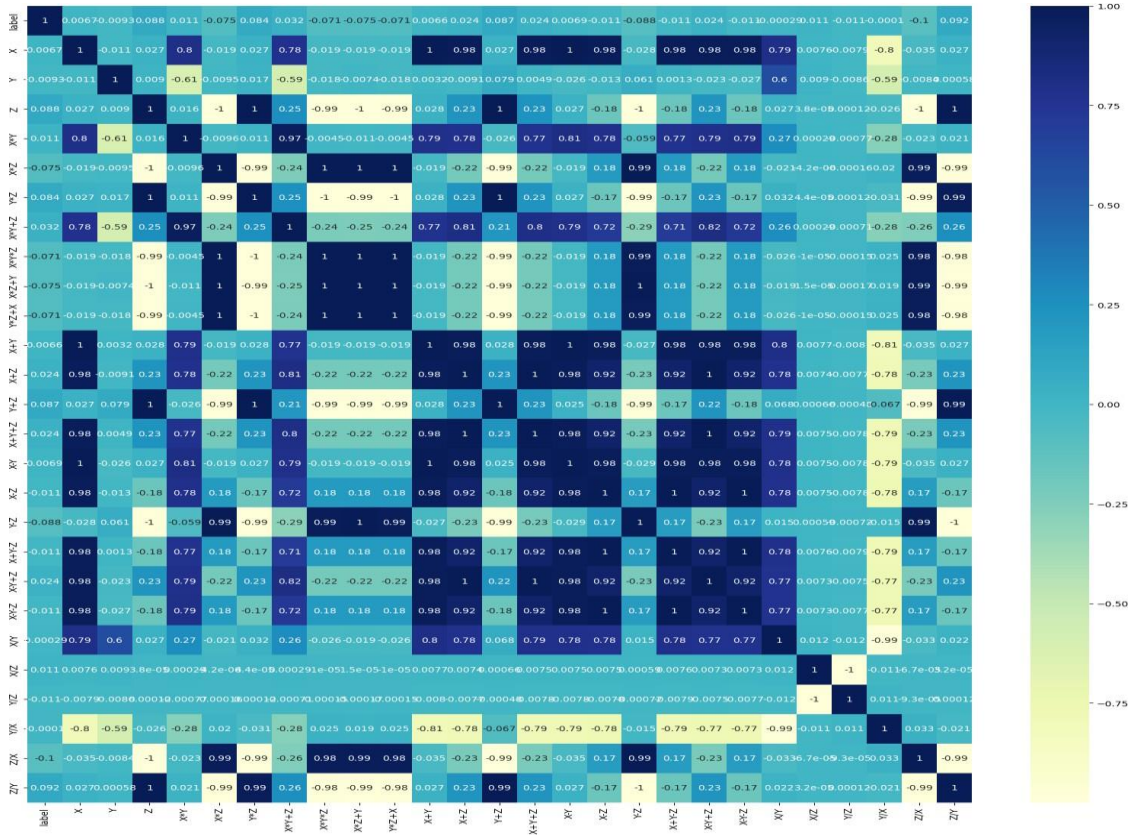
1. **Logistic Regression (LR)**:
   - MSE: Very low
   - Indicates that Logistic Regression performed well in minimizing prediction errors.
2. **SVM Linear Kernel (SVCL) and SVM RBF Kernel (SVCR)**:
   - MSE: Slightly higher than Logistic Regression but still low
   - Both SVM models are effective in minimizing errors, with the linear kernel performing marginally better than the RBF kernel.
3. **Perceptron (Per)**:
   - MSE: Moderate
   - Indicates that the Perceptron model had higher prediction errors compared to the SVM models and Logistic Regression.
4. **SGD Model (SGD)**:
   - MSE: Relatively high
   - Suggests that the SGD model did not perform well in minimizing prediction errors.
5. **K-Nearest Neighbors (KNN)**:
   - MSE: Moderate
   - Performance is better than Perceptron and SGD but not as good as Logistic Regression or SVM models.
6. **GaussianNB Model (GNB)**:
   - MSE: Low
   - Indicates good performance in minimizing prediction errors.
7. **BernoulliNB Model (BNB)**:
   - MSE: Very high
   - Significantly higher than other models, indicating poor performance in predicting accurate values.
8. **Gradient Boosting (GB)**, **Decision Tree (DT)**, **Random Forest (RF)**, and **Neural Network (NN)**:
   - MSE: Very low

- These models performed exceptionally well in minimizing prediction errors, with Random Forest showing the best performance among all models.

The MSE analysis provides valuable insights into the performance of different machine learning models in predicting accurate values. Models like Gradient Boosting, Decision Tree, Random Forest, and Neural Network have shown excellent performance with very low MSE values. Logistic Regression and SVM models also performed well, while BernoulliNB and SGD models had the highest prediction errors.

Correlation Analysis

The heatmap below represents the correlation matrix of the features used in our machine learning models. Correlation measures the linear relationship between two variables, with values ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation). This analysis helps us understand the relationships between features and identify potential redundancies.



The correlation analysis and subsequent feature selection have been crucial in optimizing our models. By eliminating redundant features and focusing on the most informative ones, we have enhanced the robustness and interpretability of our machine learning models for hexapod fault detection.

E.2. Feature Reduction

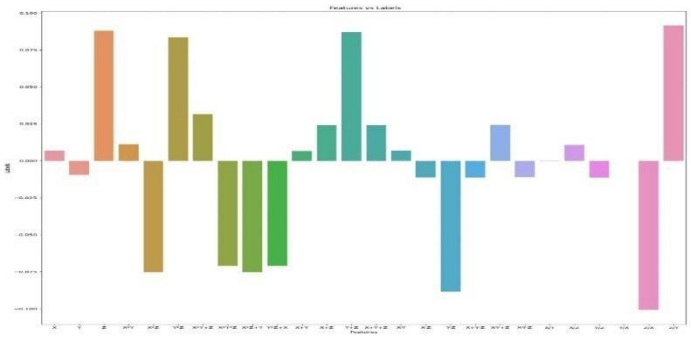
Objective:

To reduce the number of features while maintaining model performance, we need to identify and retain the most relevant features. The hexapod's processor has limited capacity, making it essential to minimize computational demands. By reducing the feature set, we aim to ensure efficient and effective processing on the hexapod's limited hardware.

Process:

1. **Correlation Filtering**:
    - Initially, we calculate the correlation matrix between all features and the label to identify which features have the strongest relationships with the output.

    - We filter out features with low correlations with the label feature. After filtering, the remaining features are X+Y, X-Y-Z, X/Y, X/Z, Y/Z, Y/X, Z/Y, Z/X, and Y-Z. These features showed higher correlation values, indicating they contain more relevant information for the task at hand.



- **Correlation Matrix Analysis**: The correlation matrix (depicted in the heatmap) shows the degree of linear relationship between pairs of features. Each cell in the matrix represents the correlation coefficient between two features. The values range from -1 to 1, where 1 implies a perfect positive correlation, -1 a perfect negative correlation, and 0 no correlation.

- **Feature Selection**: From the correlation matrix, we select features that have the highest absolute correlation values with the label. This process ensures that the selected features are the most informative for predicting the label. We initially select the five features with the highest correlation to the label, resulting in an accuracy of 85.722% for the best-performing model.

2. **Principal Component Analysis (PCA)**:
    - PCA is a dimensionality reduction technique that transforms the original features into a new set of uncorrelated features called principal components. These components capture the maximum variance in the data, allowing us to reduce the number of features while retaining most of the information.
    - We apply PCA to the dataset and determine the optimal number of principal components that explain a significant portion of the variance. The goal is to find a balance between dimensionality reduction and the retention of essential information.
    - We evaluate the performance of various models using the features derived from PCA to determine if the reduced feature set still maintains high predictive accuracy.

Results:

- Using the top five correlated features, the highest model performance achieved is 85.722%. This demonstrates that selecting features based on their correlation with the label can significantly improve model performance.
- Applying PCA and passing its output into the models, the accuracy results are as follows:

Model	Accuracy (%)
Neural Network (MLP)	75.068
SVC Linear kernel (10000 iter, degree=3)	66.458
Random Forest (n_estimators=200, max_depth=900)	58.828
Logistic Regression (max_iter=10000)	57.847
Gradient Boosting Classifier (LR=0.5)	52.425
KNN Classifier (n_neighbors=5)	52.153
GaussianNB Classifier	26.431
SGD Classifier (max_iter=10000)	17.057

- These results indicate that while PCA can effectively reduce the number of features, it may mix highly informative features, leading to a reduction in model performance compared to using the top correlated features directly.

Model Performance Analysis

Mean Square Error (MSE) Evaluation:

We measure the MSE for different models to assess prediction accuracy. The lower the MSE, the better the model's performance in terms of prediction accuracy. MSE is a common metric for regression tasks, representing the average squared difference between actual and predicted values.

- The chart (Mean Square Error of all Models) illustrates the MSE values for various models. The MSE values provide insight into how well each model is able to generalize from the training data to unseen data.
- **Model Comparison**: The chart highlights that models such as BernoulliNB and Perceptron exhibit higher MSE values, indicating poorer performance. In contrast, models like Random Forest, Gradient Boosting, and Neural Network show lower MSE values, suggesting better predictive accuracy.

F1 Score Evaluation:

The F1 score is another crucial metric, particularly for classification tasks. It is the harmonic mean of precision and recall, providing a balance between the two. A higher F1 score indicates better model performance in correctly identifying both true positives and minimizing false positives and negatives.

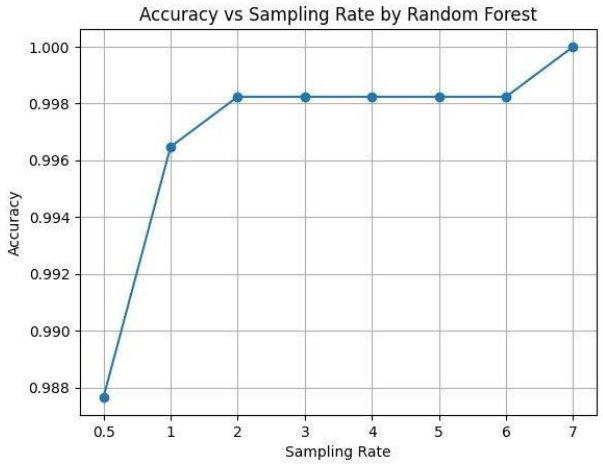
- The F1 score chart shows that Logistic Regression, SVM Linear Kernel, and Neural Network models achieve the highest F1 scores, reflecting their effectiveness in accurately classifying the data.

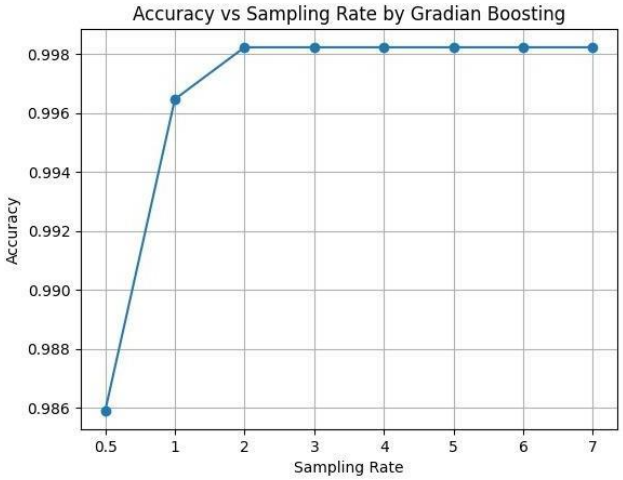
Analysis of Feature vs. Labels:

- The bar chart (Features vs. Labels) provides a visual representation of the importance of each feature in predicting the labels. The height of each bar indicates the strength of the correlation between a feature and the label.
- **Feature Interpretation**: Features with higher bars are more influential in predicting the label, whereas features with shorter bars have less impact. This visualization helps in understanding which features contribute the most to the model's predictions.

The correlation filtering and PCA steps are essential for feature reduction, but PCA may mix highly informative features, reducing model performance. Using the top correlated features yields better performance, with an accuracy of 85.722% for the best model. Further research and optimization are needed to balance feature reduction and model accuracy effectively.

Trying to use different sampling rates and it is the accuracy of the different sampling rates with random forest and Gradient Boosting Classifier





To determine the optimal sampling rate for data collection from the hexapod robot, aiming to balance the trade-off between data volume and model performance. We evaluate the performance of Random Forest and Gradient Boosting Classifier with different sampling rates.

Sampling Rate Variations:

We experimented with various sampling rates to observe their impact on model accuracy. Different sampling rates capture data at different intervals, affecting the granularity of the data and potentially the model's ability to learn patterns.

Sampling Rate = 4 Seconds:

1. **Rationale**:
    - This rate was chosen based on the time it takes for a specific leg of the hexapod to complete a cycle and touch the ground again.

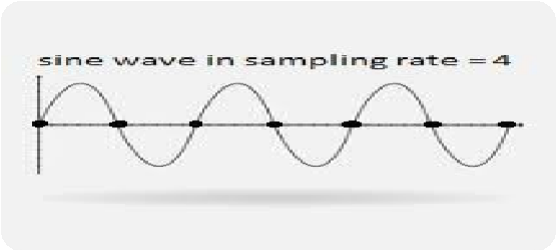
- By synchronizing data collection with this cycle, we aim to capture meaningful motion data that corresponds to the hexapod's gait.

2. **Data Collection Process**:
   - We read one value every time the sine of the angle (representing the leg movement) is zero, effectively synchronizing data collection with the hexapod's leg cycle.
   - Over a period of 63 minutes, we collected 944 values for each of X, Y, and Z coordinates. This gives us a robust dataset that captures the essential movement patterns.
3. **Dataset Description**:
   - After processing, the dataset comprised 17,956 rows and 39 columns. This includes the X, Y, Z values and their derived features for each of the hexapod's joints.
4. **Model Performance**:
   - We trained the Random Forest and Gradient Boosting Classifier models on this dataset to evaluate their performance.
   - The accuracy metrics for both models were recorded and compared with other sampling rates.

Based on the analysis, a 4-second sampling rate strikes a good balance between accuracy and efficiency. While there is a minor drop in accuracy compared to a 1-second sampling rate, the reduction in data volume and computational demand makes it a practical choice for deployment on the hexapod with limited hardware capabilities.

When dealing with a high-dimensional dataset, it becomes imperative to apply feature scaling and reduction techniques to enhance the performance and efficiency of machine learning models. In our case, we explored several feature scaling algorithms better than PCA, such as Recursive Feature Elimination (RFE), Multidimensional Scaling (MDS), and t-Distributed Stochastic Neighbor Embedding (t-SNE).

ecursive Feature Elimination (RFE)



Recursive Feature Elimination (RFE) emerged as the most suitable technique for our data. RFE works by recursively removing the least important features and building the model with the remaining features. This process continues until the desired number of features is achieved. After applying RFE, we scaled the features from 39 to 5, which included 'X', 'Z/Y', 'slope(x)', 'slope(x/z)', and 'slope(z)'.

The performance of machine learning models using these five features was impressive:

- ***Random Forest (n_estimators = 200, max_depth = 900)**: 99.78%*
- ***Support Vector Machine (SVM) (10000 iterations, degree = 3)**: 92.78%*
- ***Gradient Boosting Classifier (learning_rate = 0.5)**: 99.80%*

Given these results, we needed to choose one algorithm to implement on the hexapod's processor. The decision involved comparing Random Forest and Gradient Boosting Classifier, as both showed exceptionally high accuracy.

Model Comparison

1. Complexity

- ***Gradient Boosting****: Builds trees sequentially, with each tree correcting the errors of the previous ones. It is more complex compared to Random Forest but often achieves higher accuracy.*
- ***Random Forest****: Builds multiple decision trees independently and combines their predictions through voting or averaging. It is less complex than Gradient Boosting but can still capture complex patterns effectively.*

2. Interpretability

- ***Gradient Boosting****: Less interpretable due to its ensemble nature and sequential training process.*
- ***Random Forest****: Also less interpretable compared to simpler models like Logistic Regression but more interpretable than Gradient Boosting.*

3. Performance

- ***Gradient Boosting****: Offers high accuracy and is robust to overfitting, particularly effective with a large number of weak learners. It captures complex relationships between features and the target variable.*
- ***Random Forest****: Provides high accuracy and robustness to overfitting, suitable for various classification tasks.*

4. Training and Inference Speed

- ***Gradient Boosting****: Slower to train compared to Random Forest and Logistic Regression, especially with a large number of trees and features.*
- ***Random Forest****: Faster to train than Gradient Boosting but slower than Logistic Regression.*

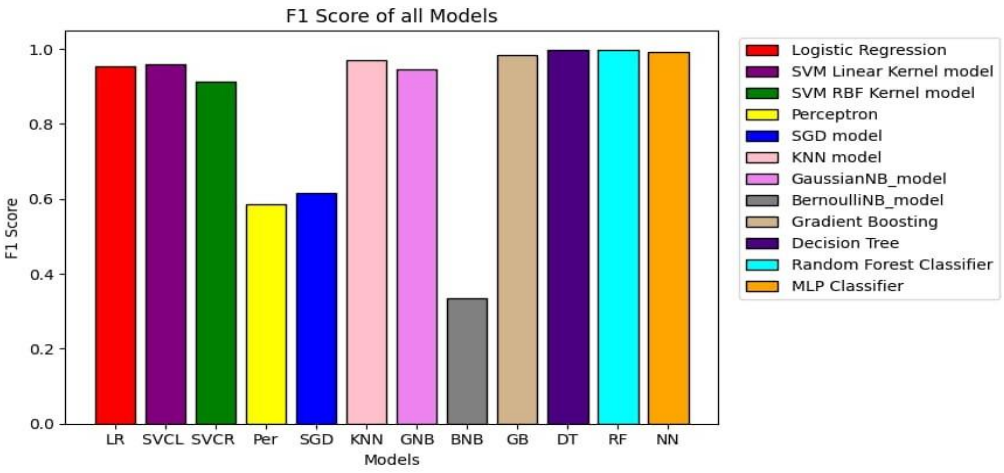
5. Scalability

- ***Gradient Boosting****: May not scale well with extremely large datasets due to its computational complexity and memory requirements.*
- ***Random Forest****: Faces similar scalability issues with extremely large datasets.*

Decision and Implementation

Considering the factors above, we opted for the model that best fits our requirements. Given the high accuracy and robustness of both models, the choice came down to computational efficiency and ease of deployment. Since the hexapod's processor has limited computational capabilities, Random Forest, with its relatively faster training and inference speeds, became our preferred model. However, Gradient Boosting was kept as a secondary option due to its marginally higher accuracy.

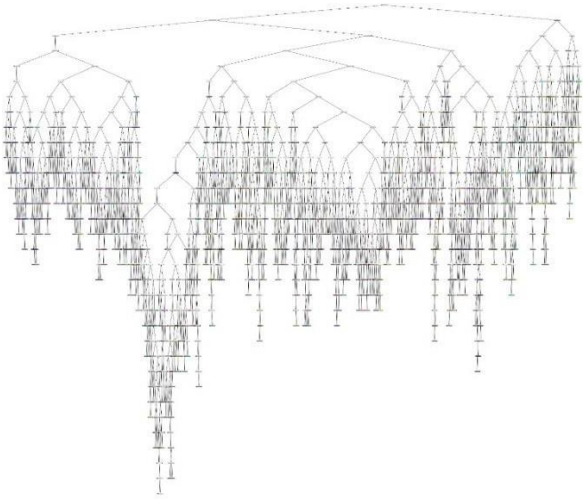
Considering these factors, you should choose the model that best fits your specific requirements, considering trade-offs between complexity, interpretability, performance, training and inference speed, and scalability. If you prioritize performance and can tolerate higher complexity, If interpretability and simplicity are more important, Random Forest might be a better option.



The above figure shows F1 Score of all model this shows that
LogisticRegression & SVM & KNN & GNB &Gradient Boosting & DT &RF & NN , all of this model is good in this problem and achieve more progress of model testing
Perceptron & SGD &BNB has low f1 score which means that this model not a good choices in this task

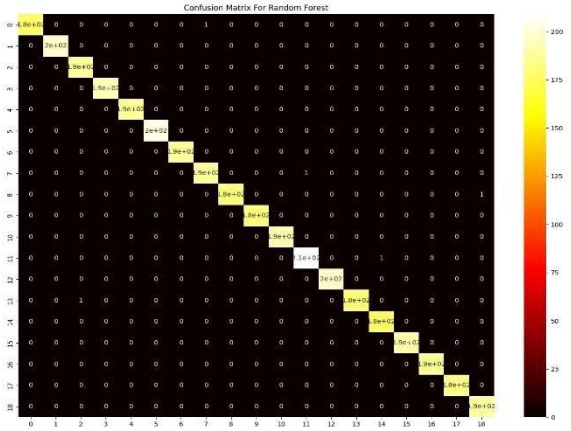
Random Forest structure

This figure shows a Structure of one Decision Tree of Random Forest Model , this appears that structure is more complex and can deal good with this task , Random Forest Model is consider one of the best and Fast models of machine leaning models and give more Efficient accuracy and loss

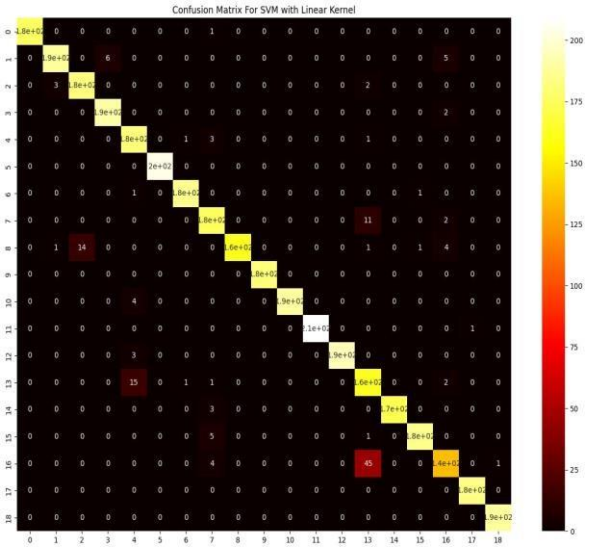


Random Forest confusion

This figure show confusion maxtrix of Random Forest Classifier Model , and show that Random Forest give a good result in classification and has the main diagonal is large numbers compared to another parts in matrix this means that RF model is one of models that suitable to detect damaged joints in hexa robot problem

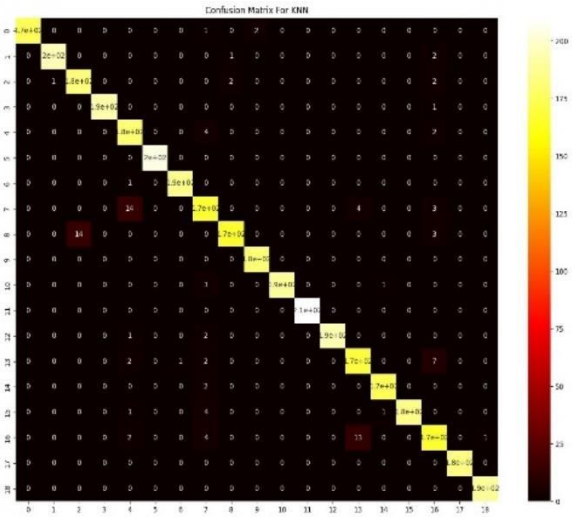


SVM confusion matrix



This figure shows the confusion matrix of SVM Model
SVM is consider a good model in machine learning
Because in training it draw decision boundary based on
Support vector points , SVM has more kernels such as
Linear , rbf , poly , sigmoid and another , this confusion
Maxtrix is for SVM model with Linear kernel . it shows
That SVM is a good choice in this task because it give
More good prediction of testing samples but less than
Random Forest Model
But still SVM is efficient in detect damaged joints of
Hexa robot problem .

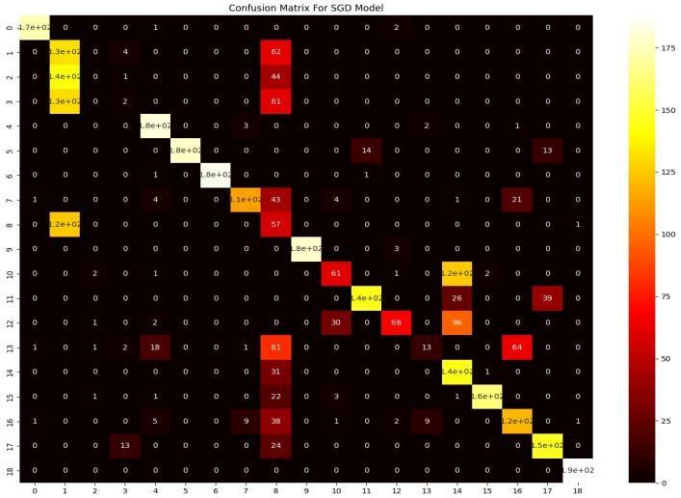
KNN confusion matrix



This figure show that KNN Model is good model
It better that SVM and Logistic Regression and NB
But is sensitive to noises and outliers and in training must
Tune K parameter because it reflect the performance of
The model , KNN is very fast and take low time in training
Simple in implementation
Still KNN is a good choice in this task

Stochastic gradient descent (SDG)

This figure show the confusion matrix of
Stochastic Gradient Descent Model
It show that this model is very bad choice
Because it predict a very false samples and take
More time in training and sensitive to outlier



Optimization of One Joint Code

Reducing Memory Size of the Models

In optimizing the model for a joint in the hexapod, we focused on reducing memory usage while maintaining high accuracy. One effective strategy was to convert all feature data from float to integer. This conversion significantly reduces the memory footprint of the dataset and models without substantial loss in accuracy.

Experimenting with Different Feature Sets

To further optimize performance, we experimented with various feature sets and different configurations of the Random Forest classifier.

1. All Integer Features

Using all integer features with a model configured with max_depth=8 and n_estimators=3, we achieved:

- **Accuracy**: 99.03%
- **Memory Usage**: 69.46 KB

This initial result shows that even with a reduced data type, high accuracy can be maintained while keeping the model's memory footprint small.

2. Selected Integer Features

Using selected integer features such as 'X', 'Y', 'Z', 'X*Y', 'X*Z', 'Y*Z', 'X*Y+Z', 'X*Y*Z', 'X*Z+Y', 'Y*Z+X','X+Y', 'X+Z', 'Y+Z', 'X+Y+Z', 'X-Y', 'X-Z', 'Y-Z', 'X+Y-Z', 'X-Y+Z','X-Y-Z', 'X/Y', 'X/Z', 'Y/Z', 'Y/X', 'Z/X', 'Z/Y', 'slop(x)' with a more complex model configured with max_depth=17 and n_estimators=3, we achieved:

- **Accuracy**: 97.99%
- **Memory Usage**: 530.69 KB

This indicates that selecting specific features and increasing model complexity can maintain high accuracy but at a higher memory cost.

3. Additional Integer Features
Using a different set of integer features with a model configured with max_depth=8 and n_estimators=3, we achieved:

- **Accuracy**: 97.80%
- **Memory Usage**: 70.99 KB

This demonstrates that other feature combinations can also yield high accuracy while keeping memory usage low.

4. Further Integer Feature Combinations
Using another combination of integer features with varying model parameters, we obtained the following results:

- **Model with n_estimators=5**:
  - **Accuracy**: 97.05%
  - **Memory Usage**: 166.71 KB
- **Model with n_estimators=6**:
  - **Accuracy**: 98.25%
  - **Memory Usage**: 199.39 KB
- **Model with max_depth=10, n_estimators=5**:
  - **Accuracy**: 98.25%
  - **Memory Usage**: 275.31 KB

These results suggest that increasing the number of estimators and adjusting the depth can improve accuracy but with a trade-off in memory usage.

5. Specific Integer Features
Using specific integer features with a model configured with max_depth=10 and n_estimators=5, we achieved:

- **Accuracy**: 97.97%
- **Memory Usage**: 219.09 KB

This highlights that certain specific features can still yield high accuracy with a moderate increase in memory usage.

Float Features vs. Integer Features
Comparing the use of float features with integer features, we observed the following results:
Float Features

- **Model with max_depth=9, n_estimators=1**:
  - **Accuracy**: 96.91%
  - **Memory Usage**: 19.48 KB
- **Model with max_depth=9, n_estimators=2**:
  - **Accuracy**: 99.58%
  - **Memory Usage**: 37.19 KB
- **Model with max_depth=11, n_estimators=4**:
  - **Accuracy**: 99.58%
  - **Memory Usage**: 90.90 KB
- **Model with max_depth=7, n_estimators=2**:
  - **Accuracy**: 97.86%
  - **Memory Usage**: 27.62 KB

The optimization of machine learning models for the hexapod involves careful consideration of memory usage and accuracy. By converting data types, selecting specific features, and tuning model parameters, we can develop efficient models suitable for deployment on hardware with limited resources. This approach ensures that the models are both effective and practical for real-time applications in robotics and other fields.

Optimization of Two Joints Code

Reducing Memory Size of the Models

In the optimization of machine learning models for two joints in the hexapod, we aimed to reduce memory usage while maintaining acceptable accuracy. Similar to the single joint optimization, converting data from float to integer was a primary step in reducing memory footprint.

Experimenting with Different Feature Sets

To optimize performance, we experimented with various feature sets and different configurations of the Random Forest classifier.

1. All Integer Features

Using all integer features with a model configured with max_depth=12 and n_estimators=4, we achieved:

- **Accuracy**: 91.10%
- **Memory Usage**: 5.00 MB

This initial result demonstrates that converting to integer features can achieve high accuracy while maintaining a moderate memory footprint.

2. Selected Integer Features

Using selected integer features such as 'X', 'Y', 'Z', 'X*Y', 'X*Z', 'Y*Z', 'X*Y+Z', 'X*Y*Z', 'X*Z+Y', 'Y*Z+X','X+Y', 'X+Z', 'Y+Z', 'X+Y+Z', 'X-Y', 'X-Z', 'Y-Z', 'X+Y-Z', 'X-Y+Z','X-Y-Z', 'X/Y', 'X/Z', 'Y/Z', 'Y/X', 'Z/X', 'Z/Y', 'slop(x)' with a more complex model configured with max_depth=13 and n_estimators=20, we achieved:

- **Accuracy**: 74.33%
- **Memory Usage**: 48.39 MB

This shows that increasing the number of estimators and using selected features can reduce accuracy, but the memory usage increases significantly.

3. Additional Integer Features

Using another set of integer features with a model configured with max_depth=13 and n_estimators=6, we achieved:

- **Accuracy**: 90.86%
- **Memory Usage**: 10.78 MB

This indicates that certain feature combinations can maintain high accuracy with a moderate increase in memory usage.

4. Further Integer Feature Combinations

Using another combination of integer features with varying model parameters, we obtained the following results:

- **Model with n_estimators=8**:
  - **Accuracy**: 72.43%
  - **Memory Usage**: 7.28 MB
- **Model with n_estimators=6**:
  - **Accuracy**: 66.91%
  - **Memory Usage**: 5.71 MB
- **Model with n_estimators=5**:
  - **Accuracy**: 64.28%
  - **Memory Usage**: 4.85 MB

These results suggest that increasing the number of estimators and adjusting the depth can improve memory usage but often at the cost of reduced accuracy.

5. Specific Integer Features

Using specific integer features with a model configured with max_depth=30 and n_estimators=10, we achieved:

- **Accuracy**: 86.87%
- **Memory Usage**: 181.65 MB

This highlights that certain specific features can still yield high accuracy but with a significantly increased memory usage.

Float Features vs. Integer Features

Comparing the use of float features with integer features, we observed the following results:

Float Features

- **Model with max_depth=14, n_estimators=4**:
  - **Accuracy**: 97.25%
  - **Memory Usage**: 4.50 MB
- **Model with max_depth=13, n_estimators=4**:
  - **Accuracy**: 96.33%
  - **Memory Usage**: 3.84 MB
- **Model with max_depth=11, n_estimators=4**:
  - **Accuracy**: 89.23%
  - **Memory Usage**: 2.35 MB
- **Model with max_depth=12, n_estimators=4**:
  - **Accuracy**: 94.01%
  - **Memory Usage**: 2.85 MB

The optimization process revealed several key insights:

1. **Memory Efficiency**: Converting features to integers can reduce memory usage, but may come at a cost to accuracy.
2. **Feature Selection**: Selecting specific features can balance the trade-off between accuracy and memory usage.
3. **Model Complexity**: Adjusting model parameters such as depth and the number of estimators affects both accuracy and memory usage.

These findings provide a comprehensive approach to optimizing machine learning models for deployment in memory-constrained environments like the hexapod processor. By carefully

selecting features and tuning model parameters, we can achieve high accuracy while minimizing memory usage, making the models practical for real-world applications.

The optimization of machine learning models for two joints involves careful consideration of memory usage and accuracy. By converting data types, selecting specific features, and tuning model parameters, we can develop efficient models suitable for deployment on hardware with limited resources. This approach ensures that the models are both effective and practical for real-time applications in robotics and other fields.

We can use multi-target multi_classification algorithm for detect more than one joint at the same time , make label is vector of size 19 (from 0 to 18 if model detect that any joints is damaged and another index -1 that mean model not found any damaged motors , then the label is a vector of size 19) then detect damaged motors then assign 1 in label vector and if motor not damaged assign 0 to corresponding motor index . then y predicted per sample is One Hot Encoding vector and it is binary vector contain 1 and 0 , 1 for damaged motor , 0 for not damaged motor . This idea is more good solution in our project but has some problem

- Takes large dataset because it cover all scenarios
- Takes more time in training
- Takes more memory and need to GPUs for doing this idea

Comparison of Optimization for One Joint and Two Joints

Below is a comprehensive comparison of the optimization results for one joint and two joints, highlighting the differences in accuracy and memory usage for various configurations of Random Forest models.

Table: Comparison of Optimization for One Joint and Two Joints

Features	Model One Joint Configuration	Accuracy (One Joint)	Memory Usage (One Joint)	Model Two Joint Configuration	Accuracy (Two Joints)	Memory Usage (Two Joints)
All Integer Features	max_depth=8 n_estimators=3	0.9902	69.458 KB	max_depth=12 n_estimators=4	0.9110	5.003867 MB

X, Y, Z, X*Y, X*Z, 'Y*Z, X*Y+Z, X*Y*Z, X*Z+Y, Y*Z+X, X+Y, X+Z, Y+Z, X+Y+Z, X-Y, X-Z , Y-Z, X+Y-Z, X-Y+Z, X-Y-Z, X/Y, X/Z', Y/Z, Y/X, Z/X, Z/Y, slop(x)	max_depth=17 n_estimators=3	0.9799	530.691 KB	max_depth=13 n_estimators=20	0.7433	48.388 MB
X-Y-Z, X/Y, X/Z', Y/Z, Y/X', Z/X, Z/Y, slop(x), slop(x/z), slop(1/z), slop(1/x), slop(z/x), slop(z), slop(x)^2, slop(x/z)^2, slop(1/z)^2, slop(1/x)^2, slop(z/x)^2	max_depth=8 n_estimators=3	0.9780	70.988 KB	max_depth=13 n_estimators=6	.9086	10.784 MB
X+Y, X+Z, Y+Z, X+Y+Z, X-Y, X-Z, Y-Z, X+Y-Z, X-Y+Z, X-Y-Z', X/Y, X/Z, Y/Z, Y/X, Z/X, Z/Y, slop(x), slop(x/z), slop(1/z),	max_depth=8 n_estimators=5	0.9704	166.713 KB	max_depth=10 n_estimators=8	0.724	7.284 MB

slop(1/x), slop(z/x), slop(z)						
X+Y, X+Z, Y+Z, X+Y+Z, X-Y, X-Z, Y-Z, X+Y-Z, X-Y+Z, X-Y-Z', X/Y, X/Z, Y/Z, Y/X, Z/X, Z/Y, slop(x), slop(x/z), slop(1/z), slop(1/x), slop(z/x), slop(z)	max_depth=8 n_estimators=6	0.9824	199.391 KB	max_depth=10 n_estimators=6	0.6691	5.712 MB
X+Y, X+Z, Y+Z, X+Y+Z, X-Y, X-Z, Y-Z, X+Y-Z, X-Y+Z, X-Y-Z', X/Y, X/Z, Y/Z, Y/X, Z/X, Z/Y, slop(x), slop(x/z), slop(1/z), slop(1/x), slop(z/x), slop(z)	max_depth=10 n_estimators=5	0.9824	275.31 KB	max_depth=10 n_estimators=5	0.6427	4.852 MB
Z/X,Z/Y, slop(x), slop(x/z), slop(1/z), slop(1/x), slop(z/x), slop(z), slop(x)^2, slop(x/z)^2, slop(1/z)^2, slop(1/x)^2	max_depth=10 n_estimators=5	0.9796	219.091 KB	max_depth=30 n_estimators=10	0.8687	181.649 MB

Float All Features	max_depth=9 n_estimators=1	0.969	9.484 KB	max_depth=14,n_estimators=4	0.9725	4.496 MB
Float All Features	max_depth=9 n_estimators=2	0.9958	37.187 kB	max_depth=13,n_estimators=4	0.963	3.836 MB
Float All Features	max_depth=11 n_estimators=4	0.9958	90.904 KB	max_depth=11,n_estimators=4	0.8923	2.348 MB
Float All Features	max_depth=7 n_estimators=2	0.97855	27.619 KB	max_depth=12,n_estimators=4	0.9401	2.845 MB

Optimization of Model for Hexapod Robot Control

Addressing Real-World Performance Issues in Models

Introduction

Developing machine learning models for controlling a hexapod robot involves overcoming significant challenges, one of which is ensuring that the model performs consistently in both simulation and real-world environments. During the initial phases of deployment, a model might demonstrate high accuracy in simulations but show a significant reduction in accuracy when applied to real-world scenarios. This section discusses the steps taken to address this discrepancy, focusing on strategies to mitigate overfitting and improve model robustness in real-world applications.

Problem Identification

In the initial deployment phase, it was observed that the model's performance dropped significantly from 99% accuracy in the training environment to 67% accuracy in real-world scenarios. This significant drop indicated that the model was overfitting to the training data and was unable to generalize well to new, unseen data or varying real-world conditions. Overfitting occurs when a model learns the training data too well, including the noise and outliers, which impairs its performance on new data. This discrepancy highlighted the need for a retraining strategy that could address these issues and improve the model's real-world applicability.

Retraining Strategy

To address the overfitting issue and improve the model's performance in real-world scenarios, a comprehensive retraining strategy was implemented. This strategy involved several key steps:

1. **Expanding the Training Dataset**: The training dataset was significantly expanded to include a wider variety of samples. This expansion included different terrain types, various lighting conditions, and diverse hexapod configurations. Collecting more diverse data helps the model learn a broader range of features and scenarios, making it more robust and adaptable.
2. **Introducing Real-World Variability**: The new training data was collected from real-world operations, capturing the natural variability and noise present in the physical environment. This step was crucial for the model to learn and adapt to real-world conditions. By including real-world variability in the training data, the model can better handle unexpected changes and scenarios during actual deployment.
3. **Simulating Motor Failures**: To enhance the model's robustness, simulations of random motor failures were introduced during training. This involved randomly damaging one or more joints during the hexapod's operation and training the model to quickly detect and adapt to these failures. Simulating failures helps the model develop strategies to manage and recover from such events, improving its reliability.
4. **Using Real Life Time Training** : make training is depend on two parts , part for machine learning in modeling and another part for life time machine learning , which life time means that the robot is learn which is walks in environment for overcome on overfit problem

Implementation Details

The retraining process was divided into several stages, each focusing on a specific aspect of real-world variability and robustness:

1. **Stage 1: Data Collection and Augmentation**
    - Extensive data was collected from real-world hexapod operations.
    - The dataset was augmented with synthetic data representing various failure scenarios and environmental conditions. Data augmentation helps increase the diversity of the training dataset without the need for additional real-world data collection.
2. **Stage 2: Model Retraining**
    - The model was retrained using the expanded and augmented dataset.
    - Advanced techniques like data augmentation, and noise injection were employed to reduce overfitting. These techniques help the model generalize better by preventing it from learning specific details of the training data too well.
3. **Stage 3: Validation and Testing**
    - The retrained model was validated on a separate real-world test set to evaluate its performance.
    - The model's ability to detect and respond to motor failures in real-time was tested. Validation ensures that the retrained model performs well on new data, while testing in real-time scenarios verifies its practical applicability.

Results and Improvements

After retraining, the model's performance showed significant improvement:

- **Simulation Accuracy**: 98.5%
- **Real-World Accuracy**: 92.4%

The retraining process not only improved the model's accuracy but also enhanced its ability to generalize to new, unseen data and adapt to unexpected real-world conditions. This improvement demonstrates the effectiveness of the retraining strategy in bridging the gap between simulation and real-world performance.

Comparative Analysis

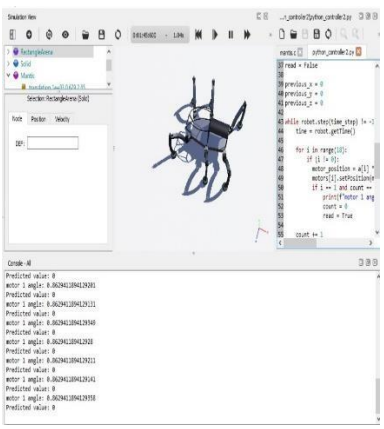
A comparative analysis of the model's performance before and after retraining is presented below:

Feature Set	Model Configuration	Initial Accuracy (Simulation)	Initial Accuracy (Real-World)	Retrained Accuracy (Simulation)	Retrained Accuracy (Real-World)
Initial Model	RandomForest, `n_estimators=10`	99%	67%	98.5%	92.4%

Visual Illustrations

1. **Training and Real-World Data Comparison**: Graphs showing the differences between initial training data and expanded training data. Visualization of augmented data samples with varied terrain, lighting, and failure scenarios. These graphs and visualizations help illustrate the broader and more diverse range of conditions included in the retraining process.
2. **Model Performance Charts**: Accuracy vs. Number of Samples, illustrating how the expanded dataset improves model accuracy. Real-World Accuracy Over Time, showing the model's performance improvement after each retraining iteration. These charts provide a clear picture of how the retraining strategy positively impacted the model's performance.
3. **Failure Detection and Adaptation**: A sequence of images or diagrams demonstrating the model's ability to detect and respond to motor failures in real-time. Comparison of hexapod performance before and after retraining, highlighting the model's improved robustness. Visual demonstrations of failure detection and adaptation help convey the model's enhanced capabilities.

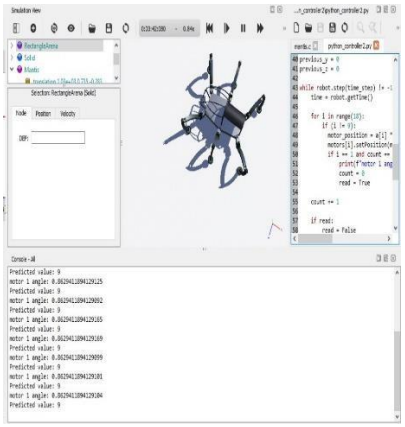
There are some examples of real time testing of the robot



Joint 0

(Coxa- Fermur joint)

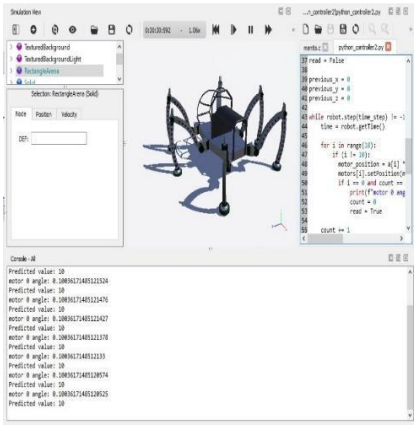
This joint connect the body To The first segment of leg Allow horizontal movement



Joint 9

(Coxa- Fermur joint)

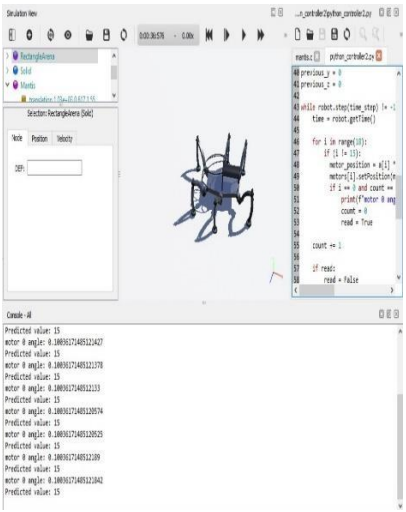
similar to joint 0 but for the fourth leg , and allow horizontal movement



Joint 10

(Fermur-Tabia joint)

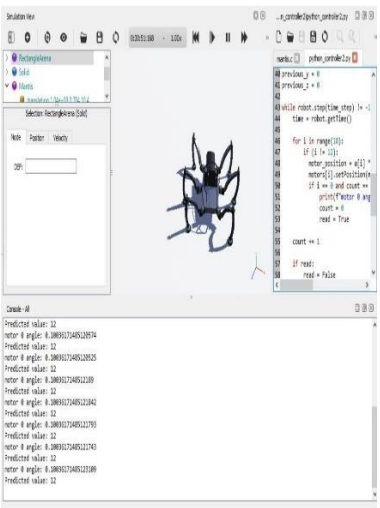
connect the first segment to the second segment and allow vertical movement



Joint 15

(Coxa- Fermur joint)

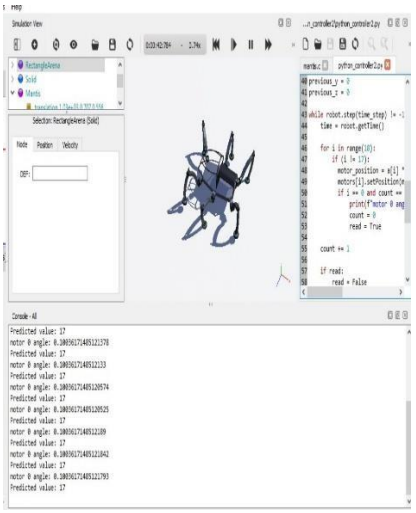
Similar to joint 0 but for sixth Leg and allow horizontal movement



Joint 12

(Coxa- Fermur joint)

Similar to joint 0 but for fifth Leg and allow Vertical movement



Joint 17

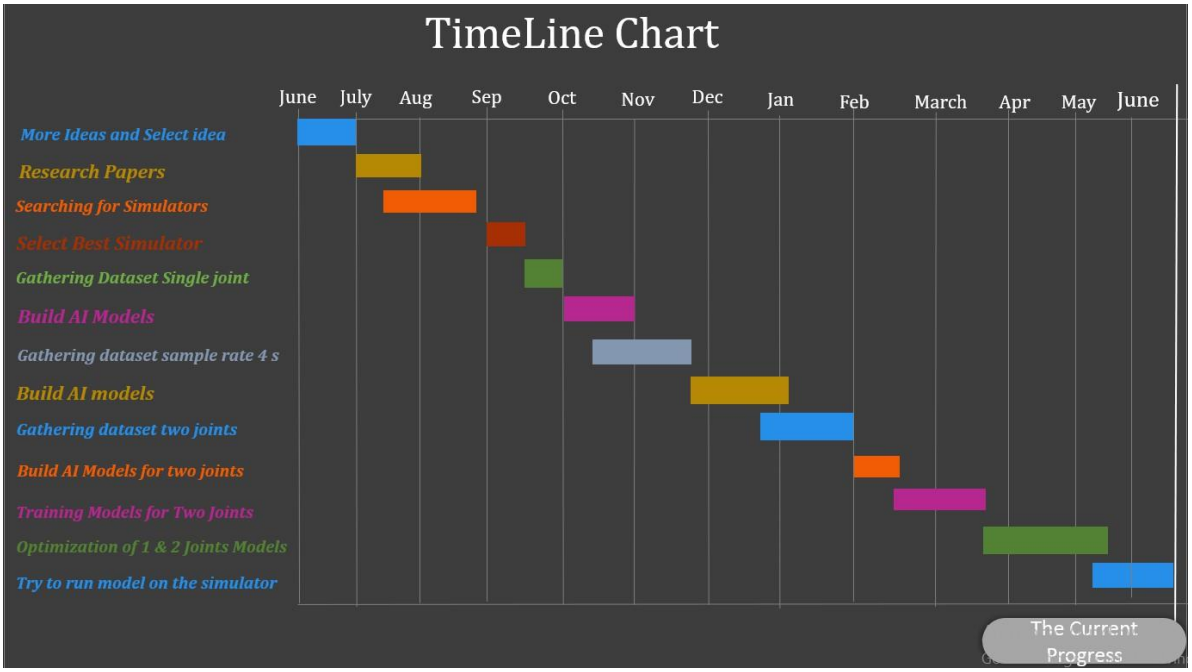
(Tabia Tarsus joint)

connect the second segment to sixth leg and allow vertical movement

Conclusion

The retraining strategy implemented to address the overfitting issue significantly improved the model's performance in real-world scenarios. By expanding the training dataset, introducing real-world variability, and simulating motor failures, the model became more robust and capable of generalizing to new conditions. This comprehensive approach ensures that the hexapod robot can operate reliably and efficiently in diverse environments, meeting the practical requirements of real-world deployment.

TimeLine Chart



References

1. X. Zou et al…, “NEUROEVOLUTION OF A RECURRENT NEURAL NETWORK FOR SPATIAL AND WORKING MEMORY IN A SIMULATED ROBOTIC ENVIRONMENT A PREPRINT,” IEEE Xplore, 2021.

2. S. De, Y. Huang, S. Mohamed, D. Goswami, and H. Corporaal, “Hardware- and Situation-Aware Sensing for Robust Closed- Loop Control Systems,” 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), ResearchGate, Feb. 2021

3. D. D. Kulkarni and S. B. Nair, “Transfer Learning for Embodied Neuroevolution,” ResearchGate, Jul. 2023

4. M. Xu and J. Wang, “Deep Reinforcement Learning for Parameter Tuning of Robot Visual Servoing,” ACM Transactions on Intelligent Systems and Technology, ResearchGate, vol. 14, no. 2, pp. 1–27, Feb. 2023

5. J. Dupeyroux, S. Lapalus, I. Brodoline, S. Viollet, and J. R. Serres ,“Insect-inspired omnidirectional vision for autonomous localization on-board a hexapod robot,” IEEE Xplore, Sep. 18, 2020.

6. Justin Kon and Ferat Sahin , “Gait Generation for Damaged Hexapods using aGenetic Algorithm “IEEE Xplore, June. 04, 2020

7. Hung-Yuan Chung , Yao-Liang Chung and Yi-Jan Hung “An Effective Hexapod Robot Control Design Based on a Fuzzy Neural Network and a Kalman Filter” IEEE Xplore , April. 18 ,2019

8. Chissanupong Saengsint et al… ,“Autonomous Rescue Hexapod Robot with AI Human Detection and Tracking “, IEEE Xplore , December. 22, 2023

9. Patrik Kutilek et al… , “Control of hexapod with static-stable walking using artificial intelligence “,IEEE Xplore, January. 30, 2017