Misan Journal of Engineering Sciences ISSN-E: 2957-4250
Vol. 3, No. 1, July 2024 ISSN: 2957-4242

Forecasting Robot Movement with Sensor Readings and Multi-Layer Perceptron Models

Sarah Sabeeh

Computers Engineering Department, University of Basrah ,Basrah ,Iraq
Author E-mail: sarahsabeeh2020@gmail.com

(Received 14 April, Revised 8 June, Accepted 8 June)

Abstract: Classification of sensor data is applied in several domains and enables application tasks such as fault detection, event recognition, and predictive maintenance. This paper proposes an application of Multi-Layer Perceptron networks for the classification of noisy and imbalanced sensor data. In contrast to conventional methods, the proposed approach addresses imbalanced classes, noisy signals, and high-dimensional data using state-of-the-art preprocessing techniques and thorough hyperparameter tuning. Using a dataset collected from a SCITOS G5 mobile robot fitted with 24 ultrasound sensors, this research discusses in depth the intricacies of Multi-Layer Perceptron (MLP) neural networks for the optimal performance of their architecture and training process. Experimentation and analysis show that the proposed system yielded a good accuracy of 93.04% on the test dataset, surpassing traditional techniques such as Support Vector Machines and Logistic Regression. In addition, thorough evaluation metrics on accuracy, precision, recall, and F1 score demonstrate the model's MLP efficacy across different classes of movements. In this respect, the study shows not only the effectiveness of MLP networks in sensor data classification but also best practices in handling challenges that come with the handling of sensor data.

Keywords: Logistic Regression, Movement Prediction, Multi-Layer Perceptron, Robot Sensor Data, Random Forest, Support Vector Machines, XGBoost

1. Introduction

There is an increase in the generation of data in numerous sectors including healthcare and industrial monitoring due to the rise in sensor technologies. The huge volume of data from sensors has possibility of unraveling knowledge that can improve decision-making, optimize resource management as well as advance operational efficiency [1]. Nonetheless, the realization of sensor data's potential depends on efficient methods used for making sense and interpretation of it.

Data classification is essential in sensor data analysis. Sensor-data classification involves clustering data points into predefined groups based on certain observed features [2]. This, in turn, helps in discerning patterns; exception and trends within the data that facilitate tasks like fault detection, event recognition and predictive maintenance [3]. Precise categorization of sensor data is important for various applications; this includes diagnosing diseases in healthcare; monitoring pollutants as well as identifying activities in smart environments [4].

DOI: https://doi.org/10.61263/mjes.v3i1.77

Sensor data classification is highly significant; nevertheless, it poses huge challenges. It includes such issues as its high-dimensionality nature, noise levels and missing values [5]. These complexities are difficult to be handled by traditional classification techniques, which leads suboptimal performance more than anything else. Therefore, lately there has been an interest in using advanced machine learning approaches in order to enhance the category of sensor-data like neural networks for example.

To ensure that sensor data is classified accurately, there has been an attempt to try the use of Multi-Layer Perceptron networks [6]. The study by authors [7] involves a deep investigation into different areas of MLP neural networks such as handling class imbalances, optimizing learning rates, applying regularization techniques, preprocessing data, implementing cross validation and fine tuning hyperparameters.

Our study is based on a dataset obtained from a SCITOS G5 (which is a mobile robot platform for research and industrial applications. See Fig. (1) that is navigating through a room following the wall in a clockwise direction for four rounds. The navigation process relies upon 24 ultrasound sensors located around its waist. Some of these files contain raw sensor readings while others consist of reduced distances and corresponding class labels showing what the robots did such as move forward or turn left etc. [8]. The minimum distance to any object within defined sections ahead of the robot represents simplified distances. This dataset serves as our initial experiment in machine-learning approaches for classification of sensor data with regard to multi-layer perceptron networks.

Fig. 1: SCITOS G5 Mobile Robot alongside Its Accompanying Devices.

This study had two objectives, which were achieved through experimentation and analysis. (1) Testing the efficiency of MLP models in grouping sensor data accurately by examining how well multi-layer perceptron neural networks perform in classifying sensor data, given various obstacles like imbalanced classes, noisy signals and accurate classification values. (2) The second objective was to identify ways in which reliable and accurate methods for classifying sensor data can be improved upon through experiments that target effective techniques and setups for improving classification accuracy rates without any error. This will guide on the best way of dealing with the sensor information.

In this paper, the goal is to employ MLP networks for classifying imbalanced and noisy sensor data differently from other studies. A new application of MLP networks is proposed for classifying imbalanced and noisy sensor data. To improve the quality of input data, our approach employs advanced preprocessing techniques that include oversampling for data imbalance as well as noise reduction methods. Furthermore, hard hyperparameter optimization is employed to ensure optimal performance of MLP model. The contributions of this paper lie in providing:

1. **Enhanced Model:** Conducted an in-depth analysis of Multi-Layer Perceptron (MLP) networks for predicting robot movements based on sensor data. This involved a detailed examination of the architecture and training processes of MLP networks, addressing challenges such as class imbalance, noisy signals, and high-dimensional data.
2. **Enhanced Preprocessing:** The aforementioned model presented in this paper is soaked with preprocessing and comprehensive optimization hyperparameter for MLP. Thus, applying, for example, oversampling the data for the data imbalance and noise on the one hand, the paper benefits from the resultant enhanced accuracy and robustness on the other. It also finds the best configuration for the methods that would yield the highest of precision score in the detector.
3. **Comparative Analysis:** Provided a comparative study of various machine learning models on this dataset, offering valuable insights into their performance and the effectiveness of the proposed algorithms. The study included thorough evaluation metrics such as precision, recall, and F1 scores across different classes of movements, demonstrating the efficacy of the MLP model.

All these contributions sum up to enrich the latest in class termed sensor data, thereby coming up with a better solution of real-international programs. It is evident that through the results presented in this study, the proposed method success with improved performance capable of being used in different practical situations, self-sufficient navigation structures and techniques, and industrial automation strategies among others.

The paper is established as follows; Section 2 introduces the exclusive techniques used inside the robotic sensor information category. Section 3 details the technique used in this research, overlaying information preprocessing steps and experimental setup. Section 4 shows the outcomes and delves into findings, while Section 5 gives insights and conclusions drawn from this take a look at, in conjunction with instructions for future studies.

2. Literature Review

This section delves into the latest developments in robot control and motion prediction. The discussed methods cover various approaches, including the use of anticipatory control to enable human-robot collaboration, the application of deep visual foresight for motion planning, the use of deep learning to predict object movements, the use of recurrent neural networks to predict human motion trajectories, the implementation of ensemble learning to predict swarm robot motion speeds, the utilization of LSTM networks to forecast mobile robot sensor data related to robot motion, the deployment of proactive anomaly detection techniques for robot navigation enhancement, and the integration of sensor fusion for estimating human walking paths.

In the 2016 study by Huang & Mutlu, an anticipatory method of robot control was presented to enhance human-robot collaboration. It predicts human intentions based on their gaze patterns and lets the robot complete tasks proactively. This approach reduced task completion time when compared with reactive control methods, showing the advantages of anticipatory behavior for effective human-robot teamwork [9]. Nonetheless, the paper also describes several issues and challenges that it faces. First, there is the reality that gaze target inference is not consistently precise, which means that the prediction of intentions and subsequent actions are not always accurate. The report states that the study observed 18.75% of the cases of incorrect target identification in this eye gaze inference technique. Additionally, the model suggests that there is no link between episodic interactions, which may be why the decision-making process is overlooked. This overly simplified description could result in incorrect predictions and ineffective actions. Fig. (2) demonstrates how these components of the anticipatory robot system are integrated into the implemented system.

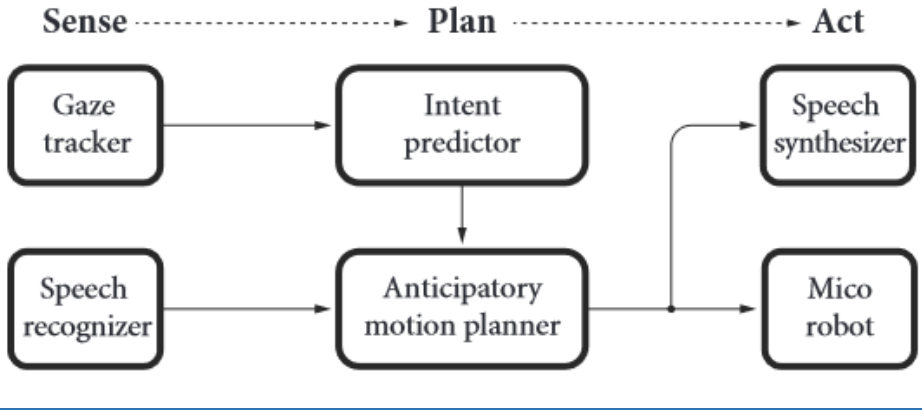

Fig. 2: Components of the Anticipatory Robot System.

Also, Finn and Levine (2017) suggested a technique for robot movement planning utilizing deep visual anticipation. Their strategy integrates action-driven video forecasting models with model-based predictive control methods, and leverages training data. This cutting-edge method allows robots to carry out non-prehensile handling duties, such as moving items, without the need for oversight or exact monitoring and operation [10]. The paper's main issue is its suitability only for tasks that can be completed in a short time. Deep video prediction models are now able to predict future frames a few steps ahead, a discovery made by researchers. This method's limitation restricts its application in situations requiring long-term planning or handling complex environmental dynamics.

Seker, Tekden, and Ugur (2019) did a study that showed a deep learning model that lets a robot guess what will happen when it moves things around. They used a computer program that can act like the real world to teach the robot how to guess how things will move when they're lifted up. This model uses something called Long Short Term Memory networks to learn from pictures of things and make a path for how they'll move. Using something that has to do with pictures and Long Short Term Memory networks is better than other ways of predicting, and it works well with all kinds of different shapes and real life stuff [11]. However, the study also acknowledges its constraints. Principally, the highest prediction performance is achieved with manually engineered features, rendering it impossible to designate distinct features for each action and task. Subsequently, the autoencoder models exhibit more variable errors due to their unsupervised nature, which can compromise prediction reliability. Moreover, the model might struggle to cope with tool-object pairs that have varying friction coefficients, especially when attempting to ascertain diverse coefficients for enhanced generalization.

In another study conducted by Zhang et al. (2020), the researchers propose utilizing a deep learning approach that leverages Recurrent Neural Networks (RNNs) to predict human motion paths in assembly operations. This facilitates preemptive robotic assistance while adhering to safety protocols [12]. However, despite introducing their technique, the article does not address potential downsides or the limitations of the suggested approach. This model may not account for all human movement patterns during assembly tasks, so it might not work in real-world scenarios. The model's computational complexity and training process also aren't mentioned, which is a significant issue because it could impact real-time applications in industrial environments. In short, the method's potential application to other assembly tasks or environments isn't fully explored, so more research and improvements are necessary. Fig. (3) shows an example structure of the advanced RNN.

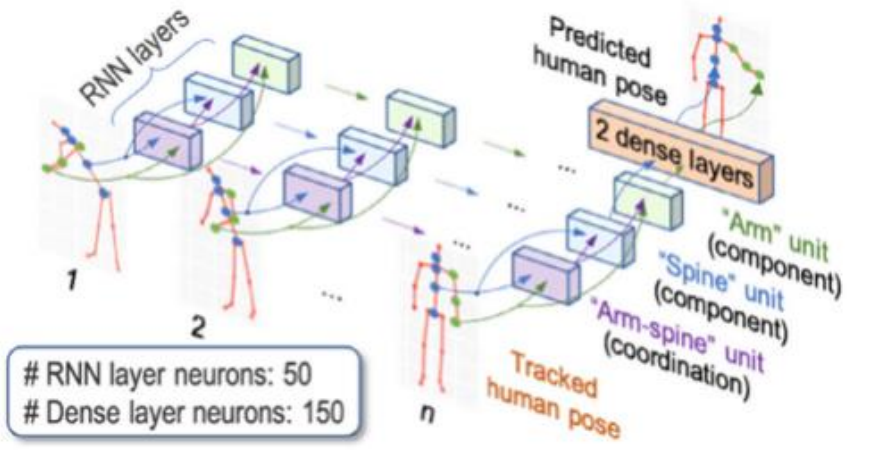

Fig. 3: Enhanced RNN for Improved Accuracy in Motion Prediction.

Khaldi et al. (2021) explore the application of Ensemble Learning (EL) methods for forecasting swarm robot movement speed. They assess the performance of Boosted Trees (BST) and Bagged Trees (BT) algorithms against Support Vector Regressors (SVRs) and Gaussian Process Regressors (GPRs). Utilizing simulated sensor data analysis, they demonstrate the superior effectiveness of BST and BT compared to traditional machine learning models across diverse swarm scenarios [13]. However, the paper has its shortcomings, such as focusing predominantly on current swarm movement speed prediction without considering its progression, the absence of discussions on sensor noise except for future improvement suggestions, and the limited examination of model performance in different obstacle-laden In addition, discussing the resolution of certain issues without mentioning transfer learning and deep learning-based fault detection is not feasible, even though they are not explored in this study, this reality leaves ample opportunity for future research.

In a separate study, Vagale et al. (2021) examined the use of ensemble-learning methods to predict swarm motion speeds. They used LSTM network structures to predict time series data from motion sensors on robots. The results showed that the LSTM stacked model performed slightly better in short-term forecasts. The research analyzed motion sensor data from mobile robot driving sessions. The study concluded by comparing the predicted trajectories to the actual paths of the robots [14]. Nevertheless, the paper highlights the weaknesses and areas that require further research. The research indicates that the selection of prediction sequence length and the limitation of dataset diversity, which includes similar trajectories in a confined environment, can impact the reliability of trained models. The paper recommends compiling a larger and more varied dataset to enhance model performance. It also explores the potential of employing alternative neural network structures, such as Bayesian neural networks, to improve forecasting outcomes. Additionally, it suggests preparing the robot to calculate the next coordinates based on future simulations rather than sensor readings for future advancements in this field.

Ji et al. (2022) presented an anomaly detection network named PAAD. This network is designed to optimize robot navigation in uncertain environments by predicting potential future failures. It performs this prediction by examining planned movements and current sensor data, effectively synthesizing signals from sensors to accurately detect anomalies. The experimental outcomes show its capability to detect failures in real time with a low rate of incorrect detections [15]. PAAD is a system that helps robots find their way around, and it can spot anything strange happening. However, there are some issues with it. If the sensors on the robot get blocked, it might see things that aren't really there. Also, if there are too many things moving around quickly, the robot might struggle to keep up. Plus, the robot might not be able to use this system very

easily because it needs a lot of computing power. To make things better, there is a need to find ways to deal with the sensors getting blocked, to make it more adaptable to places that are always changing, and to make it require less computing power. There's a picture of how the network is set up in Fig. (4).

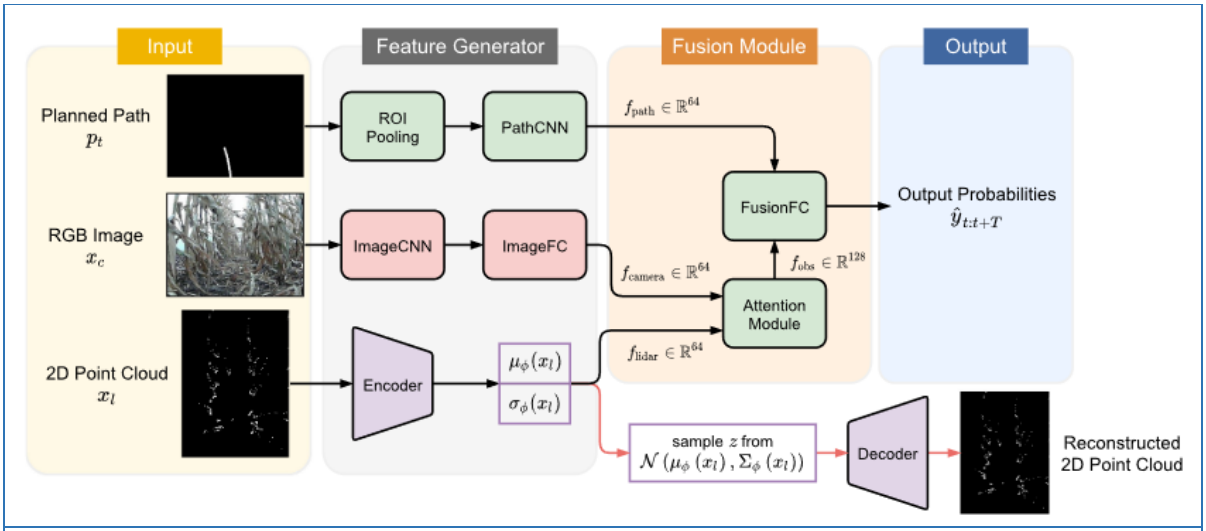

Fig. 4: Model Architecture Overview of the Proposed System.

Additionally, Rabb and Steckenrider (2023) introduced a new method for predicting an individual's walking path using sensor fusion and a step model. This approach integrates global and inertial measurements to continuously update estimations with real-time data inputs. In comparison to a particle filter, this method shows similar precision and enhanced effectiveness, providing potential applications in various fields such as biomechanics and human-robot interactions [16]. Other comments that have been made include the need to adjust the step model and its updating method to potentially improve accuracy even further. Subsequent studies are recommended to validate the framework using actual experimental data, both offline and in real-time. These applications can be utilized in the following areas: ecological biomechanics measurements, hazardous military and civilian fieldwork scenarios, human-robot collaboration, and the localization of bipedal humanoid robots.

Table (1) gives a summary of research advancements in the field of robotics displaying different methods and their impacts on improving robot capabilities and interactions with humans.

Table 1: Recent Advances in Robotics Research: Approaches and Contributions

Authors	Year	Approach	Contribution	Drawbacks
Huang & Mutlu (2016)	2016	Anticipatory robot control for collaboration	Proactive task completion through gaze-pattern prediction	The method's main drawback is uncertainty in inferring gaze targets, causing errors in intent prediction and action.
Finn and Levine (2017)	2017	Deep visual foresight for motion planning	Unsupervised learning for non-prehensile manipulation	Computational limitations hinder planning horizons, suggesting the need for faster platforms
Seker, Tekden, & Ugur (2019)	2019	Deep effect trajectory prediction	LSTM-based model for predicting object movement trajectories	limitations in specifying features for different actions, variability in autoencoder model errors affecting prediction reliability, and challenges in handling tool-object pairs with varying friction coefficients for robust generalization
Zhang et al. (2020)	2020	Recurrent neural network for trajectory	Proactive assistance in human-robot collaborative assembly	The model may not cover all human movement patterns in assembly tasks
Khaldi et al. (2021)	2021	Ensemble learning for swarm motion prediction	Superior performance in predicting swarm robot motion speeds	Lacks investigating swarm behavior evolution, sensor noise, and obstacle environments, and skips exploring transfer learning and deep learning-based fault detection.
Vagale et al. (2021)	2021	Time series forecasting of mobile robot motion	Comparison of LSTM network architectures for motion prediction	Dataset diversity and prediction sequence length, suggesting larger datasets and alternative architectures like Bayesian networks for better performance
Ji et al. (2022)	2022	Proactive anomaly detection for navigation	Effective multi-sensor fusion for real-time failure detection	Susceptibility to sensor occlusion causing false detections, potential limitations in dynamic environments, and computational complexity
Rabb and Steckenrider (2023)	2023	Walking trajectory estimation using sensor fusion	Accurate estimation of human walking trajectories with sensor fusion	Drawbacks mentioned include the need for tuning the step model and its updating mechanism to potentially increase accuracy further

The weaknesses noted in the papers reviewed in the literature define why MLP networks are going to be suitable for this research, and the rationale for that is highlighted below. For example, Huang & Mutlu (2016) have pointed to the drawbacks of the coming solutions as including a faulty eye direction target which is likely to elicit wrong predictions and ineffective actions. MLP are well suited to the current situation that need the more flexible and stable approach to the classification tasks, by which the erroneously classified objects could be avoided. For example, the works such as Finn & Levine (2017) or Seker, Tekden & Ugur (2019) have the same thing to discuss in their papers – the inefficiency of available tools because they might not work well for strategic planning or they might not be effective when facing environmental fluctuations, for example. Thanks to the ability to incorporate different features within MLP networks, it can work with any complexity level, and address each pattern instantly, eliminating these drawbacks. Furthermore, Khaldi et al. (2021) and Vagale et al. (2021) pointed out that accuracy is mandatory for IoT applications otherwise the possibility of MLP networks enhancing the performance as a result of their capacity to capture

complexities inherent in tested sensors may be realized. Furthermore, in the independent study by Rabb and Steckenrider (2023), there are indications of the need for simulation in mapping out walking paths and areas as the ML prototocracies have the fired as they keep on improving and expanding for such purposes.

This proposed study will build on the advances of MLP networks, as mentioned in the previous research and will consider their shortcomings as well. The purpose was to improve the preeminence and classification of robot movement, and thus facilitate the advancement of these fields.

To sum up, this work aims at improving the categorization of robot sensor data to use Multi-Layer Perceptron neural networks and evaluate the methods to enhance classification accuracy. This way, its practical tests and evaluations' aim is to identify better ways of dealing with challenges in the analysis of sensor data to help drive the progress needed for real life applications that require reliable classification of sensor data.

3. Methodology

This research involves employing a specific multi-layer perceptron neural network to classify data derived from sensor measurements. A systematic process is applied step-by-step (See Fig. 5). Initially, the sensor data is collected from the SCITOS G5 mobile robot and organized in quad/unimodal patterns, utilizing 24 ultrasound sensors along with corresponding move markers. The subsequent step involves collecting the data and then conducting a preprocessing and cleaning phase to ensure its quality and consistency. Following this, the MLP network architecture is carefully developed, taking into consideration the attributes of the sensor data and the performance of the classifier. By employing backpropagation during the preprocessing of the training data, the network is able to normalize the weights and biases. This multi-level perceptron network, once analyzed and evaluated, provides insights into its effectiveness in classifying sensor data using metrics such as accuracy and precision. This methodical approach seeks to understand the characteristics and potential of MLP models for classification tasks involving real-world sensor data.

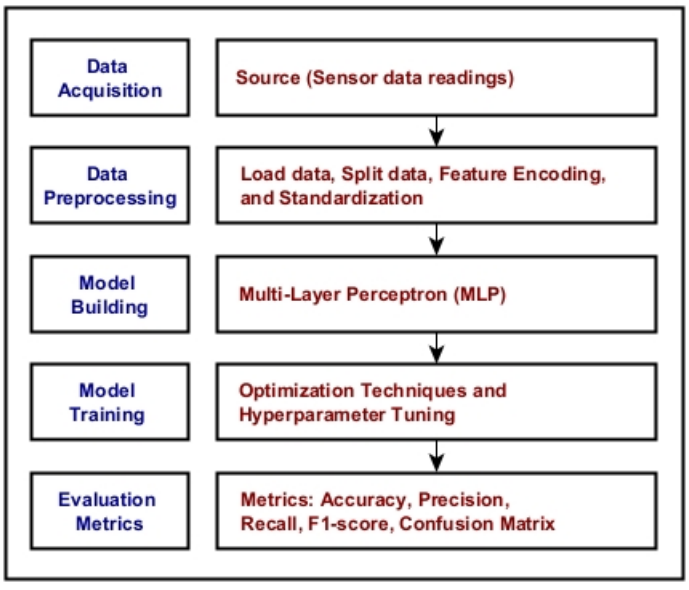

Fig. 5: Methodology for Applying MLP in Sensor Data Classification.

3.1 Dataset Description and Preprocessing

The section describes the dataset used for developing and testing a MLP neural network, as well as the steps taken to prepare the data for training the model.

3.1.1 Dataset Description

The dataset is taken from a SCITOS G5 mobile robot that followed a set path within a room. The robot is fitted with 24 ultrasound sensors located around its perimeter, recording distance measurements in multiple angles. The robot completed four full circuits around the room, moving in a clockwise direction, to ensure a comprehensive and varied data set.

The dataset comprises 24 simultaneous ultrasound sensor measurements, with each sensor recording the distance to the nearest object within its range. To simplify analysis, these original sensor readings are categorized into simplified distance measurements by taking the minimum readings within specific angle arcs around the robot, thereby condensing proximity data into essential directional zones. A movement label indicating the robot's movement state, which includes 'move forward,' 'turn left,' 'turn right,' and 'stop', accompanies each set of sensor readings. Data was collected at a rate of 9 samples per second, resulting in a comprehensive dataset of 5,456 instances over four navigation rounds.

3.1.2 Preprocessing Steps

To prepare the dataset for training the MLP neural network, several preprocessing steps were undertaken. First, the CSV dataset was loaded using NumPy's **np.loadtxt()** function, splitting it into features (sensor readings) and target variables (movement labels). The movement labels, initially in categorical format, were converted to numerical values using **LabelEncoder** from **sklearn.preprocessing**. Next, the sensor readings were standardized with **StandardScaler** to ensure a mean of 0 and a standard deviation of 1, thus preventing any single sensor from disproportionately affecting the model. Missing values in the dataset were handled by imputing the mean sensor readings, maintaining data consistency. Finally, the dataset was split into training (80%) and testing (20%) sets using **train_test_split** from **sklearn.model_selection**, ensuring that the label distribution was representative in both sets.

The data was gathered from the SCITOS G5 cell phone as it made its way around the room over a period of four laps. The arrangement of the items that were in the examination room can be observed in Fig. (6), and the route that the machine adhered to is depicted in Fig. (7) [17].

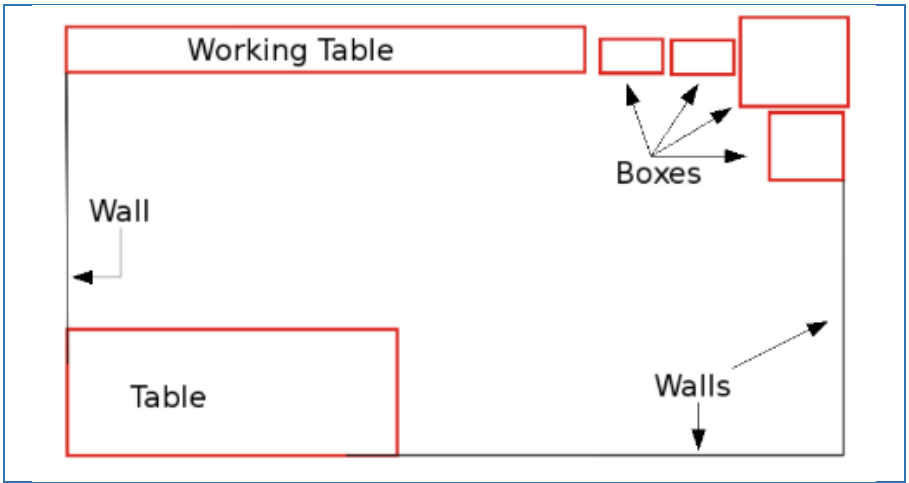

Fig. 6: Sketch of the Robot's Environment Where the Robot Navigates.

The process of data collection was carried out at a rate of nine samples per second. This created a database that contains 5456 examples. This data is utilized to train the multi-layer perceptron classifier that is discussed in this study.

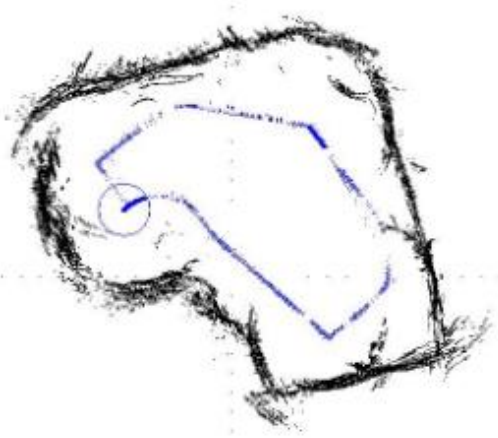

Fig. 7: Perception of the Environment and Path followed by SCITOS G5 Robot [17].

3.2 Overview of Multi-Layer Perceptron (MLP) Neural Networks

In this part, the structure, setup, and learning process of the MLP neural network used for identifying the robot's movement states from ultrasound sensor readings are explained.

3.2.1 Network Architecture

The MLP neural network utilized in this research is a fully connected feedforward neural network, ideal for supervised learning tasks like classification. The network includes the following layers [18] (Refer to Fig. 8).

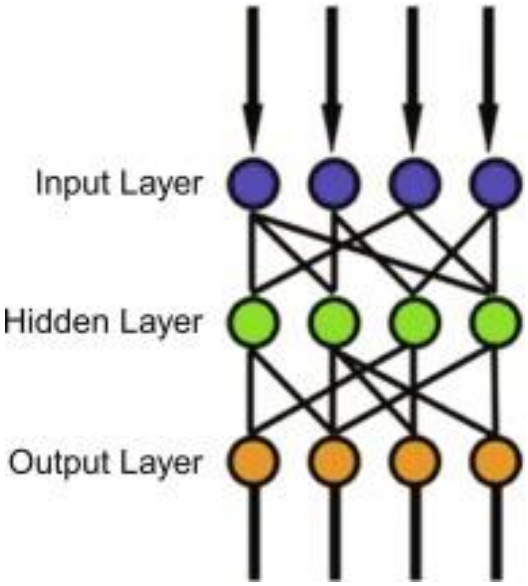

Fig. 8: Overview of Multi-Layer Perceptron Neural Network Architecture.

1. **Input Layer:**

The input layer gets the standardized sensor readings. Since there are 24 ultrasound sensors, the input layer has 24 neurons, each one corresponding to one of the sensor readings.

2. **Hidden Layers:**

The neural network has two hidden layers that process complex information from the sensors. Each of these layers consists of 64 units, providing a balance between learning capacity and computational efficiency. Both layers use the ReLU activation function, enabling the network to capture non-linear relationships. The first layer extracts initial features, while the second layer refines these features to enhance the network's understanding.

3. **Output Layer:**

In the output layer, four neurons represent potential movement states for the robot: 'move forward,' 'turn left,' 'turn right,' and 'stop.' These neurons utilize the softmax activation function, converting raw output scores into probabilities. By ensuring that the probabilities of all classes sum up to 1, this function generates a probability distribution for the possible movement states, facilitating accurate prediction.

3.2.2 Configuration and Hyperparameters

The architecture and hyperparameters of the MLP neural network substantially impact its efficacy. Key hyperparameters were defined for training as follows:

1. **Loss Function:** A categorical cross-entropy loss function was implemented, suitable for multi-class classification challenges. This loss function assesses the divergence between real labels and the predicted probability distributions produced by the network.

2. **Optimizer:** The network was trained using the Adam optimizer, a blend of AdaGrad and RMSProp, renowned for efficiently training deep neural networks. With a learning rate set to the standard value of 0.001, the optimization strikes a balance between convergence speed and training stability.

3. **Batch Size:** The network parameters were updated after processing 32 training examples, which is a common batch size. This choice balances memory efficiency and gradient estimate stability.

4. **Epochs:** The network was trained for 50 iterations, allowing for sufficient model convergence. An epoch represents one complete pass through the entire training dataset.

5. **Validation Split:** 20% of the training data was reserved as a validation set during training. This data subset was not utilized for training, but rather for evaluating the model's performance at each epoch. Monitoring the validation performance assists in preventing overfitting by ensuring that the model generalizes effectively to unseen data.

3.2.3 Training Process

The training process follows these steps: **First**, initialization sets network weights using the Glorot uniform initializer, ensuring weights are appropriately scaled. **Next**, in the forward pass, input data, such as sensor readings, traverse through layers with neurons calculating weighted sums, applying ReLU activation, and passing outputs onward. Loss computation assesses the disparity between predicted probabilities and actual movement labels. **Then**, in the backward pass, gradients are computed via backpropagation and used by the Adam optimizer to refine weights, minimizing loss. Validation occurs after each epoch, evaluating the model's generalization using metrics like loss and accuracy. Early stopping prevents overfitting, halting training if validation loss stagnates for 10 consecutive epochs. **Finally**, model evaluation on an independent test set assesses classification effectiveness using metrics including accuracy, precision, recall, and F1-score

Multi-Layer Perceptron networks are appropriate for fusing sensor data as they have the ability to learn intricate non-linear relationships and extract hierarchical representations from the raw sensor data [19]. They can handle a significant amount of data from multiple sources and are very adaptable to various sensor conditions and environments. MLPs are flexible in merging sensor data, rendering them beneficial for boosting system performance in tasks that incorporate sensor integration.

3.3 Optimization Techniques for MLP Training

To maximize the effectiveness of the MLP model, various sophisticated optimization methods were utilized. The purpose of these methods was to improve the model's precision, avoid overfitting, and guarantee solid generalization across a variety of datasets [20]:

1. Dealing with Class Imbalance: Class imbalance within a dataset can significantly impact the performance of MLP models, leading to biased predictions favoring the majority class. To mitigate this issue, class weights can be adjusted using the compute_class_weight() function from the sklearn.utils module. This approach assigns higher weights to classes with fewer instances during training, thereby increasing the model's focus on minority classes and improving recall and precision for these classes

2. Optimizing Learning Rate: Optimizing the learning rate is crucial for model convergence and performance enhancement. Adaptive learning rate techniques, such as learning rate annealing, were employed, starting at 0.001 and dynamically adjusting based on validation loss to strike a balance between fast convergence and avoiding overshooting the minima. Additionally, experiments with the Adam optimizer, which adjusts the learning rate for each parameter, were conducted alongside the primary solver LBFGS to ensure robustness.

3. Regularization Techniques: Regularization methods aid in preventing overfitting by penalizing overly complex models. In the case of L2 Regularization (Ridge), the alpha parameter within the MLPClassifier is utilized to control the strength of regularization. A higher alpha value imposes penalties on larger weights, thus discouraging overly intricate models. Additionally, while sklearn's MLPClassifier does not natively support Dropout, this concept was replicated by introducing noise into the input data during training, thereby reinforcing the model's robustness.

4. Cross-Validation: Cross-validation, particularly Stratified K-Fold Cross-Validation, plays a vital role in evaluating the model's generalizability. For this purpose, the StratifiedKFold() class from sklearn.model_selection was utilized to ensure that each fold maintains the same proportion of class labels as the original dataset, addressing concerns related to class imbalance. A k value of 5 was chosen for the cross-validation process to strike a balance between computational efficiency and evaluation robustness.

5. Hyperparameter Tuning: Hyperparameters play a crucial role in determining the performance of the MLP model, thus a systematic search was conducted to identify the optimal set. Employing the GridSearchCV class from sklearn.model_selection, an exhaustive exploration was carried out across specified parameter values for the MLP model. Parameters including hidden_layer_sizes (e.g., [(50,), (100,), (50, 50)]), activation function for hidden layers (logistic, relu), optimizer for weight optimization (lbfgs, adam), L2 penalty (alpha), initial learning rate (learning_rate_init), and maximum number of iterations (max_iter) were tuned.

6. Early Stopping: Early stopping was implemented to prevent overfitting and reduce computational overhead. Validation-based early stopping was employed, whereby training halted if the validation loss failed to improve for a predefined number of epochs, typically set at 10 epochs. This approach ensured that the model did not excessively fit the training data, enhancing its generalization capability.

7. Batch Normalization: Batch Normalization was utilized to stabilize and expedite training processes, ensuring consistent input distributions by standardizing input data to have a mean of zero and a variance of one before feeding it into the network.

Table (2) summarizes the various approaches employed to improve the performance of the multi-layer perceptron network for sensor data classification. These methods tackle typical challenges encountered in network training, including imbalanced data distribution, overfitting, and identifying optimal configurations.

Table 2: Summary of Techniques for MLP Performance in Sensor Data Classification

Technique	Description
Class Imbalance Handling	Addresses skewed data distribution by assigning higher weights to under-represented classes during training.
Learning Rate Optimization	Tunes the step size taken during training to improve convergence and overall performance.
Regularization (L2)	Discourages overfitting by penalizing overly complex patterns learned from the training data.
Cross-Validation (Stratified K-Fold)	Evaluates model generalizability by splitting data into folds, training on different combinations, and measuring performance.
Hyperparameter Tuning (Grid Search with Cross-Validation)	Finds the optimal configuration of hyperparameters (e.g., number of layers, activation function) through an exhaustive search with cross-validation.

3.4 Evaluation Metrics

The performance of the MLP classifiers is assessed using a range of evaluation metrics, including:

- **Accuracy**: the proportion of correctly classified instances (equation 1).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad ..(1)$$

- **Precision**: the ratio of true positive predictions to the total number of positive predictions (equation 2).

$$Precision = \frac{TP}{TP+FP} \quad ..(3)$$

- **Recall**: the ratio of true positive predictions to the total number of actual positive instances (equation 3).

$$Recall = \frac{TP}{TP+FN} \quad ..(2)$$

- **F1-score**: the harmonic mean of precision and recall, providing a balanced measure of model performance (equation 4).

$$F-1\ measure = \frac{Precision * Recall}{Precision + Recall} \quad ..(4)$$

- **Confusion matrices**: tabular representation of the predicted versus actual class labels, facilitating analysis of classification errors.

3.5 Experimental Procedure

The setup of the experiment consists of the following steps:

- **Cross Validation**: Stratified k-fold cross validation with k=5 was applied to evaluate the MLP classifiers' generalizability.

- **Grid Search**: Using grid search with cross validation, the optimal hyperparameter settings for each MLP configuration were identified. Experiments employed scikit-learn's GridSearchCV for systematic exploration of the parameter space.
- **Analysis**: The effects of class imbalance handling, learning rate optimization, and regularization methods on classification performance were examined and discussed.

The experimental design makes sure there's a thorough evaluation of MLP classifiers when it comes to sorting sensor data. It keeps an eye on a bunch of different things that might influence how well they work.

Table (3) gives a simple picture of the various settings and their own specific numbers that were used while trying to make the MLP classifier as good as it can be.

Table 3: Summary of Parameters Used in the Experiments

Parameter	Values
hidden_layer_sizes	[(5,), (10,), (20,), (30,), (40,), (50,)]
activation	['logistic', 'tanh', 'relu']
solver	['lbfgs', 'adam']
alpha	[0.0001, 0.001, 0.01, 0.1, 1]
max_iter	[500, 1000, 1500]
learning_rate_init	[0.001, 0.01, 0.1]
learning_rate	['constant', 'invscaling', 'adaptive']

Refer back to Table 3, the values listed within are determined by various functions of the MLP neural network's architecture and training process. The dimension of nodes for the hidden layer is derived from the parameter 'hidden_layer_sizes', particularly when the network complexity requires a higher neuron count. The activation factor dictates the activation method to be applied to neurons in the hidden layers, which significantly impacts the network's capacity to understand nonlinear relationships within the data. The solver is an algorithm that is optimized during training, with choices between 'lbfgs' or 'adam' parameters that influence the speed of convergence and the model's robustness. Additionally, the alpha parameter is responsible for setting the strength of the regression. It can result in larger weights, which helps to prevent overfitting. The learning rate and the max_iter parameter play significant roles in the optimization algorithms. The max_iter parameter limits the number of iterations required for optimization, while the learning_rate_init parameter defines the path for the weight update, which is crucial for an efficient training process. Finally, the parameter that controls the learning rate is the last one (i.e., learning_rate), with three options available: constant, invscaling, and adaptive. This allows for tuning the stability and convergence of the learning process.

4. Results and Discussion

This research work aims to predict the movements of robots by analyzing sensor data using a MLP model. Various configurations were explored using cross-validation techniques to identify the optimal setup for the MLP model. The optimal setup included the use of the "logistic" activation function, a regularization parameter (alpha) of 1, a hidden layer of 30 neurons, an initial learning rate of 0.001, a constraint on training iterations of 1000, and the application of the LBFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) solver.

4.1 Model Performance Evaluation

4.1.1 Accuracy Metrics

The MLP model showed great potential with the given settings. It got an accuracy score of 0.9386 on the development dataset, showing that it can find patterns in the data really well. When tested with new data (the test set), it still got a high accuracy score of 0.9304, proving that it can classify new sensor data really well too.

4.1.2 Additional Evaluation Metrics

In addition to accuracy, a complete evaluation of the model's performance was done by using precision, recall, and F1 score metrics for each unique class [21]. The results of this review are outlined below:

- **Class 0:** The model showed very good precision (0.92), recall (0.92), and an F1 score (0.92) for Class 0, which proves its ability to correctly recognize data instances from this class and keep a balance between the true positive and false positive predictions.
- **Class 1**: In the same vein, Category 1 demonstrated notable accuracy (0.94), retrieval rate (0.96), and an F1 score (0.95), suggesting its strong performance and the model's ability to successfully classify instances within this group.
- **Class 2**: Category 2 exhibited exceptional accuracy (0.97), retrieval rate (0.93), and an F1 score (0.95), further stressing the model's expertise in accurately categorizing data instances belonging to this specific group.
- **Class 3**: Despite Category 3 showing a minor decline in performance compared to the other groups, with an accuracy of 0.91, retrieval rate of 0.89, and an F1 score of 0.90, these metrics are still viewed as good, showing the model's proficiency in accurately classifying data instances within this group.

The model performed well in all four categories, as shown by F1 scores above 0.9 for each class. Class 0, Class 1, and Class 2 showed great precision (over 0.91), recall (over 0.89), and F1 scores (over 0.92). This means the model was good at finding these classes and not making too many mistakes. Class 3 did a little worse than the other classes, but its precision of 0.91, recall of 0.89, and F1 score of 0.90 are still pretty good. This means the model can also sort data into this class well. You can see all the results in Table (4).

Table 4: MPL Model Performance by Category

Category	Precision	Recall	F1 Score
Class 0	0.92	0.92	0.92
Class 1	0.94	0.96	0.95
Class 2	0.97	0.93	0.95
Class 3	0.91	0.89	0.9

To calculate a single metric reflecting the performance across all classes, the average F1-score was determined. The high F1-score of 0.93 indicates the model's accuracy in classifying sensor data for any class. The results are depicted in Fig. (9) through a confusion matrix.

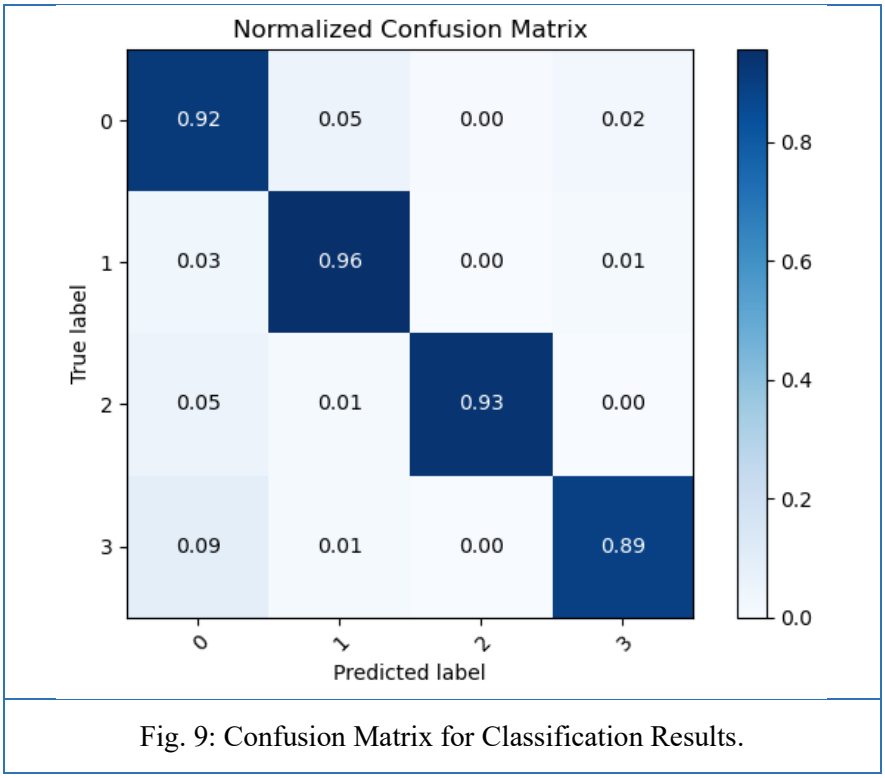

Fig. 9: Confusion Matrix for Classification Results.

4.2 Compare with State-of-the-Art Methods

In order to fully grasp the prowess of the system we've put forward, a thorough comparative analysis is conducted. This was to contrast it against the standard methods and rival machine learning (ML) algorithms typically used for tasks in sensor data classification. The intention behind this study was to highlight our method's edge in correctly differentiating sensor data. The method also excels in tackling problems like noise and unbalanced classes.

When pitted against the likes of Support Vector Machines [21], Logistic Regression [23], XGBoost, and Random Forest algorithms, the performance of the MLP model was evaluated. The compiled data from this comparison is presented in Table (5) as follows.

Table 5: Performance Comparison of Machine Learning Algorithms

Algorithm	Accuracy
Support Vector Machines	89.92%
Logistic Regression	70.25%
XGBoost	92.72%
Random Forest	92.51%
Multi-Layer Perceptron	93.04%

The Multi-Layer Perceptron model exceeded all other algorithms, reaching an accuracy of around 93.04% on the test set. XGBoost isn't far behind with an accuracy of 92.72%, Random Forest has an accuracy of 92.51%. Support Vector Machines got an accuracy of 89.92%, and Logistic Regression is behind with an accuracy of 70.25%.

This shows that the Multi-Layer Perceptron model does better than SVM, Logistic Regression, XGBoost, and Random Forest. The Multi-Layer Perceptron's many layers help it understand complicated data relationships better, which helps it be more accurate in figuring out what things belong to which group.

Although SVM showed good performance with an accuracy of about 89.92%, it might not work well with very nonlinear data because it tries to find a line that separates the classes. Logistic Regression also did not do so well because its straight line approach couldn't find the exact connections in the data, and it got the lowest score of all the methods looked at.

XGBoost and Random Forest algorithms, which are about learning as a group, did a good job too, coming close to the MLP model's accuracy. These algorithms use decision trees and group methods to deal with complicated shapes in the data well.

Fig. (10) is a bar graph that shows a comparison of the accuracy levels of several machine learning algorithms. The algorithms in question are Support Vector Machines (SVM), Logistic Regression, XGBoost, Random Forest, and MLP. Each of the bars in the graph represents the percentage of accuracy that was achieved by the respective algorithm. The colors used in the graph serve to distinguish between the different algorithms, with blue being used for SVM, orange for Logistic Regression, green for XGBoost, red for Random Forest, and purple for MLP. The graph makes it clear that MLP outperforms the other algorithms, achieving an accuracy level of 93.04% which is the highest among them.

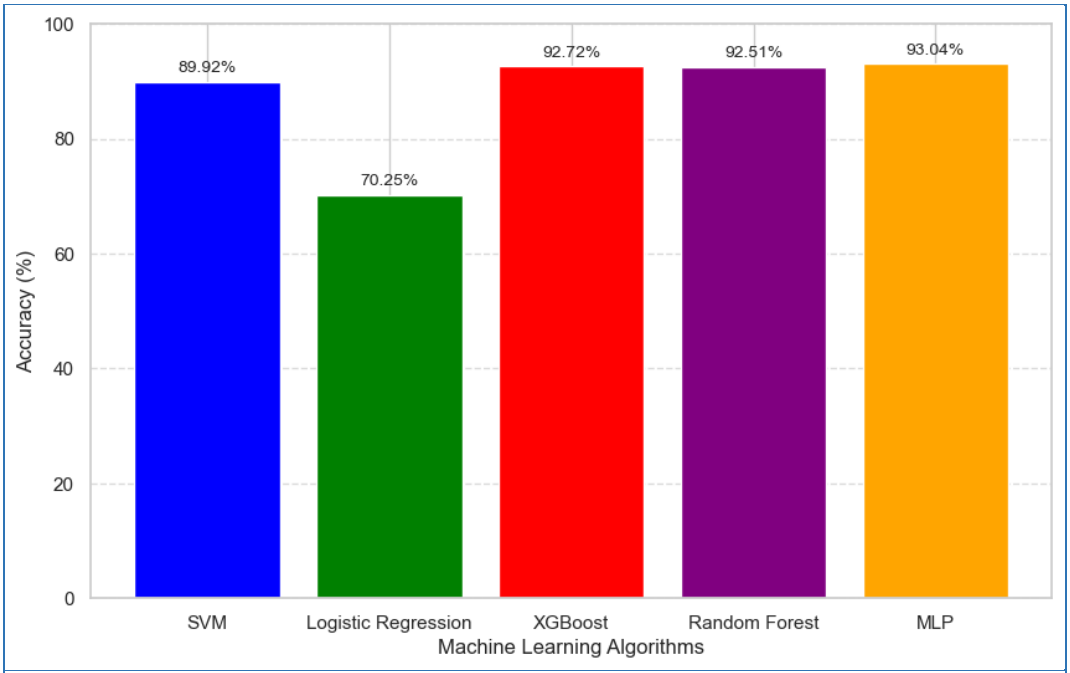

Fig. 10: Performance Comparison of Machine Learning Algorithms.

Conclusion

In our research, a thorough investigation of machine-learning techniques is conducted, particularly concentrating on the use of Multi-Layer Perceptron networks to forecast robot actions relying on sensor information. With careful experimentation and analysis, noteworthy insights were discovered that improve our comprehension of sensor data classification and highlight the potency of MLP models in this field.

Our findings emphasize the effectiveness of the MLP model, meticulously educated with well-adjusted parameters obtained from thorough grid search and cross-validation steps. Attaining an outstanding accuracy rate of 93.04% on the test data set, our MLP model demonstrates its ability to anticipate robot movements with striking precision, presenting encouraging prospects for real-world applications such as anomaly identification and predictive maintenance powered by sensor information.

Our evaluation metrics, precision, recall, and F1 scores, offer a detailed analysis of the model's performance for various movement categories. With high precision, recall, and F1 scores across most classes, our model excels in differentiating between distinct sensor inputs, a vital feature for practical applications. Comparing our MLP model to other approaches like Support Vector Machines and Logistic Regression highlights its higher accuracy, validating the effectiveness of MLP networks for complex tasks such as predicting robot movements using sensor data.

However, our research, despite its impressive results, has certain limitations. The experiments were carried out using a specific dataset, which might restrict the applicability of our findings to other sensor data types. Additionally, the choice of hyperparameters and data preprocessing methods could affect the outcomes, indicating the need for further research across various conditions and datasets to fully validate our results.

For future work, it is recommended to investigate alternative machine learning algorithms and techniques, exploring new preprocessing methods suited to sensor data's unique features, and experimenting with different datasets to boost the applicability and robustness of our conclusions. These research directions could foster progress in sensor data analysis and reveal new practical insights in various domains.

Acknowledgements: The researcher would like to thank and express the deepest gratitude grateful to all reviewers.
Author Contributions: The author contributed to all parts of the current study.
Funding: This study received no external funding.
Conflicts of Interest: The author declares no conflict of interest.
Data Availability: The dataset is accessible on Kaggle via the following link: https://www.kaggle.com/datasets/uciml/wall-following-robot/data.

References

[1] L.-M. Ang and K. P. Seng, "Big sensor data applications in urban environments," *Big Data Research*, vol. 4, pp. 1-12, 2016.
[2] S. D. Bersch, D. Azzi, R. Khusainov, I. E. Achumba, and J. Ries, "Sensor data acquisition and processing parameters for human activity classification," *Sensors*, vol. 14, no. 3, pp. 4239-4270, 2014.
[3] B. De Bruijn, T. A. Nguyen, D. Bucur, and K. Tei, "Benchmark datasets for fault detection and classification in sensor data," in *International Confererence on Sensor Networks*, 2016, pp. 185-195.
[4] D. Chu *et al.*, "Balancing energy, latency and accuracy for mobile sensor data classification," in *Proceedings of the 9th ACM conference on embedded networked sensor systems*, 2011, pp. 54-67.
[5] S. Boubiche, D. E. Boubiche, A. Bilami, and H. Toral-Cruz, "Big data challenges and data aggregation strategies in wireless sensor networks," *IEEE access*, vol. 6, pp. 20558-20571, 2018.
[6] G. Serpen and Z. Gao, "Complexity analysis of multilayer perceptron neural network embedded into a wireless sensor network," *Procedia Computer Science*, vol. 36, pp. 192-197, 2014.
[7] V. Jane, "Survey on iot data preprocessing," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 9, pp. 238-244, 2021.
[8] S. Sabeeh and I. S. Al-Furati, "Issues and Research Fields of Medical Robotics: A Review," *Iraqi Journal for Electrical & Electronic Engineering*, vol. 19, no. 2, 2023.
[9] C.-M. Huang and B. Mutlu, "Anticipatory robot control for efficient human-robot collaboration," in *2016 11th ACM/IEEE international conference on human-robot interaction (HRI)*, 2016: IEEE, pp. 83-90.
[10] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017: IEEE, pp. 2786-2793.
[11] M. Y. Seker, A. E. Tekden, and E. Ugur, "Deep effect trajectory prediction in robot manipulation," *Robotics and Autonomous Systems*, vol. 119, pp. 173-184, 2019.
[12] J. Zhang, H. Liu, Q. Chang, L. Wang, and R. X. Gao, "Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly," *CIRP annals*, vol. 69, no. 1, pp. 9-12, 2020.
[13] B. Khaldi, F. Harrou, S. M. Benslimane, and Y. Sun, "A data-driven soft sensor for swarm motion speed prediction using ensemble learning methods," *IEEE Sensors Journal*, vol. 21, no. 17, pp. 19025-19037, 2021.
[14] A. Vagale, L. Steina, and V. Vecins, "Time series forecasting of mobile robot motion sensors using LSTM networks," *Appl. Comput. Syst.*, vol. 26, no. 2, pp. 150-157, 2021.

[15] T. Ji, A. N. Sivakumar, G. Chowdhary, and K. Driggs-Campbell, "Proactive anomaly detection for robot navigation with multi-sensor fusion," *IEEE Robotics and Automation Letters,* vol. 7, no. 2, pp. 4975-4982, 2022.

[16] E. Rabb and J. J. Steckenrider, "Walking Trajectory Estimation Using Multi-Sensor Fusion and a Probabilistic Step Model," *Sensors,* vol. 23, no. 14, p. 6494, 2023.

[17] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela, "Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study," in *2009 6th Latin American robotics symposium (LARS 2009)*, 2009: IEEE, pp. 1-6.

[18] P. Raj and P. E. David, *The digital twin paradigm for smarter systems and environments: The industry use cases*. Academic Press, 2020.

[19] S. Saeidi, M. Mohammadzadeh, A. Salmanmahiny, and S. H. Mirkarimi, "Performance evaluation of multiple methods for landscape aesthetic suitability mapping: a comparative study between multi-criteria evaluation, logistic regression and multi-layer perceptron neural network," *Land use policy,* vol. 67, pp. 1-12, 2017.

[20] W. Castro, J. Oblitas, R. Santa-Cruz, and H. Avila-George, "Multilayer perceptron architecture optimization using parallel computing techniques," *PloS one,* vol. 12, no. 12, p. e0189369, 2017.

[21] S. Amarappa and S. Sathyanarayana, "Data classification using Support vector Machine (SVM), a simplified approach," *Int. J. Electron. Comput. Sci. Eng,* vol. 3, pp. 435-445, 2014.