**Abstract: This subsequent research outlines an elaborate strategy for improving the accuracy of echo detection in ultrasonic sensors. By integrating techniques such as Convolutional Neural Networks (CNN), Random Forests, and XGBoost, this study proposes a diverse machine learning approach to better identify the initial echo signal. Utilizing the Red Pitaya board along with the Ultrasonic Sensor SRF02, our attention is centered on dataset #3 to test our proposed methods. This document elaborates on the software solutions deployed and examines the effectiveness of each algorithm, signifying a notable progress in the field of echo detection technology.**

Reliability test and improvement of a sensor system for object detection

Course Information Technology

Modules Autonomous Intelligent Systems and Machine Learning

By Dr. Peter Nauth and Dr. Andreas Pech

|  |  |
| --- | --- |
| Richa Padhi  Mat No: 1438260 | Theertha Bharathan  Mat No: 1445457 |
| richa.padhi@stud.fra-uas.de | theertha.bharathan@stud.fra-uas.de |

Reliability test and improvement of a sensor system for object detection

Course Information Technology

Modules Autonomous Intelligent Systems and Machine Learning

By Dr. Peter Nauth and Dr. Andreas Pech

|  |  |
| --- | --- |
| Deblina Karmakar  Matriculation: 1427365 | Indranil Saha  Matriculation: 1427190 |
| deblina.karmakar@stud.fra-uas.de | indranil.saha@stud.fra-uas.de |

**Keywords—Echo Detection, Ultrasonic Sensing, Machine Learning, Convolutional Neural Networks, Random Forest, XGBoost, Precision Enhancement**

# INTRODUCTION

In this study, we delve deeper into the use of advanced machine learning techniques to improve the precision of detecting the initial echo in ultrasonic sensor systems, building on the foundation laid by our earlier work. We adopt a comprehensive approach that integrates a variety of machine learning models, each chosen for their unique strengths and synergistic potential. Convolutional Neural Networks (CNN) are utilized for their superior ability in processing signals, crucial for identifying patterns in complex data sets. Random Forests are selected for their robustness in classification, ensuring reliable performance across different scenarios. Furthermore, XGBoost is employed for its efficiency and effectiveness in predictive modeling, offering a rapid and powerful solution for analysis.

Our objective is to set new benchmarks in the accuracy of echo detection, which is vital for various applications, from autonomous vehicles to sophisticated safety systems. Our methodology focuses on improving the precision of detecting the first echo by meticulously identifying the highest peak within specific time frames. These frames are intelligently defined by our machine learning models, allowing for the precise localization of the echo's most significant aspect with unprecedented accuracy.

To test and validate our advanced detection techniques, we turn our attention to dataset #3, which will act as a critical testbed for our refined algorithms. Here, we expect to see marked enhancements in detection performance. Through continuous experimentation and improvement, our research aims to significantly advance the field, exploring new frontiers in ultrasonic sensor technology and echo detection capabilities.

# METHODOLOGY

The approach undertaken leverages a two-tiered strategy that initially employs machine learning to determine specific time windows, followed by a focused analysis to identify peak signals. This section delves into the implementation details of the CNN, Random Forest, and XGBoost models, shedding light on their training processes, evaluation methods, and their deployment in pinpointing echo signals.

## Convolutional Neural Networks (CNN):

Convolutional neural network (CNN) is a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization. Vanishing gradients and exploding gradients, seen during backpropagation in earlier neural networks, are prevented by using regularized weights over fewer connections[8][9]. Below figure 4 displays a typical CNN architecture consisting of different layers.

A diagram of a diagram of a diagram

Description automatically generated

Fig.1. A typical CNN Architecture

The working of the CNN model can be described as per the following points –

• Convolutional Neural Networks (CNNs) are composed of layers containing artificial neurons or nodes, which perform calculations to produce activation maps, identifying key features in input data, such as images.

• Each node calculates a weighted sum of its inputs and applies an activation function to generate an output, focusing on specific features like edges or textures in the initial layers.

• Through a process known as convolution, these networks can identify and emphasize important visual features by applying filters that detect patterns, edges, and other significant elements.

• As data progresses through the layers, CNNs use max pooling to reduce dimensionality and focus on the most relevant features, ensuring that subsequent layer’s work with the most informative aspects of the data.

• The final layers of a CNN typically involve classification, where the network makes predictions or identifications based on the processed features, assigning input data to predefined categories.

• Training a CNN involves adjusting the weights of the nodes based on the accuracy of the output compared to known labels in a training dataset, with careful attention to avoid overfitting, ensuring the model generalizes well to new, unseen data.

## Random Forest

A Random Forest is like a group decision-making team in machine learning. It combines the opinions of many “trees” (individual models) to make better predictions, creating a more robust and accurate overall model. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification. It performs better for classification and regression tasks. The steps involved in the algorithm of Random Forest are as follows –

• In the Random Forest model, a subset of data points and a subset of features is selected for constructing each decision tree. Simply put, n random records and m features are taken from the data set having k number of records.

• Individual decision trees are constructed for each sample.

• Each decision tree will generate an output.

• Final output is considered based on Majority Voting or averaging for Classification and regression, respectively.

1. *XGBoost*

XGBoost is a machine learning algorithm that belongs to the

ensemble learning category, specifically the gradient boosting framework. It utilizes decision trees as base learners and employs regularization techniques to enhance model generalization. Known for its computational efficiency, feature importance analysis, and handling of missing values, XGBoost is widely used for tasks such as regression, classification, and ranking.

In contrast to bagging techniques like Random Forest, in which trees are grown to their maximum extent, boosting makes use of trees with fewer splits. Such small trees, which are not very deep, are highly interpretable. Parameters like the number of trees or iterations, the rate at which the gradient boosting learns, and the depth of the tree, could be optimally selected through validation techniques like k-fold cross validation.

Among the various ensemble techniques, the gradient boosting consists of majorly three steps –

• An initial model F0 is defined to predict the target variable y. This model will be associated with a residual (y-F0).

• A new model h1 is fit to the residuals from the previous step.

• Now, F0 and h1 are combined to give F1, the boosted version of F0. The mean squared error from F1 will be lower than that from F0.

Performance of F1 is improved, by modelling the residuals of F1 and creating a new model F2.

This can be done for ‘m’ iterations, until residuals have been minimized as much as possible.

In this method, the additive learners do not disturb the functions created in the previous steps. Instead, they impart information of their own to bring down the errors.

# IMPLEMENTATION

1. *Implementation Workflow*

The implementation is outlined using a flowchart, as shown in the below for a machine learning model starting with the collection and reading of data. This data then undergoes a series of transformations including cleaning and normalization, preparing it for further analysis. Key stages highlighted include the identification of significant data points through peak analysis and the subsequent organization and labeling of data, crucial steps for pattern recognition and categorization. Following the data preparation, the process involves defining the model tailored to the data’s characteristics and the analytical goals. This model is trained with the organized data, learning to identify patterns and make predictions. Upon reaching satisfactory performance levels, the model is saved, allowing for future reuse or deployment, thus encapsulating a streamlined approach from data collection to model readiness for predictive tasks.

Load Dataset

Signal Processing

Peak Analysis and Distance calculation

Distance Labelling and Grouping

Define the Model

Train and save the model

Predicted Value

1. *Pre-Processing*

Pre-Processing of the raw data from the ultrasonic sensor is done in several stages. They are mentioned as follows-

Extraction of the relevant information from the obtained dataset: The ‘read\_and\_prepare\_data’ function plays a pivotal role in the data processing pipeline by loading signal data from a CSV file, with a particular focus on columns that contain pertinent information. This initial step is crucial for the preparation of raw ultrasonic signals, setting the stage for in-depth analysis. It involves filtering and extracting the most relevant data from the multitude of available information, ensuring that only the data of interest is forwarded for subsequent processing. This targeted approach helps streamline the analysis, making it more efficient and focused on the signals that are most likely to yield valuable insights.

Signal Windowing: The ‘apply\_window’ function serves as a critical step in signal processing, specifically designed to address and mitigate spectral leakage—a common issue that can distort frequency analysis. By applying a Hanning window to each signal, this process effectively shapes the data, tapering the beginning and end of each signal. This tapering is essential for smoothing transitions and minimizing abrupt changes, which, in turn, significantly enhances the accuracy and reliability of frequency analysis. The level of precision is crucial for the accurate detection of echoes, a fundamental aspect of analyzing ultrasonic signals, ensuring that the data analysis leads to trustworthy and actionable insights.

Noise Reduction and Peak Detection: In the ‘reduce\_noise\_and\_label’ process, the initial step involves converting each signal from its time domain to the frequency domain through the Fast Fourier Transform (FFT). This transformation is pivotal for identifying and isolating the signal's frequency components. Following this, a Power Spectral Density (PSD) threshold is applied as a filter to effectively eliminate background noise, sharpening the focus on relevant signal components. The process then employs an inverse FFT to revert the filtered signal back to the time domain, but with significantly reduced noise. To further refine the signal for analysis, the Hilbert Transform is applied to the denoised signal to extract its envelope. This envelope is crucial for the effective detection of peaks using the ‘find\_peaks’ function, enabling more precise identification of significant features within the signal.

Labelling the data: This specialized function is adept at taking a compilation of peak positions identified within signal data and systematically arranging these observations into an organized structure. This structure not only highlights the occurrence of peaks but also delineates their presence within specific temporal segments or windows across a multitude of signals. The function embarks on this task by first ascertaining the total number of distinguishable windows present within a signal. This determination is made by dividing the entire length of the signal by the predefined width attributed to each window, thereby establishing a count for the possible windows.

Following the calculation of windows, the function proceeds to construct a matrix, referred to as ‘y\_label’. This matrix is designed to encapsulate binary labels that correspond to each signal across the entirety of the windows, effectively indicating the presence or absence of peaks within each window. The dimensions of this matrix are carefully defined, with the number of rows equaling the total number of signals—this is inferred from the length of the list containing peak positions (peaks\_list). Concurrently, the number of columns within the matrix matches the previously calculated number of windows (n\_windows). Through this process, the function achieves a comprehensive overview, mapping out the distribution and incidence of peaks across various signals and temporal windows, thereby facilitating a structured analysis of signal characteristics.

The described function meticulously processes a list of detected peak positions within signal data, structuring these observations in a way that clearly marks the occurrence of peaks across distinct time frames within multiple signals. To achieve this, it begins by calculating the total number of potential time windows for any given signal, a figure derived by dividing the signal's total duration by the designated width of each time window.

For every individual signal under analysis, the function then scrutinizes each identified peak position. When a peak is found (signified by a position value that is not negative), it calculates the specific time window that this peak corresponds to by dividing the peak's position by the width of the window. This calculation determines exactly where within the temporal structure of the signal the peak occurs.

Upon determining the correct time window for a peak, the function updates a specially designed matrix, known as the label matrix, setting the element that corresponds to this signal and window to 1. This action signals the presence of a peak within that precise window, creating a binary map of peak occurrences. Conversely, time windows that do not house any peaks are marked with a 0, clearly indicating a lack of peak activity in those intervals.

*Training the Model*

## Convolutional Neural Networks (CNN)

The function is designed for the training of a Convolutional Neural Network (CNN), which is adept at handling data characterized by spatial relationships, such as images or time-series datasets. The process unfolds in several meticulously structured steps:

**Initialization Phase**: Training parameters are established, encompassing the verbosity level, the total number of training epochs, and the specified batch size.

**Model Definition:**

• A linear sequence of layers is constructed using ‘Sequential()’ function.

• For convolution operations on one-dimensional data, Conv1D layers are integrated, each configured with 64 filters and a kernel size of 3.

• To mitigate the risk of overfitting, a Dropout layer is applied, setting the rate at 0.5.

• Dimensionality reduction is achieved through MaxPooling1D, enhancing the model's ability to generalize.

• A Flatten layer is utilized to transform the 2D feature maps into a 1D vector, preparing the data for dense layer processing.

• The architecture includes Dense layers, which are fully connected, with the final layer employing softmax activation to calculate probabilities across various classes.

**Model Compilation:** In this step, the model undergoes compilation with the categorical cross entropy loss function and the Adam optimizer, setting the groundwork for training.

**Training:** The model is subjected to training using the provided datasets (x\_train, y\_train), adhering to the previously set epochs and batch size.

**Evaluation:** Post-training, the model's performance and accuracy are rigorously evaluated against the test dataset.

**Model Saving:** Subsequent to evaluation, the fully trained model is preserved on the file system, facilitating future access and utilization.

**Return:** Ultimately, the function concludes by returning the trained model, ready for deployment.

## Random Forest

The Random Forest algorithm is a powerful ensemble learning technique known for its versatility and ease of use. It builds upon the concept of decision trees, aggregating multiple such trees to form a "forest" that enhances the predictive accuracy and controls over-fitting. The approach towards deploying a Random Forest model involves several critical stages, each contributing to the model's overall effectiveness:

**Parameter Grid Construction:** The first step in optimizing a Random Forest model involves setting up a comprehensive grid of hyperparameters. This grid encompasses a variety of parameters crucial for the model's performance, including the number of trees in the forest (n\_estimators), the maximum number of features considered for splitting a node (max\_features), the maximum depth of each tree (max\_depth), the minimum number of samples required to split a node (min\_samples\_split), the minimum number of samples necessary at a leaf node (min\_samples\_leaf), and the decision to use bootstrap samples for building trees (bootstrap). This grid serves as the foundation for the subsequent search for the optimal set of parameters.

**Randomized Hyperparameter Search:** To efficiently navigate the vast parameter space outlined by the grid, a ‘RandomizedSearchCV’ is employed. This method diverges from exhaustive search techniques by randomly sampling from the parameter space, offering a balance between exploration and resource consumption. The output of this phase is the identification of the best hyperparameter combination, crucial for constructing the most effective Random Forest model.

**Model Training with Best Parameters:** Equipped with the optimal set of parameters discovered in the previous step, a new Random Forest model is instantiated. This model is then trained on the entire training dataset, leveraging the best hyperparameter values to ensure superior learning and generalization capabilities.

**Model Preservation:** Upon successful training, the refined model is preserved onto the file system, typically in a pickle file format. This step ensures that the model can be easily retrieved and deployed for future predictions without the need for retraining.

**Prediction and Performance Evaluation:** The trained model is then put to the test, predicting outcomes for both the training and test datasets. The effectiveness of these predictions is quantified using the F1 score, a balanced metric considering both precision and recall, for each set of predictions. These scores are crucial indicators of the model's performance and its ability to generalize beyond the training data.

**Return of the Trained Model:** The culmination of this process is the return of the trained Random Forest model, now fine-tuned with the best possible parameters. This model stands ready for application in predictive tasks, equipped with the ability to provide high-quality predictions.

## XGBoost

XGBoost stands as a prominent and efficient gradient boosting library that has garnered acclaim for its performance on structured or tabular data across various machine learning competitions and applications. Implementing an XGBoost model involves a series of meticulously planned steps, each designed to maximize the model's predictive power and efficiency:

**Model Initialization:** The process starts with the initialization of an XGBoost classifier. This step involves configuring a set of hyperparameters that define the model's structure and learning process. Key parameters include the objective function, which is often set to binary:logistic for binary classification tasks, indicating the model's goal to predict a binary outcome. Additionally, the number of estimators specifies how many boosting rounds or trees the model shall be built. The maximum depth of trees controls the complexity of the model, the learning rate dictates the step size at each iteration to prevent overfitting, and the subsample ratio helps in reducing variance by training on a subset of the data at each step.

**Model Training:** With the classifier initialized, the model is then trained by fitting it to the training data. This process involves iteratively building trees, each designed to correct the errors of its predecessors, thereby continuously improving the model's accuracy with each round. The training phase is crucial, as it allows the model to learn from the structure and patterns within the data, adjusting its parameters to minimize prediction errors.

**Model Saving:** After the model has been trained and its parameters optimized, it is saved to the file system, typically in a pickle file. This serialization step ensures that the model can be stored and later retrieved without the need to retrain it from scratch, facilitating easy deployment in predictive applications or further analysis.

**Evaluation:** The model's effectiveness is then evaluated by making predictions on both the training and test datasets. The F1 score, which serves as the evaluation metric, is calculated for each set of predictions. The F1 score, being the harmonic mean of precision and recall, provides a more balanced measure of a model's performance, especially in cases of imbalanced datasets where accuracy alone might be misleading.

**Return of the Trained Model:** The final step is the return of the trained XGBoost model. This model, now ready and equipped with the learned parameters, stands as a powerful tool for making predictions on new data, embodying the culmination of the training and optimization efforts.

# RESULT AND ANALYSIS:

Based on training the above 3 models for ADC data for hard objects at 1m. We could see that following results:

## Convolutional Neural Networks (CNN)

The CNN model shows high precision, recall, and F1-score for the majority classes (128, 129, 130), indicating that it is predicting these classes very well. The overall accuracy is 0.905 (90.5% correct predictions). The weighted average of F1-score and recall are both also high at around 0.91, suggesting good performance across all classes while considering the number of instances (support) in each class. A screenshot of a computer

Description automatically generated

Fig.5. Classification report of CNN Model

## Random Forest

The Random Forest model, after hyperparameter tuning, achieved the best performance with 100 trees (n\_estimators), a minimum of 5 samples required to split an internal node (min\_samples\_split), only 1 sample required at a leaf node (min\_samples\_leaf), max\_features set to use the square root of the number of features, a maximum depth of the trees (max\_depth) at 30, and without bootstrapping (bootstrap set to False).This model also performs well on the major classes, with slightly lower performance on class 130 compared to the CNN model. The overall accuracy of this model is higher at 0.925 (92.5% correct predictions). The weighted F1-score and recall are also high and consistent with the overall accuracy at around 0.92.

A screenshot of a computer

Description automatically generatedFig.6. Classification report of Random Forest Model

## XGBoost

The XGBoost model has precision, recall, and F1-scores similar to the Random Forest model for the major classes, with a very slight decrease in recall for class 130. The accuracy of the XGBoost model is 0.865 (86.5% correct predictions), which is lower than the CNN and Random Forest models. The weighted average F1-score and recall are correspondingly lower at around 0.88. A screenshot of a computer

Description automatically generated

Fig.7. Classification report of XGBoost Model

# V. CONCLUSION & FUTURE SCOPE

In conclusion, the comparative analysis of CNN, Random Forest, and XGBoost models on the 1m dataset has provided valuable insights into the strengths and weaknesses of each approach. The CNN model emerged as the top performer, likely due to its ability to harness spatial dependencies within the data—a characteristic that is particularly relevant for signal processing tasks. The Random Forest model also demonstrated commendable accuracy, suggesting that ensemble methods are robust and effective for this class of problems. XGBoost, while slightly trailing, still showed a respectable level of accuracy and remains a competitive option, especially considering its speed and scalability.

*Future Scope*

Moving forward, to enhance the robustness and applicability of the models, the following future directions could be considered:

**Dataset Expansion**: Introducing a broader range of scenarios, including signals reflected from hard objects at greater distances such as 50m, and incorporating data from soft objects with different postures (sitting and standing), would enrich the dataset. This diversity could help improve the model's generalization capabilities and accuracy.

**Model Optimization**: Further tuning of the model hyperparameters, perhaps through more sophisticated methods like Bayesian Optimization, could lead to better performance. Additionally, exploring more complex CNN architectures or advanced ensemble techniques could uncover improvements.

**Feature Engineering**: Developing more intricate features that capture the essence of signal reflections from various surfaces and distances might enhance the models' ability to distinguish between different classes more effectively.

**Class Imbalance Mitigation**: Addressing the class imbalance with techniques such as synthetic minority oversampling (SMOTE) or adaptive sampling could improve the model performance, particularly for underrepresented classes.

**Real-time Processing**: Adapting the models for real-time signal processing could be beneficial for applications that require immediate decisions, such as autonomous driving or active surveillance systems.

**Model Ensemble**: Combining the predictions from multiple models in an ensemble method could yield better performance than any single model, leveraging the strengths of each.

# REFERENCES

[1] Ivan Koudar, Echo detection, <https://patents.google.com/patent/EP2255446A1/en>, 2019

[2] Liao Qiang, Design and Implementation of Digital Ultrasonic Flaw Detector, vol. 4, 2009.

[3] R. C. Luo, S. L. Lee, Y. C. Wen and C. H. Hsu, “Modular ROS Based Autonomous Mobile Industrial Robot System for Automated Intelligent Manufacturing Applications,” 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), [Online]. Available: doi: 10.1109/AIM43001.2020.9158800.

[4] “SRF02 Ultrasonic range finder,” robot-electronics, [Online]. Available: <https://www.robot-electronics.co.uk/htm/srf02tech.htm>

[5] W.T. Cochran; J.W. Cooley; D.L. Favin; H.D. Helms; R.A. Kaenel; W.W. Lang; G.C. Maling; D.E. Nelson; C.M. Rader; P.D Welch, <https://ieeexplore.ieee.org/document/1447887/authors>, 1967

[6] W. Gentleman, G. Sande, Fast Fourier Transforms: for fun and profit, published in AFIPS '66 (Fall), November 1966

[7] Aniruddha Bhandari, Understanding & Interpreting Confusion Matrix in Machine Learning, <https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/>, 2024

[8] Venkatesan, Ragav; Li, Baoxin, Convolutional Neural Networks in Visual Computing: A Concise Guide, 2017

[9] Balas, Valentina E.; Kumar, Raghvendra; Srivastava, Rajshree, Recent Trends and Advances in Artificial Intelligence and Internet of Things, 2019