# Research on hyperparameters for the classification of ultrasonic signals

Course Information Technology
Modules Individual Project
Theertha Bharathan
Mat No:1445457
theertha.bharathan@stud.fra-uas.de

***Abstract*—This study investigates an innovative technique for material classification that integrates ultrasonic signal analysis with machine learning approaches. The research aims to enhance the accuracy and reliability of material identification by analysing ultrasonic signals reflected from various materials, such as aluminium, steel, and different wood types. The methodology involves converting the captured ultrasonic signals into the frequency domain using the Fast Fourier Transform (FFT) technique, and then extracting relevant features to train multiple classification models** [1]**. The study examines the performance of various machine learning techniques, including Logistic Regression, Support Vector Machines (SVM), Random Forest, and Convolutional Neural Networks (CNN), in the task of differentiating between distinct materials** [2]**. In addition, Long Short-Term Memory (LSTM) networks were used to extract temporal patterns from the signal data** [3]**. The proposed system combines sophisticated signal processing methods with resilient classification algorithms to attain high precision in material recognition** [1]**. This study demonstrates the potential of ultrasonic sensing and machine learning together for a range of applications, such as industrial automation, quality assurance, and materials testing. The study intends to improve material categorization systems' capabilities in a variety of industries by utilizing these technologies** [4]**.**

***Keywords—Red Pitaya, Ultrasonic Sensor SRF02, Fast Fourier Transform, Machine Learning, Convolutional Neural Networks, Random Forest, LSTM, Material Classification, Signal Processing.***

## I. INTRODUCTION

In the ever-advancing landscape of technology, sensor systems are becoming integral to numerous industries, from industrial automation to quality control and beyond. One key area where sensor technology plays a crucial role is in the accurate classification and identification of materials, which is vital in fields such as non-destructive testing, manufacturing, and autonomous systems. Ultrasonic sensors, known for their ability to capture precise information about material properties through sound wave reflections, offer a powerful solution for this task. However, achieving reliable material classification requires sophisticated analysis techniques that can accurately interpret the complex data captured by these sensors. Enhancing the accuracy and dependability of sensor systems is a priority in today's rapidly changing technological environment, with applications ranging from industrial automation to vehicle safety.

Sensor systems are fundamentally based on the analysis of ultrasonic waves, which provide important environmental information. These systems use specialized signal processing techniques to extract useful information from reflected signals so that educated decisions may be made, which is essential for safe and effective vehicle operations.

The methodology makes use of a sophisticated workflow for signal processing, which begins with the acquisition of ultrasonic echoes from diverse materials. The Fast Fourier Transform (FFT) is then used to convert these signals into the frequency domain, which allows for efficient feature extraction. After that, a variety of machine learning models use the retrieved features as inputs.

Our sensor system architecture consists of a well-balanced set of hardware and software components, each specifically engineered to fulfil a specific role in the signal processing process. Every part of the system, from the ultrasonic transducer responsible for sending and receiving acoustic signals to the embedded control system managing signal acquisition to the computational unit running complex analysis algorithms contributes to the overall goal of enhancing detection dependability.

Our research endeavours are mostly dependent on the meticulous gathering and examination of empirical data sets. These data sets are thoughtfully selected to cover a broad spectrum of item scenarios and environmental variables. Through an ongoing process of refinement and validation, we try to consistently strengthen the accuracy and reliability of our sensor system outputs, assuring their applicability for real-world applications. The research goes beyond data analysis; in order to uncover deeper insights and capabilities from the sensor system, we also investigate machine learning techniques. Our goal is to create prediction models that can identify complex patterns in the sensor data by utilizing supervised learning techniques, which will ultimately improve the system's ability to discern between various objects with accuracy.

The comprehensive research endeavours culminate in the goal of pushing the limits of ultrasonic sensor technology.

This will open up new opportunities in a variety of industrial applications and make vehicle operations safer and more effective. Our path towards enhancing the precision and reliability of sensor systems is a reflection of our constant commitment to innovation in the field of autonomous systems and intelligent technologies, as we traverse the intricate interplay between theory and practice, hardware and software.

## II. METHODOLOGY

The theoretical background of the experiment is mentioned in the above section which consists of the description of the Ultrasonic Red Pitaya sensor, FFT data analysis, Machine Learning algorithm background.

### *A. Ultrasonic sensor and Red Pitaya Measurement Board*

A test and measurement board called Red Pitaya STEM Lab [5] is based on a system-on-a-chip (SoC) [6] from the former company Xilinx. It may be configured to function as an oscilloscope, spectrum analyser, LCR meter, or network analyser and can be remotely controlled. The Ultrasonic Sensor SRF02 [7], a single transducer ultrasonic rangefinder in a tiny footprint PCB, was utilized in this configuration. The minimum range of the SRF02 is greater than that of other dual transducer rangers since it only employs one transducer for transmission and receiving. The smallest measuring range is approximately 15 cm (6 inches). With a 5V grounded power supply, it can operate. The Red Pitaya device makes it possible to wirelessly transfer data to a laptop for additional processing.

The sensor is operated under the GNU/Linux operating system. On a computer or mobile device, they can be used to manage and record measurements. In addition to 16 standard input and output ports, the main Red Pitaya unit incorporates two analog RF inputs and outputs. A micro-SD card slot, an RJ45 socket for Ethernet, a USB port, and a micro-USB connector for the console are also included on the board. Radio frequency transmissions can both be received and transmitted by the Red Pitaya which operates in the frequency range of 50MHz.

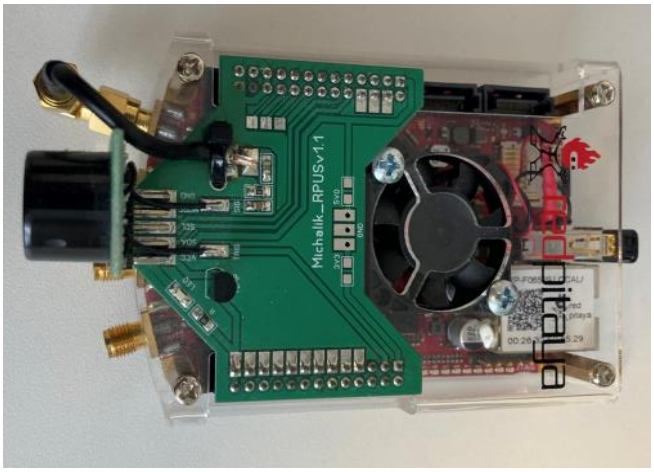


Fig.1. Ultrasonic sensor and red pitaya device

The sensor is operated under the GNU/Linux operating system. On a computer or mobile device, they can be used to manage and record measurements. In addition to 16 standard input and output ports, the main Red Pitaya unit incorporates two analog RF inputs and outputs. A micro-SD card slot, an RJ45 socket for Ethernet, a USB port, and a micro-USB connector for the console are also included on the board. Radio frequency transmissions can both be received and transmitted by the Red Pitaya which operates in the frequency range of 50MHz.

### *B. Theory of Ultrasonic Object Differentiation*

Object differentiation with ultrasound involves using ultrasonic waves to distinguish between different objects or materials based on their physical properties, such as density, composition, and structure. This technique is commonly used in various fields, including medical imaging, non-destructive testing, and industrial inspection.

Ultrasound works on the principle of sending high-frequency sound waves into a material or object and analysing the echoes that bounce back. When an ultrasonic wave encounters a boundary between two different materials (e.g., air and tissue, metal, and plastic), part of the wave is reflected back while the rest continues to penetrate deeper into the material. By analysing the time, it takes for the echoes to return and their amplitude, it's possible to determine the properties of the materials and differentiate between them.

*Physical Phenomena and Their Impacts:*

*1. Attenuation:* Attenuation refers to the gradual loss of intensity of the ultrasound signal as it travels through a medium. This phenomenon occurs due to absorption, scattering, and reflection of the sound waves by the material. Attenuation can vary depending on factors such as the frequency of the ultrasound waves, the properties of the material, and the distance travelled. High levels of attenuation can reduce the reliability and precision of ultrasonic object differentiation, as it may result in weaker echoes and decreased signal-to-noise ratio [8].

*2. Reflection and Refraction:* When an ultrasound wave encounters a boundary between two materials with different acoustic impedances (the product of density and sound velocity), part of the wave is reflected back, and part is transmitted into the second material. The angle of reflection and refraction depends on the acoustic properties of the materials involved. Reflection and refraction can affect the accuracy of object differentiation, especially at interfaces between materials with significant differences in acoustic impedance [9].

*3. Scattering:* Scattering occurs when ultrasound waves interact with small irregularities or inhomogeneities within a material, causing the wavefront to deviate from its original path. Scattering can result in diffuse reflections and a loss of coherence in the received signal. In materials with high levels of scattering, such as biological tissues, object differentiation may be more challenging due to the complexity of the echo patterns.

*4. Multipath Propagation:* Multipath propagation refers to the phenomenon where ultrasound waves travel along multiple paths between the transmitter and receiver, leading to the reception of multiple echoes from the same object. This can complicate the interpretation of the received signal and make it more challenging to differentiate between objects, especially in environments with complex geometries or structures [10].

*5. Noise:* Noise sources, such as electronic interference, environmental factors, and system artifacts, can introduce additional signals into the received ultrasound data, leading to false detections or reduced signal quality. Minimizing noise and optimizing signal processing algorithms are essential for improving the reliability and precision of object differentiation with ultrasound.

In summary, while ultrasound offers valuable capabilities for object differentiation, various physical phenomena such as attenuation, reflection, scattering, multipath propagation, and noise can impact the reliability and precision of the technique. Understanding these phenomena and their effects is crucial for developing effective ultrasonic sensing systems and interpreting ultrasound data accurately in practical applications.

### *C. Theoretical Framework and Signal Analysis Methodology for Ultrasonic Sensor Systems*

This research explores the effectiveness of Frequency-Modulated Integrated Ultrasonic Sensors (FIUS) in distinguishing between inanimate objects and humans by examining the ultrasonic backscatter signals. It's predicated on the notion that the unique surface features of different entities alter the reflected ultrasonic signals in distinct ways, enabling their identification based on the characteristics of these signals. The FIUS sensor works by emitting ultrasonic waves, typically around 40 kHz, and analysing the frequency spectrum of the signals that bounce back from the target. The variation in the reflected signals, influenced by the material and texture of the target's surface, results in discernible changes in the spectrum of the received signals. When these ultrasonic waves are emitted, their interaction with the target's surface generates a backscattered signal that encodes the surface attributes [11].

*Surface Characterization Through Signal Analysis:*

*1. Signal Modulation by Surface Texture:* The study theorizes that smooth surfaces, such as car bumpers, reflect ultrasonic waves more uniformly, resulting in a backscattered signal with a smoother frequency spectrum. In contrast, more complex surfaces, like clothing on pedestrians, cause diffraction and scattering of the ultrasonic waves, leading to a frequency spectrum with multiple peaks and a less uniform distribution.

*2. Frequency Shift Analysis:* The analysis focuses on detecting shifts in the peak frequencies of the backscattered signal. A shift from the mean frequency of the emitted signal indicates interaction with non-uniform surfaces, providing a basis for distinguishing between different types of objects.

*3. Spectral Shape Examination:* Beyond frequency shifts, the overall shape of the signal's frequency spectrum provides additional insights into the surface's texture. The study posits that non-Gaussian spectral shapes are indicative of complex surfaces, such as those encountered in pedestrian clothing.

The research methodology involves a comparative analysis of the frequency spectra from the backscattered signals off known solid and soft targets. This comparison is intended to uncover specific spectral patterns and markers linked to various surface types. Statistical methods are employed to measure these differences and validate the spectral features' effectiveness as indicators of surface texture [11].

The anticipated results of this theoretical investigation are set to enhance the comprehension of how ultrasonic signals interact with different surfaces, aiding the development of more refined and precise pedestrian detection technologies. By exploiting the subtle variations in ultrasonic backscatter, this approach seeks to improve object recognition capabilities, potentially benefiting a wide range of applications [11].

### *D. Signal Processing and Feature Extraction*

This stage is critical in transforming raw ultrasonic data into meaningful features that allow for the accurate classification of different materials. This process is grounded in several well-established principles of signal analysis, each contributing to the reduction of noise, enhancement of signal fidelity, and extraction of key characteristics that distinguish one material from another. Below is a breakdown of the theoretical framework that underpins the signal processing and feature extraction methodologies.

*Windowing to Reduce Spectral Leakage:*

The first step in examining a signal's frequency composition is to minimize the effects of spectral leakage. Spectral leakage occurs when abrupt transitions or sharp edges in a signal cause the energy of some frequency components to "spill over" into adjacent frequencies, so warping the signal's true frequency representation.

To address this issue, a windowing function is applied to the time-domain signal. The goal of windowing is to smooth the signal at its edges, reducing the abrupt changes at the boundaries and thereby preventing excessive spectral leakage in the frequency domain. In this case, the Hann window is employed, which gradually tapers the signal at both ends, ensuring that the frequency spectrum is more concentrated around the true frequency components of the signal. This step enhances the accuracy of the subsequent frequency domain analysis.

Mathematically, the Hann window is expressed as:

$$w(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right), \quad 0 \leq n \leq N-1$$

Where N is the total number of samples in the signal and n represents the sample index. By using this window, the signal's energy is kept concentrated in its main frequency components, which makes it better suited for frequency-domain analysis.

*Fourier Transform for Frequency Analysis:*

The Fourier Transform is a powerful mathematical tool that allows for the decomposition of a signal into its constituent frequencies. More specifically, the Fast Fourier Transform (FFT) is utilized to efficiently calculate the discrete Fourier transform (DFT) of the signal.

The Fourier Transform is a useful tool for analysing signal frequency composition because it may identify the main frequencies resulting from the interaction of ultrasonic waves with the substance under study. Since every material has a different way of reflecting and absorbing sound waves, resulting in distinct frequency patterns, these frequency components are crucial for discriminating between different materials.

The Fourier Transform is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i\,2\pi\,k\,n\,/\,N}$$

In the frequency domain, the Power Spectral Density (PSD) of the signal is also calculated, which offers a measure of the signal's power as a function of frequency. This helps determine the amount of the signal's energy that is concentrated at specific frequency components.

*Feature Extraction:*

Feature extraction is the process of identifying and quantifying significant signal properties that can help distinguish between different materials. The frequency-domain representation of the ultrasonic signals yields some significant features, which are as follows:

*1. Signal-to-Noise and Distortion Ratio (SINAD):* SINAD is a measure of the overall quality of a signal by comparing the power of the signal of interest to the combined power of noise and distortion. In the context of ultrasonic signal analysis, SINAD allows us to quantify how much of the signal's energy corresponds to useful frequency components (e.g., those reflected by the material) versus how much is due to noise. A higher SINAD value indicates a cleaner, more distinguishable signal, making it easier to classify the material.

$$\text{SINAD} = 10 \log_{10} \left( \frac{P_{\text{signal}}}{P_{\text{noise}}} \right)$$

*2. Peak Detection:* The frequency-domain representation often exhibits peaks corresponding to the dominant frequencies reflected by the material. The number of these peaks (peak count) and their positions in the frequency spectrum are important features that can be used to differentiate between materials. Peaks represent frequencies where the signal exhibits high energy, and different materials tend to produce distinct peak patterns due to their unique acoustic properties.

*3. Autocorrelation for Periodicity Detection:* The autocorrelation of the signal is used to identify periodic structures within the signal. Autocorrelation measures how well the signal correlates with a time-shifted version of itself. Materials with periodic surface features (such as wood grain) may exhibit periodicity in the reflected ultrasonic signal. The maximum value of the autocorrelation function provides insight into the signal's periodicity, which can serve as a distinguishing feature.

$$R(\tau) = \sum_{t=0}^{N-\tau-1} x(t) \cdot x(t+\tau)$$

*E. ML Models*

Implementing a machine learning method for identifying the first reflection in an audio or acoustic environment involves several key steps. Below is a generalized approach that you can follow:

1. *Define the Problem*: Clearly define the problem and the objectives of identifying the first reflection. Understand the characteristics of first reflections in the context of your application [12] [13].

2. *Data Collection*: Gather a dataset that includes audio recordings with labelled information about the presence and characteristics of first reflections. Ensure diversity in the dataset to capture various acoustic conditions [12] [13].

3. *Data Preprocessing*:

- Audio Segmentation: Divide audio recordings into segments relevant to the analysis.
- Feature Extraction: Extract relevant features from the audio segments (e.g., spectral features, time-domain features, MFCCs).
- Labelling: Ensure accurate labelling of the dataset, marking segments with and without first reflections [12] [13].

4. *Data Splitting*: Split the dataset into training, validation, and test sets to evaluate the model's performance [12] [13].

5. *Model Selection*: Choose an appropriate machine learning model based on the nature of the problem. Experiment with different algorithms such as SVM, decision trees, CNNs, or hybrid models depending on your dataset and problem requirements.

6. *Model Architecture*: Design the architecture of the chosen model, considering input features, hidden layers, and output layer. For deep learning models, experiment with architectures like CNNs or hybrid CNN-LSTM models [13].

7. *Feature Scaling and Normalization*: Standardize or normalize the input features to ensure that the model trains effectively and converges faster.

8. *Model Training*: Train the model using the training dataset. Fine-tune hyperparameters through iterative training and validation [12].

9. *Model Evaluation*: Evaluate the model's performance on the validation set to avoid overfitting. Adjust the model architecture or hyperparameters as needed [13].

10. *Testing*: Assess the final model on the test set to gauge its generalization performance.

11. *Post-Processing (if needed):* Implement any post-processing steps, such as filtering or thresholding, to refine the model's output.

12. *Interpretability and Visualization:* Analyse and visualize the model's predictions to gain insights into its decision-making process. Understand which features are most informative for detecting first reflections [13].

13. *Deployment*: If the model meets the desired performance, deploy it in the target environment. Consider real-time or batch processing depending on the application requirements [12].

14. *Continuous Monitoring and Improvement*: Implement a monitoring system to track the model's performance over time. Consider retraining the model periodically with new data to adapt to changing conditions. [12]

15. *Documentation*: Document the entire pipeline, including data preprocessing steps, model architecture, hyperparameters, and any insights gained during the development process.

As part of the project, it was experimented with six models:

*1. K-Nearest Neighbours (KNN)*

K-Nearest Neighbours (KNN) is a non-parametric classification technique that assigns a sample to the class that is most common among its k closest neighbours in the feature space. The simplicity of KNN makes it easy to comprehend and implement, as it does not rely on any assumptions about the underlying data distributions. It works by calculating the distances between samples and then using the k-nearest samples to determine the class of the input. KNN is particularly well-suited for problems where the decision boundaries between classes are irregular, making it a good choice for complex feature spaces [14]. KNN can be effective when the feature space is well-structured and the decision boundaries between classes are relatively smooth [15] [16]. However, its effectiveness depends heavily on the selection of k (the number of neighbours) and the distance metric, such as Euclidean distance.

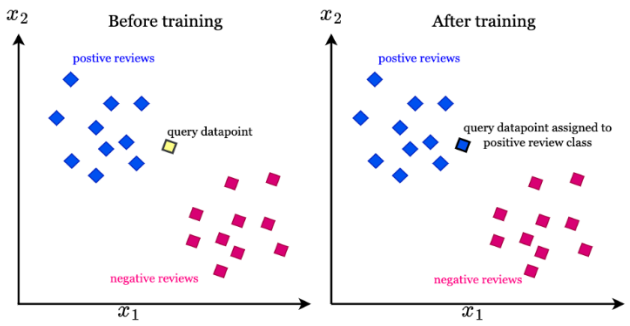


Fig.2. K-Nearest Neighbours classification graph

*2. Support Vector Machine (SVM)*

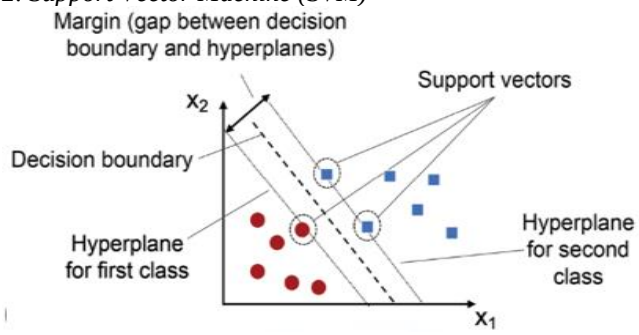


Fig.3. SVM graph

Support Vector Machine (SVM) is a supervised learning algorithm that is highly adept at tackling both binary and multi-class classification problems, especially when dealing with high-dimensional data. SVM operates by identifying the optimal hyperplane that can effectively separate the different classes in a feature space, while simultaneously maximizing the distance between these classes [17]. For cases where data is not linearly separable SVM uses the kernel technique to transform the data into a higher-dimensional space, making it easier to separate the classes [18]. The Radial Basis Function (RBF) kernel is especially adept at capturing non-linear patterns in the signal characteristics. The model's hyperparameters, including the regularization parameter (C) and kernel coefficient (gamma), are adjusted to optimize the classification performance. For signal classification problems, SVMs can be highly effective when working with well-designed features, particularly in situations where the number of features surpasses the number of samples [15] [16].

*3. Random Forest*

Random Forest (RF) is an ensemble learning method that involves constructing multiple decision trees during training and uses the majority vote from these trees for classification. The method then employs a technique called bagging, where each tree is trained on a random subset of the data. This helps to reduce the risk of overfitting and enhances the model's ability to generalize well to new, unseen data [19]. Random Forest is particularly well-suited for datasets where interactions between features are complex, and it also provides valuable insights into feature importance [20]. The model's interpretability is a key advantage, as it helps identify which features contribute most to the classification task.

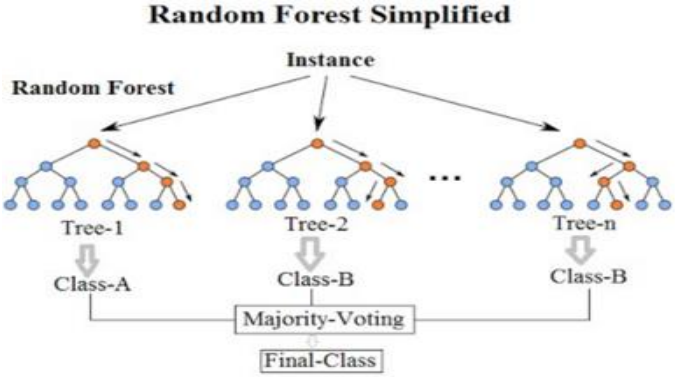


Fig.4. Random Forest

*4. Multilayer Perceptron (MLP)*

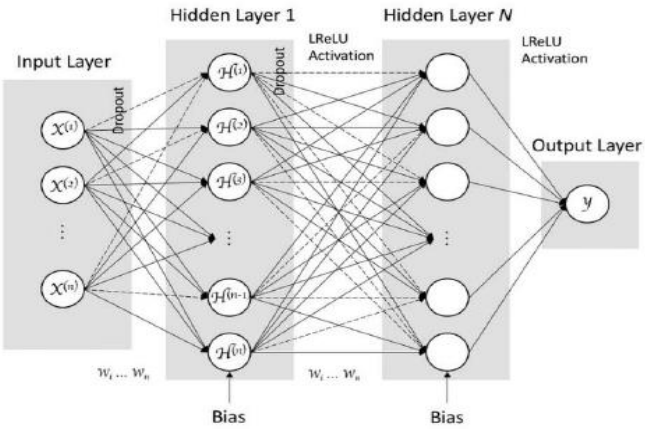


Fig.5. Multilayer Perceptron (MLP) Architecture

Multilayer Perceptron (MLP) is a type of artificial neural network that has a feedforward structure, meaning the information flows in one direction from the input to the output. Unlike traditional algorithms, MLP is capable of learning complex, non-linear relationships between the input features and output labels [21]. The network consists of multiple layers of interconnected neurons - an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is fully connected to the neurons in the subsequent layer, with the connections representing weighted inputs. MLP utilizes a backpropagation technique for training the network, which makes it suitable for complex pattern recognition tasks. This approach involves adjusting the weights of the connections between neurons to minimize the error between the predicted and actual outputs. Notably, MLPs are generally most effective when provided with well-processed input features, such as those extracted from the ultrasonic signals in this study. In contrast to other neural network architectures, every node in each layer of an MLP is connected to every node in the next layer, forming a dense network of connections. This structure allows the MLP to learn and represent complex, non-linear relationships between the input and output data.

*5. Convolutional Neural Networks (CNN)*

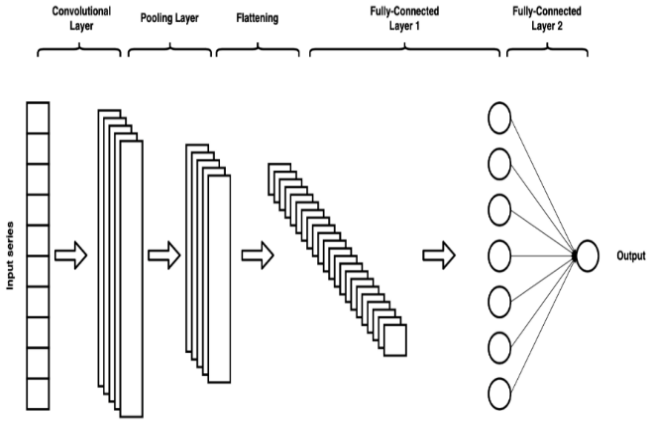


Fig.6. Simplified schema of 1D CNN model

Convolutional Neural Networks (CNN) are deep learning models that are particularly effective for processing structured grid-like data, such as images, spectrograms, and time-series signals [22]. CNN utilize convolutional layers to automatically learn spatial hierarchies within the data by applying filters that detect local patterns. In the context of ultrasonic signal classification, CNNs can identify relevant frequency patterns, such as the dominant peaks that arise from material-specific reflections of ultrasonic waves. Pooling layers are then used to reduce the dimensionality of the data, making the model computationally efficient while preserving important features. Lastly, fully connected layers are employed to combine the learned features and perform the classification task. CNNs are particularly well-suited for this study due to their ability to automatically extract and learn hierarchical features from the FFT-transformed ultrasonic signal data, rendering them highly effective for material classification. [23].

*6. Long Short-Term Memory (LSTM)*

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) that are engineered to capture long-term dependencies within sequential data. LSTMs achieve this by maintaining a memory of past inputs over time, utilizing a system of gates that selectively retain relevant information and discard irrelevant information [24]. The temporal structure of the data is a crucial factor in certain tasks, such as recognizing reflections in ultrasonic signals that occur at different time points. LSTMs are particularly well-suited for such tasks, as they excel at handling time-series data and capturing relationships over time. In this study, LSTMs are employed to classify materials based on sequences of extracted features from the ultrasonic signals. The inherent capability of LSTMs to work with time-series data and capture temporal dependencies makes them highly effective for this type of signal classification task. [25].

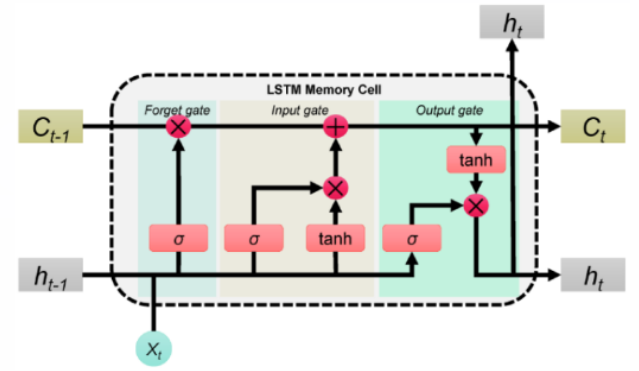


Fig.7. Long Short-Term Memory (LSTM) Architecture

*F. Hyperparameter Tuning for Optimization*

Hyperparameter tuning is a crucial step in enhancing the performance and generalization capabilities of machine learning models. Hyperparameters are specific parameters that control the model's learning process but are not directly learned from the data itself. These hyperparameters have a significant impact on the model's ability to capture patterns and generalize effectively to new, unseen data. In this project, various hyperparameter tuning techniques were employed.

*Grid Search Cross-Validation (CV):* Grid Search is a comprehensive search method that determines the best configuration by analysing every conceivable combination of a given set of hyperparameters. Grid Search is computationally expensive since it must test every possible combination of hyperparameters, which can take a long time for bigger models or parameter spaces, even though it can provide optimal hyperparameter configurations.

*Randomized Search Cross-Validation (CV):* Randomized Search effective option to Grid Search is Randomized Search, which selects a predetermined number of hyperparameter combinations at random from a predetermined distribution. This approach lowered computing time considerably while allowing for extensive hyperparameter exploration. Faster exploration of the hyperparameter space was made possible by Randomized Search with only a minor performance trade-off.

*Bayesian Optimization:* Bayesian Optimization takes a more intelligent approach which models the relationship between hyperparameters and performance using historical evaluations. In an effort to identify the ideal parameters more quickly, it chooses the subsequent hyperparameter set based on prior assessments rather than picking combinations at random. The optimizer focuses its search on regions of the hyperparameter space that are most likely to produce better results by using past knowledge about which hyperparameters have worked successfully. This procedure is an effective technique for huge models like Random Forest since it drastically cuts down on the amount of time needed to converge to optimal settings.

*Hyperband Search (HalvingGridSearchCV):* Hyperband is a resource-efficient approach that computes more hyperparameter configurations by giving less resources (such training time or dataset size) to underperforming configurations and increasing resources to higher-performing configurations as the search goes on. Hyperband is a helpful strategy for models like Random Forest where the search space might be enormous since it allows for the evaluation of a large number of configurations.

*SMOTE (Synthetic Minority Over-sampling Technique):* When a class has much fewer examples than the other classes, it is said to be imbalanced. When the model makes predictions, this could cause it to be skewed in favour of the dominant class. By creating artificial examples for the minority class, SMOTE is a technique that solves this problem. Interpolating between instances of the minority class that already exist allows it to accomplish this. Through this method of balancing the class distribution, SMOTE makes sure the model has enough samples from each class to adequately learn the underlying patterns in the data.

## III. IMPLEMENTATION

### A. Measurement Environment and Setup

*Laboratory Configuration:* The experiments were carried out in the controlled setting of the Machine Learning laboratory at the Frankfurt University of Applied Sciences. The significance of a controlled environment is to minimize variables that could affect sensor accuracy.

*Sensor and Object Placement:* A FIUS sensor was strategically placed atop an elongated metal stand to facilitate a clear line of sight to the object of interest. Beneath it, the object—central to the experiment—was stationed on a white stand. This arrangement ensured consistent positioning for measurement repeatability.

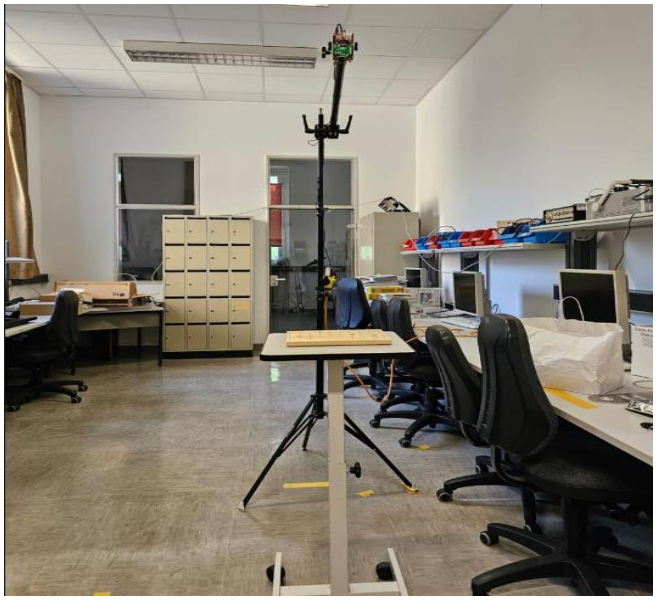

Fig.8. Measurement Setup

*Object Utilization:* The objects used in this project include aluminium foil, plain wood, wood with a significant number of nails placed both closely together and spaced farther apart, and steel.

Fig.9. Aluminium foil and Fig.10. Steel

Fig.11. Plain wood, Fig.12. wood with a significant number of nails placed both closely together and Fig.13. spaced farther apart

### B. *Measurement Software Utilization*

*Software Interface and Functionality:* The data from the Red Pitaya measurement board is collected using a Measurement software developed in Frankfurt University of Applied Sciences. The custom software interface, as detailed in Figure 14, was a critical tool in the data collection process. Its graphical user interface (GUI) facilitated real-time analysis and visualization of the sensor data. Users could choose between analysing raw analog data or applying Fast Fourier Transform (FFT) for frequency domain analysis.

*Data Acquisition and Analysis:* The software allowed for the export of FFT-transformed data in a text format, which could then be processed and analysed. Its capability to plot FFT graphs and time-series data provided a comprehensive understanding of the sensor's performance under various conditions.

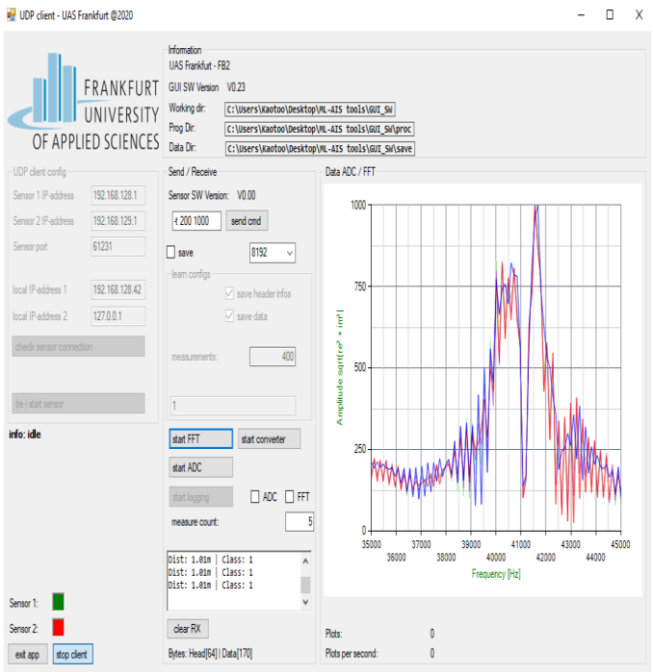


Fig.14. FFT data from the measurement software

### C. *Data Collection*

For this measurement, we used text-based ADC data that was exported from the software. Each data has 16384 columns overall, with rows representing amplitude values in the range of 0 to 1000 and columns representing frequency values.

Data headers are also exported by the program in addition to ADC data. The length of the data, the classification outcome from the software's current model, or the sampling frequency are just a few examples of the useful information included in data headers. Figure 15 shows a sample data format, and the table below, Table I, provides information on the data headers. The extensive collection of data was crucial to train our model effectively, allowing it to recognize and adapt to different scenarios.

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 64 | 32768 | 1 | 1.1 | 512 | 0 | 1953125 | 12 | 0 | 0.1 | 0.98 | 0.2 | 0.3 V0.2 | 0.4 | 0.5 | -3 | -4 | -5 | -1 | -3.1 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -3 | -2 | 1 | -187 | -155 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -9 | -5 | -3 | -5 | -232 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -11 | -7 | -6 | -228 | -248 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -2 | -4 | -6 | -4 | -216 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -5 | -4 | -6 | -7 | 268 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -69 | -76 | -71 | -71 | -166 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -9 | -8 | -9 | -14 | -7 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | 0 | -1 | -1 | -9 | -220 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | 6 | 3 | 1 | 185 | 195 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -4 | -1 | -249 | -228 | -200 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -12 | -14 | -12 | -262 | -256 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | 0 | 1 | 3 | -113 | -80 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | 0 | 3 | 6 | -161 | -130 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -2 | -4 | -1 | -22 | 23 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | 2 | -2 | -2 | -1 | -21 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -4 | -1 | 1 | 170 | 186 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -1 | -2 | 1 | 173 | 184 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | 5 | -1 | -2 | -5 | 101 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -2 | -2 | -3 | 64 | 99 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -7 | -8 | -9 | -9 | 183 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -5 | -7 | -8 | -9 | 167 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -94 | -91 | -88 | -167 | -192 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | 1 | 2 | 1 | 0 | 1 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -3 | -6 | -2 | -224 | -238 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | 11 | 8 | 5 | 8 | 9 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | 0 | 2 | 2 | 2 | 131 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.98 | 0 | 0 V0.2 | 0 | 0 | -1 | -4 | -2 | 36 | 74 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -10 | -10 | -12 | -10 | -255 |
| 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 0 | 0.99 | 0 | 0 V0.2 | 0 | 0 | -3 | -7 | -4 | -7 | -6 |

Fig.15. Raw measurement data

TABLE.I DESCRIPTION OF THE HEADERS IN THE MEASUREMENT FILE

| Column | Header Description | Value | Remarks |
|---|---|---|---|
| 1 | Header lengths | 64 | |
| 2 | Data length | 170 | |
| 3 | Class detected | 1 or 2 | 1: Object<br>2: Human |
| 4 | Measurement type | 0 or 1 | 0 : FFT<br>1 : ADC |
| 5 | Frequency resolution f or sampling time t depend on measurement type | 119 | |
| 6 | - | 0 | irrelevant |
| 7 | Sampling frequency (Hz) | 1953125 | |
| 8 | ADC resolution | 12 | |
| 9 | | 0.0 | irrelevant |
| 10 | Distance between sensor and first object (round-trip-time in µs) | - | |
| 11 | FFT Window length | 0 | |
| 12 | | 0 | irrelevant |
| 13 | software version (RP) | V0.2 | |

### D. *Data Preprocessing*

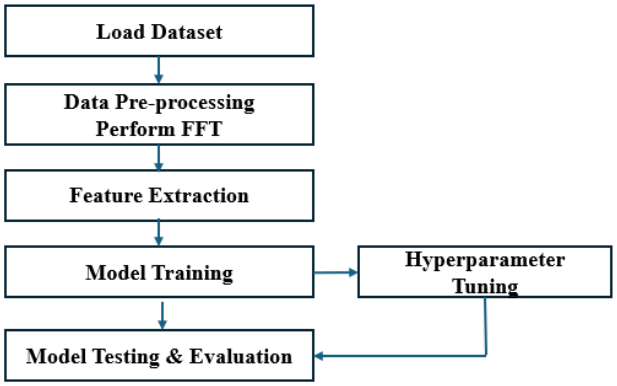


Fig.16. Flowchart of the Model

The implementation details a structured methodology for processing and analysing ultrasonic signal data to facilitate the classification of various materials. Each step in the data preprocessing and feature extraction phases serves a crucial purpose in preparing high-quality data for training machine learning models. The primary objective is to extract meaningful features from the raw ultrasonic signals, which will enhance the accuracy of the classification process, enabling the models to effectively differentiate between distinct materials based on their unique signal properties.

*Data Pre-Processing and Feature Extraction Steps:*

*Data Preprocessing:* The initial phase involves reading and preparing the data for subsequent analysis. The process begins with importing signal data from a CSV file each corresponding to different materials (e.g., aluminium, various types of wood, and steel). Given that not all columns in the file may be relevant to the analysis, a selection is made to focus on specific columns containing the essential signal data. This refined dataset is then subjected to further preprocessing to ensure it is in a suitable format for analysis.

*Signal Windowing:* Signal windowing is a pivotal preprocessing step designed to mitigate spectral leakage, a common issue in signal processing where energy from the signal 'leaks' into adjacent frequencies. To address this, a windowing function, specifically a Hanning window, is applied to each signal. The Hanning window, characterized by its smooth, sinusoidal shape, effectively tapers the signal at its beginning and end, thereby reducing spectral leakage. This process not only enhances the quality of the frequency analysis but also improves the model's ability to differentiate the objects.

*Noise Reduction and Peak Detection:* The next step involves converting each signal from the time domain to the frequency domain using the Fast Fourier Transform (FFT). This transformation allows for the analysis of the signal's frequency components, which enables the identification and removal of noise through a Power Spectral Density (PSD) thresholding method. Noise components with a PSD below the threshold are filtered out, and the signal is then reconstructed in the time domain using an inverse FFT.

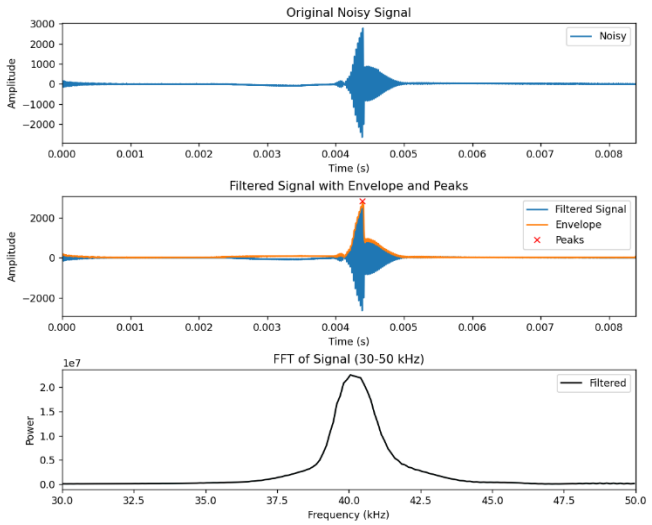


Fig.17. ADC to FFT Plot

Once the noise is reduced, peak detection is performed to identify prominent frequency components in the signal that may correspond to reflections from the material's surface. The function calculates various features, including SINAD (Signal-to-Noise and Distortion Ratio), the number of detected peaks, and the position of the first significant peak. These features are crucial for distinguishing between different materials, as each material may have a unique frequency signature. Peak detection is essential for identifying key features that can be used by machine learning models to classify materials based on their ultrasonic response. Every signal's autocorrelation is computed in addition to its frequency-based properties to find any periodic patterns in the data. The autocorrelation function is a valuable tool for determining recurring patterns in a signal, such as periodic reflections or echoes that are exclusive to a given material. It quantifies how similar a signal is to itself over time. The autocorrelation is calculated using the correlate() function, and the greatest value is noted as an extra feature. Periodicity can indicate material properties, such as surface roughness or internal structure, making it another valuable feature for classification.

*Feature Combination and Data Labelling:* After extracting features such as SINAD, peak count, peak position, and autocorrelation, all these features are combined into a Data Frame. Each row in the Data Frame corresponds to a single ultrasonic signal, with columns representing the extracted features and a label indicating the material associated with that signal. This final structured dataset is now ready to be utilized for training machine learning models, where each signal is described by a set of features that capture its most significant properties.

Extracted Features:

| | sinad | peak_count | peak_position | autocorr_max | label |
|---|---|---|---|---|---|
| 0 | 16.888277 | 2 | 85 | 8.770884e+08 | aluminum |
| 1 | 15.771295 | 3 | 5 | 8.747460e+08 | aluminum |
| 2 | 16.307615 | 2 | 85 | 9.741237e+08 | aluminum |
| 3 | 16.905421 | 4 | 1 | 8.861499e+08 | aluminum |
| 4 | 16.266501 | 2 | 85 | 9.645380e+08 | aluminum |

Fig.18. Features extracted

### *E. Training the Model*

The model architecture selected for a data-driven project in the machine learning sector is crucial to its success. This section explores six sophisticated machine learning models: Random Forest, Convolutional Neural Networks (CNN), Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Multilayer Perceptron (MLP), and Long Short-Term Memory (LSTM). Each of these models is well-known for its unique capabilities in managing intricate datasets with high-dimensional features.

#### *1.* K-Nearest Neighbours (KNN)

The *initial implementation* of the KNN model was basic and used the default values provided by the KNeighboursClassifier from the scikit-learn module. The model in question categorized data points according to the five nearest neighbours by using the default value of n_Neighbours=5. In this version, there was no hyperparameter adjustment done.

- *Dataset Preparation:* The dataset was divided into training and testing sets following the data preprocessing step. The KNN model was fitted using the training data, and its performance was assessed using the test set.
- *Feature Scaling:* Due to its distance-based algorithmic design, KNN is sensitive to the input feature scale. To ensure that each feature contributed equally to the distance calculation, the features were normalized using StandardScaler.
- *Model Training:* The scaled training dataset was used to train the KNN classifier. Based on the majority class of its five closest neighbours in the feature space, the simple KNN model classified each test data point using its default settings.
- *Model Evaluation:* The trained model was evaluated on the test set using several metrics, including accuracy, precision, recall, F1-score, and a confusion matrix. This provided a baseline performance for the KNN model, allowing for comparison with the hyperparameter-tuned version.

Although the simple KNN model produced passable classification results, its performance might be enhanced by adjusting the hyperparameters (number of neighbours, distance metric, and weighting scheme) that govern the model's behaviour. Using RandomizedSearchCV, hyperparameter adjustment was used to enhance the KNN model's performance. Finding the ideal set of hyperparameters to increase the model's classification accuracy was the main goal.

*Hyperparameter Tuning*

a) Hyperparameter Search Space

The following hyperparameters were optimized:

- *n_Neighbours*: The number of Neighbours to consider when making a prediction is an important factor. Having a larger number of Neighbours can help mitigate the impact of noisy data, but it may also diminish the model's capacity to capture localized patterns.
- *weights*: The approach of assigning weights to Neighbours. Two alternatives were evaluated: equal weighting (all Neighbours have the same contribution) and distance-based weighting (Neighbours closer to the target have a greater influence).
- *metric*: The distance metric used to compute the distances between data points. Common options include Euclidean, Manhattan, Chebyshev, and Minkowski, each representing different methods of calculating the distance between points.

b) RandomizedSearchCV

- A randomized search was carried out over the hyperparameter space using RandomizedSearchCV. This method selects a random subset of hyperparameter combinations and evaluates their performance. This approach is more computationally efficient than a comprehensive grid search, especially when the hyperparameter space is vast.
- The search was run for 50 iterations, and for each iteration, the model underwent 5-fold cross-validation to ensure that the selected hyperparameters would generalize well to new, unseen data.

*Best Hyperparameter Selection*

After the randomized search process was completed, the best set of hyperparameters was chosen based on the model's performance during cross-validation. The optimized hyperparameters included the ideal number of Neighbours, the most suitable distance metric, and the most appropriate weighting method.

*Model Training and Evaluation*

The KNN model was re-trained using the best hyperparameters on the training data. After training, the model was evaluated on the test set using the same evaluation metrics as the basic KNN model. The confusion matrix and classification report provided insights into the performance improvements achieved through hyperparameter tuning.

*Model Saving*

In order to guarantee that the best-performing model was kept for deployment or more research, the optimized model was stored for later use.

#### *2. Support Vector Machine (SVM)*

The *initial implementation* included utilizing a conventional SVM model without any hyperparameter adjustments. The SVM model was employed with its default parameters, namely the Radial Basis Function (RBF) kernel, which is frequently useful for non-linear classification applications.

- *Dataset Preparation:* The dataset was divided into training and testing sets following the data preprocessing step. The SVM model was fitted using the training data, and its performance was assessed using the test set.
- *Feature Scaling:* The use of StandardScaler was made for feature scaling because SVM is sensitive to the scale

of input features. By ensuring feature standardization, this phase enables SVM to perform at its best when determining the distances between data points.

- *Model Training:* The training dataset was used to train the SVM model. Because of its adaptability in capturing non-linear patterns—a crucial feature in signal classification tasks—the RBF kernel was selected.
- *Model Evaluation:* The model was evaluated on the unseen test dataset following training. The model's performance was assessed using a number of metrics, including the confusion matrix, accuracy, precision, recall, and F1-score. The SVM model's baseline performance was given by these metrics.
- *Overfitting and Underfitting Check:* The comparison of training and testing accuracies made sure the model wasn't overfitting.

In the second stage of the SVM implementation, hyperparameter adjustment was undertaken to optimize the model's performance. Two techniques were used: RandomizedSearchCV and GridSearchCV. Finding the ideal set of parameters to increase the model's precision and capacity for generalization is the aim of hyperparameter tuning.

*Hyperparameter Tuning*

a) Hyperparameter Search Space

The key hyperparameters that were optimized include:

- *C:* The regularization parameter that determines the trade-off between maximizing the margin and minimizing classification error. While a lesser C value permits a bigger margin but may accept more classification errors, a higher C value yields a smaller margin but fewer classification errors
- *kernel:* The kernel function that creates a higher-dimensional space map from the given features. The most popular choices are poly, RBF, and linear. While the linear kernel is faster but less versatile, the RBF kernel is appropriate for non-linear classification.
- *gamma:* The kernel coefficient for the RBF kernel It indicates the extent to which a single training example has an impact. A greater gamma indicates a more confined influence, whereas a lesser gamma indicates a more widespread influence.
- *degree:* Indicates the degree of the polynomial that was used to fit the data and is only relevant for the polynomial kernel.

b) GridSearchCV

- In the first approach, a grid of hyperparameter values was searched thoroughly using GridSearchCV. Every feasible combination of the given hyperparameters was assessed using this procedure. Five-fold cross-validation was employed by GridSearchCV to make sure the model performed well when trained on untested data.
- Following a GridSearchCV run, the optimal C, kernel, and gamma combination was chosen. These ideal hyperparameters were then used to retrain the model, and the test dataset was used to assess how well it performed.

c) RandomizedSearchCV

- RandomizedSearchCV was used as an alternative to GridSearchCV. Rather than evaluating every conceivable combination of hyperparameters, this technique sampled a predetermined number of combinations from the given distributions. This method allowed the model to investigate a larger range of hyperparameter values and was computationally faster.

Model Evaluation and Performance Comparison

On the test set, the optimized SVM models from GridSearchCV and RandomizedSearchCV were assessed. The model's performance was evaluated using important metrics such the confusion matrix, F1-score, accuracy, precision, and recall. To further assess for overfitting or underfitting, the training and testing accuracies of the optimized model were compared. After hyperparameter optimization, we were able to see a considerable improvement in classification accuracy when comparing the performance of the basic and optimized SVM models.

3. Random Forest

The *initial implementation* of the Random Forest model involves creating a baseline performance using the classifier's default settings. During training, Random Forest builds several decision trees and produces a class that is the mode of the classes (classification) from each individual tree. It is a potent ensemble learning technique.

- Dataset Preparation: The dataset was divided into testing and training sets so that the model could be tested on untested data and assessed on part of the data to gauge its generalization capacity.
- Feature Scaling: Although tree-based models like Random Forest are not sensitive to the scaling of features, feature scaling using StandardScaler was undertaken to standardize the features across models, assuring consistency between different algorithms.
- Model Training: Using the Gini Impurity as the splitting criterion and 100 trees (n_estimators=100) as default settings, a simple Random Forest classifier was employed. The training dataset was used to train the model.
- Model Evaluation: The model's performance was assessed on the test set following training using a number of metrics, including accuracy, precision, recall, and F1-score. Furthermore, the confusion matrix was displayed to show how accurately the model identified the various material groups.
- Overfitting and Underfitting Check: To look for any indications of overfitting or underfitting, a comparison of the test and training accuracies was performed. A tiny difference in these accuracies suggested that the model was well-fit.

The second phase of the implementation focused on optimizing the hyperparameters of the Random Forest model to improve its performance. To determine the best parameters for the model, a number of hyperparameter tuning techniques were used, each with its own unique methodology.

*Hyperparameter Tuning*

a) Hyperparameter Search Space

The key hyperparameters optimized include:

- *n_estimators:* The number of trees in the forest. More trees generally lead to better performance but increase computation time.
- *max_depth:* The maximum depth of the trees. Deeper trees can model more complex relationships but may overfit.
- *min_samples_split:* The minimum number of samples required to split a node. Higher values can prevent overfitting.
- *min_samples_leaf:* The minimum number of samples required to be at a leaf node. Larger values encourage simpler models.
- *criterion:* The function to measure the quality of a split. gini for Gini Impurity and entropy for information gain.
- *max_features:* The number of features to consider when looking for the best split. This hyperparameter affects the randomness of the trees.

b) Grid Search

In the first approach, GridSearchCV was used to perform an exhaustive search over a manually specified parameter grid. GridSearchCV evaluates all combinations of hyperparameters using 5-fold cross-validation and returns the best set of hyperparameters. The hyperparameter grid included values for n_estimators, max_depth, min_samples_split, min_samples_leaf, bootstrap, and criterion.

c) Randomized Search

RandomizedSearchCV is an alternative to GridSearchCV where random combinations of hyperparameters are sampled and evaluated. This method is faster than GridSearchCV because it does not evaluate every possible combination. A wide range of hyperparameters were sampled, and a fixed number of iterations (n_iter=50) was specified for the search. This approach allowed the model to explore a broader range of hyperparameter values in less time.

d) Bayesian Optimization

Bayesian Optimization using BayesSearchCV is a more advanced method of hyperparameter tweaking. Rather than experimenting with different combinations, Bayesian Optimization models the link between performance and hyperparameters using previous evaluations. Then, it selects the next set of hyperparameters based on this model to maximize performance. Compared to GridSearch and RandomizedSearch, this approach drastically shortens the time needed to converge on an ideal set of hyperparameters, making it an effective tool for intricate models like Random Forest.

e) Hyperband (HalvingGridSearchCV)

Hyperband is a hyperparameter tuning method that is resource-efficient and is implemented using HalvingGridSearchCV. By allocating fewer resources (i.e., training time or data) to poorly performing configurations and focusing more resources on better configurations, it assesses more configurations with less processing. By assessing a large number of configurations with fewer resources and gradually focusing on the best-performing configurations by increasing the resource allocation as the search goes, the HalvingGridSearchCV lowers the computational cost.

Model Evaluation and Performance Comparison

Using the test dataset, every optimized model was assessed. To assess the effect of hyperparameter tuning, the outcomes were contrasted with the Random Forest model used as a baseline. The evaluation was conducted using accuracy, precision, recall, and F1-score as the primary measures. Furthermore, cross-validation was carried out to assess how effectively the model extrapolates to previously unobserved data.

4. *Multilayer Perceptron (MLP)*

The *initial implementation* of the MLP model was carried out using the default parameters of the MLPClassifier from scikit-learn to serve as a baseline for performance comparison.

- *Dataset Preparation:* An 80-20 split was used to separate the pre-processed dataset into training and testing sets. This made sure that the model was tested on untested data to gauge its generalization capacity after being trained on a subset of the data.
- *Feature Scaling:* For best performance, feature scaling is necessary for neural networks, including MLP. In order to verify that every feature had a mean of zero and a standard deviation of one, the features were standardized using StandardScaler, which speeds up the model's convergence.
- *Model Architecture:* The MLP Classifier's default setting of one hidden layer with 100 neurons was used to generate the basic MLP model. This hidden layer made use of the Rectified Linear Unit (ReLU) activation function, which works well with non-linear input.
- *Model Training:* The MLP model was trained using backpropagation with Stochastic Gradient Descent (SGD). Up to 1000 iterations, or epochs, were used to train the model, or until convergence.
- *Model Evaluation:* After training, the test set was used to assess the model. The F1-score, recall, accuracy, and precision were the main evaluation measures. To further visualize the model's performance across several material categories, a confusion matrix was created.
- *Overfitting and Underfitting:* The comparison of training and testing accuracies made sure the model wasn't overfitting.

The next phase of the implementation focused on optimizing the MLP model's hyperparameters to improve its performance. To discover the best combination of hyperparameters, RandomizedSearchCV was used to tune the hyperparameters. It searches across a predetermined parameter grid.

*Hyperparameter Tuning*

a) Hyperparameter Search Space

The key hyperparameters that were tuned include:

- *hidden_layer_sizes:* The number of neurons in each hidden layer. This parameter affects the model's capacity to learn complex patterns. Various configurations were tested, including single-layer and multi-layer architectures (e.g., (50,), (100,), (50, 50), and (100, 100)).
- *activation:* The activation function for the hidden layers. Several activation functions were explored, including logistic, ReLU, and tanh.
- *solver:* The solver for weight optimization. Both adam and sgd (Stochastic Gradient Descent) were tested.
- *alpha:* The regularization term (L2 penalty) to prevent overfitting. Values of 0.0001, 0.001, and 0.01 were considered.
- *learning_rate:* The learning rate schedule, which controls how the learning rate changes during training. Both constant and adaptive schedules were evaluated.

b) Randomized Search

To effectively search over a selection of potential hyperparameter combinations, RandomizedSearchCV was employed. Because it samples a finite amount of random hyperparameter combinations instead than assessing every conceivable combination, this method is faster than GridSearchCV. Each set of hyperparameter combinations was tested and trained on the MLP model using 5-fold cross-validation. The random hyperparameter combinations were assessed via a total of 20 iterations. Once the best hyperparameters were found, the MLP model was retrained using the entire training set with the optimized settings, and its performance was evaluated on the test set.

Model Evaluation and Performance Comparison

The optimized MLP model was assessed using the test dataset following hyperparameter adjustment. The performance enhancement was observed by comparing the obtained results with the baseline MLP model. The confusion matrix, F1-score, accuracy, precision, and recall were the evaluation measures that were employed.

5. Convolutional Neural Networks (CNN)

In the *initial implementation*, a simple CNN architecture was created and trained on the dataset without any hyperparameter adjustments.

- *Data Preparation:* The pre-processed dataset was loaded and split into training and testing sets with an 80-20 split. Each sample was reshaped into a 3D format suitable for 1D convolutional layers. Specifically, the features were reshaped to have a format of (samples, timesteps, features), where timesteps refer to the length of the feature vector and features are set to 1 for this task.
- *Feature Scaling:* The features were scaled using StandardScaler to ensure that each feature has a mean of zero and a standard deviation of one, which helps the CNN converge more efficiently.
- *Model Architecture:* The CNN architecture consisted of the following layers:
  - Input Layer: Accepts data in a 3D format suitable for 1D convolution.
  - Conv1D Layer: A convolutional layer with 64 filters and a kernel size of 3, followed by the ReLU activation function.
  - MaxPooling1D Layer: Reduces the dimensionality by selecting the maximum value in each pooling window.
  - Flatten Layer: Flattens the 3D data into a 1D vector for input into the dense layers.
  - Dense Layer: A fully connected layer with 100 neurons and ReLU activation to handle complex patterns in the data.
  - Output Layer: A dense layer with softmax activation, producing probabilities for each class.
- *Model Training:* The model was compiled using the Adam optimizer and trained using the sparse categorical crossentropy loss function due to the multi-class nature of the problem. The model was trained for 10 epochs with a batch size of 32.
- *Evaluation:* The test set was used to assess the CNN following training. For a more in-depth understanding of the model's performance, evaluation metrics included accuracy, precision, recall, F1-score, and a confusion matrix.
- *Overfitting and Underfitting Check:* The comparison of training and testing accuracies made sure the model wasn't overfitting.

Several architectural and training-related hyperparameters could be optimized with Keras Tuner, which was used to tune hyperparameters in order to enhance the CNN model's performance.

*Hyperparameter Tuning*

a) Hyperparameter Search Space

The key hyperparameters that were optimized include:

- *filters:* Number of filters in the convolutional layer. Different values (e.g., 32, 64, 128) were tested to find the optimal number of filters for feature extraction.
- *kernel_size:* Size of the convolutional filters. Values from 2 to 5 were tested to capture patterns of different lengths in the input data.
- activation: Activation function for the hidden layers. ReLU and tanh were evaluated.
- *pool_size:* Pooling window size for the MaxPooling layer. Values of 2 and 3 were considered.

- *dense_units:* Number of neurons in the dense (fully connected) layer. Various configurations (e.g., 64, 128, 256) were evaluated.
- *learning_rate:* Learning rate for the Adam optimizer. Rates of 0.01, 0.001, and 0.0001 were tested to determine the optimal learning speed.

b) Hyperparameter Tuning with Keras Tuner
Keras Tuner was used to explore the hyperparameter space efficiently by testing different combinations of the above parameters.

- RandomSearch was employed to evaluate 10 different combinations of hyperparameters. The tuner selected the best combination based on the validation accuracy during training.

- A batch size of 32 and 10 epochs were used for the tuning procedure. The top model was chosen based on how well it performed on the validation set following the hyperparameter search.

Optimized Model Training
The CNN was retrained using the complete training set with the ideal hyperparameter configuration once the best hyperparameters were determined. 20 epochs of training were added to the course to facilitate more learning.

Model Evaluation and Performance Comparison
The optimized MLP model was assessed using the test dataset following hyperparameter adjustment. The performance enhancement was observed by comparing the obtained results with the baseline MLP model. The confusion matrix, F1-score, accuracy, precision, and recall were the evaluation measures that were employed.

*6. Long Short-Term Memory (LSTM)*
The *initial implementation* of LSTM model was designed to provide a baseline performance, demonstrating how well the LSTM can capture patterns in the time-series data without additional complexity.

- *Dataset Preparation*: The pre-processed data was split into training and test sets using an 80/20 split. Each sample in the dataset consists of a sequence of features, with the label indicating the class of the object being analysed.
- *Data Reshaping:* LSTM models require the input data to be in a three-dimensional format, where the dimensions represent (samples, timesteps, features). To accommodate this, the 2D feature matrix was reshaped into 3D by adding a time dimension.
- *Model Architecture:* The basic LSTM architecture comprised two LSTM layers. The first LSTM layer had 100 units and used tanh activation. A second dense layer was added with a softmax activation function for multiclass classification.
- *Training Process:* The LSTM model was trained using the Adam optimizer and categorical cross-entropy loss function, with 10 epochs and a batch size of 32. Training was validated on the test set after each epoch, allowing us to monitor the model's performance and detect overfitting.
- *Evaluation:* The model's performance was evaluated using several classification metrics such as accuracy, precision, recall, and F1-score. The confusion matrix provided insights into the model's ability to differentiate between different material types.
- *Model Performance:* The LSTM's ability to capture temporal dependencies made it particularly suitable for this classification task. The basic model, though lacking fine-tuning, demonstrated reasonable classification accuracy, confirming its suitability for signal classification tasks.

To improve performance, hyperparameter tuning was introduced using Random Search via the Keras Tuner. This allowed the model to automatically search for the optimal hyperparameters, significantly enhancing its predictive capabilities.

*Hyperparameter Tuning*
Key hyperparameters that were tuned include:

- *Number of LSTM units:* The number of hidden units in each LSTM layer.
- *Dropout Rate:* Dropout was added between the LSTM layers to prevent overfitting. The dropout rate defines the percentage of units to drop during training.
- *Optimizer:* Both the Adam and RMSProp optimizers were tested to see which would yield better performance.
- *Batch Size:* The batch size during training was varied to explore its impact on training speed and model performance.

a) Hyperparameter Tuning Process
The Random Search approach from Keras Tuner was used to search through different combinations of these hyperparameters. Random Search is efficient because it does not exhaustively test every combination of hyperparameters but instead randomly samples from the hyperparameter space. The search process involved running 5 trials with 3 executions per trial, aiming to find the best combination of hyperparameters that maximized validation accuracy.

b) Optimized Model Architecture
The best-performing model included two LSTM layers with units ranging from 32 to 128, followed by dropout layers with rates between 0.2 and 0.5 to prevent overfitting. The final dense layer used a softmax activation function for multiclass classification, outputting the probabilities for each class.

Evaluation of the Optimized LSTM Model
After tuning, the optimized model was evaluated on the test set. The tuned model showed significant improvements over the baseline LSTM, especially in terms of accuracy and generalization. It was able to more accurately classify the materials from the signal data.

Model Evaluation and Results

The optimized MLP model was assessed using the test dataset following hyperparameter adjustment. The performance enhancement was observed by comparing the obtained results with the baseline MLP model. The confusion matrix, F1-score, accuracy, precision, and recall were the evaluation measures that were employed.

Handling Overfitting and Regularization

To avoid overfitting, several techniques were employed:

- *Dropout:* Dropout layers with varying dropout rates were used between the LSTM layers. This technique randomly drops a fraction of the units during training, preventing the model from becoming too reliant on specific neurons.
- *Early Stopping:* Early stopping was used during training to halt the process if the validation performance plateaued or began to degrade, ensuring that the model did not overfit to the training data.

## IV. RESULT AND ANALYSIS

### *A. Model Performance for Object Classification*

The performance of various machine learning models developed for ultrasonic signal classification is evaluated. The results include confusion matrices and comparative metrics that help in understanding how well each model performs in classifying the different materials.

#### *1. K-Nearest Neighbours (KNN)*

The K-Nearest Neighbours (KNN) model was implemented to classify ultrasonic signals from various objects. The analysis of the model's performance is divided into two stages: one without hyperparameter tuning and one with Randomized Search hyperparameter tuning.

a) Performance Without Hyperparameter Tuning

In its default setup, the KNN classifier had an overall accuracy of 82%. The model excelled at classifying the aluminium class, achieving near-perfect precision and recall of 0.99. The plain wood and steel classes also performed well, with f1-scores of 0.92 and 0.86, respectively. However, the model struggled to differentiate between wood with sparse needle coverage and wood with dense needle coverage, where precision and recall declined to 0.65-0.72.

Classification Report for KNN Classifier:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| aluminum | 0.99 | 0.99 | 0.99 | 200 |
| plain wood | 0.92 | 0.91 | 0.92 | 200 |
| steel | 0.82 | 0.91 | 0.86 | 200 |
| wood with lessneedles | 0.72 | 0.66 | 0.69 | 200 |
| wood with lotofneedles | 0.65 | 0.65 | 0.65 | 200 |
| accuracy | | | 0.82 | 1000 |
| macro avg | 0.82 | 0.82 | 0.82 | 1000 |
| weighted avg | 0.82 | 0.82 | 0.82 | 1000 |

Fig.19. Classification Report – KNN without hyperparameters

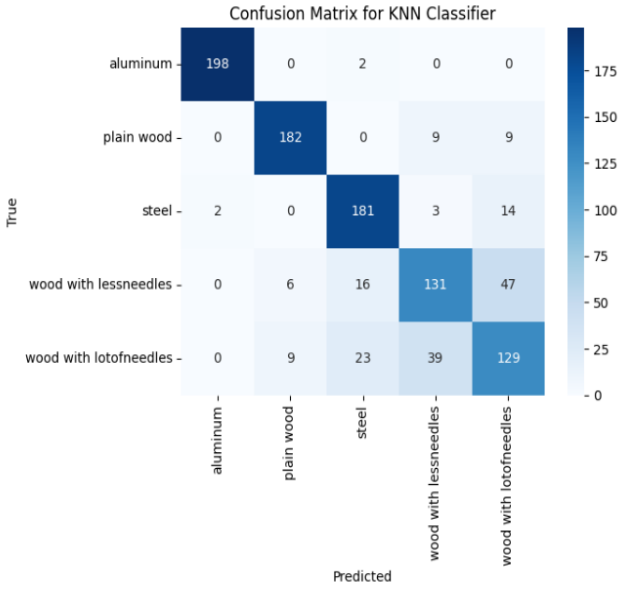


Fig.20. Confusion Matrix – KNN without hyperparameters

The confusion matrix indicates that the model accurately classified 198 out of 200 samples for the aluminium class, demonstrating its high reliability in identifying this material. However, the model struggled to distinguish between the two wood categories. Specifically, 16 samples of densely needled wood were mistakenly classified as steel, and 47 samples of sparsely needled wood were incorrectly predicted as densely needled wood. This suggests that the KNN model had difficulty differentiating between the subtle variations in the ultrasonic signals produced by the different wood types.

b) Performance With Hyperparameter Tuning (Randomized Search)

After tuning the hyperparameters using Randomized Search, the overall accuracy increased slightly to 83%. The aluminium class continued to perform exceptionally well, while the wood classes with both sparse and dense needle coverage saw minor improvements. Specifically, the precision for the wood class with dense needle coverage rose from 0.65 to 0.69, and the wood class with sparse needle coverage increased from 0.72 to 0.73. These improvements led to better recall and F1-scores for the wood classes. The best hyperparameters for the KNN model were {'metric': 'manhattan', 'n_Neighbours': 17, 'weights': 'distance'}.

Classification Report for K-Nearest Neighbors (Randomized Search Tuned):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.94 | 0.91 | 0.92 | 200 |
| 2 | 0.85 | 0.91 | 0.88 | 200 |
| 3 | 0.73 | 0.66 | 0.69 | 200 |
| 4 | 0.65 | 0.69 | 0.67 | 200 |
| accuracy | | | 0.83 | 1000 |
| macro avg | 0.83 | 0.83 | 0.83 | 1000 |
| weighted avg | 0.83 | 0.83 | 0.83 | 1000 |

Fig.21. Classification Report – KNN with hyperparameters

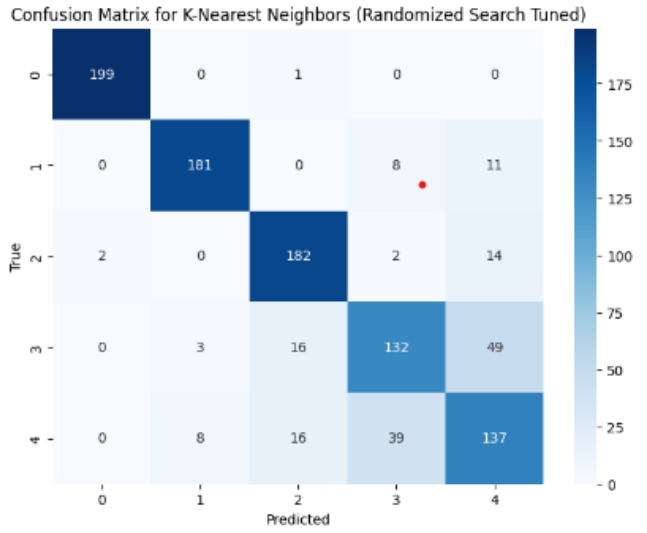


Fig.22. Confusion Matrix – KNN with hyperparameters

The confusion matrix for the optimized model offers a graphical depiction of the enhancement. There is a discernible decrease in misclassifications for the categories of wood with dense needle coverage and wood with sparse needle coverage; The quantity of samples wrongly classified between these two wood groups has diminished, suggesting improved differentiation between these analogous classes. The aluminium and plain wood categories maintained their high classification accuracy with only minor misclassifications. The distinct ultrasonic signatures of these materials led to impressive f1-scores. While hyperparameter optimization reduced misclassification errors for the wood with sparse needle coverage and wood with dense needle coverage classes, challenges persist due to the overlap in signal features. Although the overall accuracy increased only slightly from 82% to 83%, the reduced misclassifications for the challenging wood classes indicate that hyperparameter tuning enhanced the model's robustness.

*2. Support Vector Machine (SVM)*

The Support Vector Machine (SVM) model was implemented to classify ultrasonic signals from various objects, with two stages of analysis: one without hyperparameter tuning and one with hyperparameter tuning using both Grid Search and Randomized Search.

a) Performance Without Hyperparameter Tuning

In its initial setup, the SVM classifier produced an overall accuracy of 82%. The model demonstrated excellent performance for the aluminium class, with a precision and recall of 0.99, indicating near-flawless classification. The plain wood and steel classes also showed high f1-scores of 0.91 and 0.87, respectively. However, like the KNN model, the SVM struggled to distinguish between the wood with sparse needle coverage and wood with dense needle coverage classes, where precision and recall fell within the range of 0.59-0.72.

--- Classification Report for Support Vector Machine ---

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.95 | 0.88 | 0.91 | 200 |
| 2 | 0.84 | 0.90 | 0.87 | 200 |
| 3 | 0.68 | 0.72 | 0.70 | 200 |
| 4 | 0.63 | 0.59 | 0.61 | 200 |
| accuracy | | | 0.82 | 1000 |
| macro avg | 0.82 | 0.82 | 0.82 | 1000 |
| weighted avg | 0.82 | 0.82 | 0.82 | 1000 |

Fig.23. Classification Report – SVM without hyperparameters

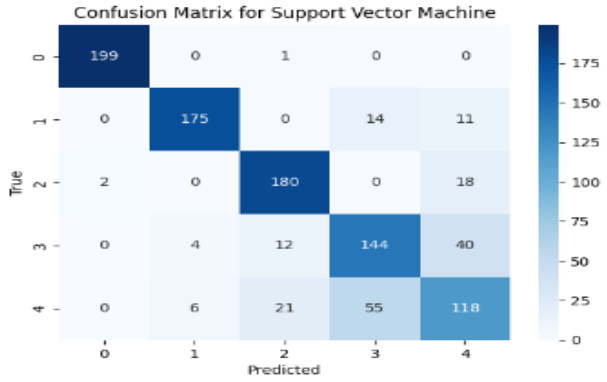


Fig.24. Confusion Matrix – SVM without hyperparameters

The confusion matrix for this setup demonstrates where the SVM model misclassified samples. For instance, 14 samples of wood with sparse needle coverage were incorrectly identified as steel, and 55 samples of wood with dense needle coverage were misclassified as wood with sparse needle coverage. The confusion matrix reveals that while the SVM model successfully classified aluminium and plain wood, it struggled with objects that produced similar ultrasonic signals, especially the wood categories.

b) Performance With Hyperparameter Tuning (Grid Search)

After applying hyperparameter tuning using Grid Search (C=1000, degree=2, gamma=1, kernel=rbf), the model's accuracy improved to 86%. Significant improvements were noted in the recall and precision of the wood categories, which had been problematic in the untuned model. For example, the f1-score for wood with dense needle coverage improved to 0.73, and for wood with sparse needle coverage, it increased to 0.74.

--- Classification Report for Optimized SVM (Grid Search) ---

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.98 | 0.98 | 200 |
| 1 | 0.97 | 0.94 | 0.95 | 200 |
| 2 | 0.89 | 0.93 | 0.91 | 200 |
| 3 | 0.77 | 0.71 | 0.74 | 200 |
| 4 | 0.78 | 0.76 | 0.73 | 200 |
| accuracy | | | 0.86 | 1000 |
| macro avg | 0.86 | 0.86 | 0.86 | 1000 |
| weighted avg | 0.86 | 0.86 | 0.86 | 1000 |

Fig.25. Classification Report – SVM (Grid Search)

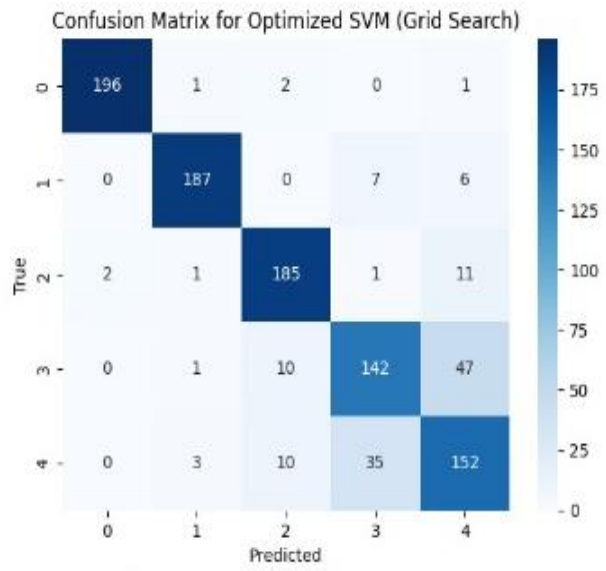


Fig.26. Confusion Matrix – SVM (Grid Search)

There is a noticeable reduction in misclassifications in the confusion matrix for the Grid Search-tuned model, particularly for wood with dense and sparse needle coverage. The model's improved capacity to distinguish between these two wood groups was demonstrated by the decline in misclassifications between them. This implies that fine-tuning the hyperparameters improved the resilience of the model and decreased the amount of misclassifications in the wood categories that are challenging to categorize.

c) Performance With Hyperparameter Tuning (Randomized Search)

The model's accuracy increased to 86.3% using Randomized Search for hyperparameter tweaking (C=100, degree=4, gamma=1, kernel=rbf), which is marginally better than the outcome using Grid Search. With a f1-score of 0.74, the recall and precision for wood with dense needle coverage further increased. Misclassifications were reduced across most classes, particularly for wood categories.

--- Classification Report for Optimized SVM (Random Search) ---

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.95 | 0.93 | 0.94 | 200 |
| 2 | 0.89 | 0.94 | 0.91 | 200 |
| 3 | 0.78 | 0.70 | 0.74 | 200 |
| 4 | 0.70 | 0.76 | 0.73 | 200 |
| accuracy | | | 0.86 | 1000 |
| macro avg | 0.86 | 0.86 | 0.86 | 1000 |
| weighted avg | 0.86 | 0.86 | 0.86 | 1000 |

Fig.27. Classification Report – SVM (Randomized Search)

The confusion matrix for Randomized Search reflects these improvements. The model performed better overall in differentiating between different types of wood, with misclassifications between wood with sparse needle coverage and wood with dense needle coverage continuing to decline.

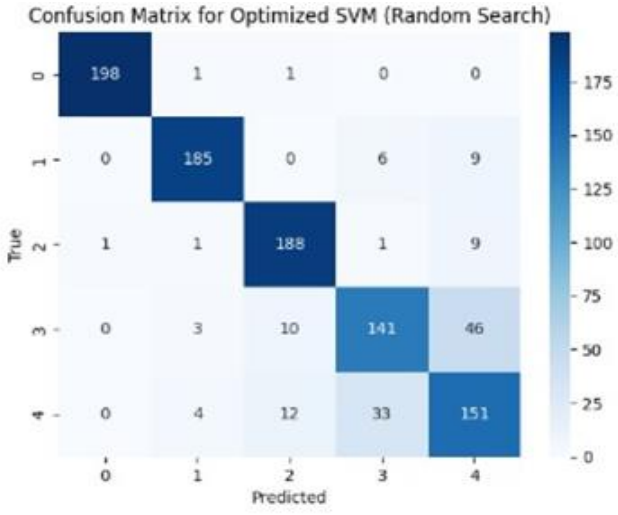


Fig.28. Confusion Matrix – SVM (Random Search)

Untuned and tuned models both performed well in accurately classifying aluminium and plain wood, with only a small number of misclassifications. This was due to the distinct ultrasonic signal characteristics of these materials, which enabled the models to achieve high precision and recall scores. Although the overall accuracy increased from 82% to 86.3% after tuning, the most significant improvement was the reduction in misclassifications for the wood categories. This indicates that hyperparameter tuning was instrumental in making the model more robust, especially for challenging classification tasks.

*3. Random Forest*

The Random Forest model was implemented to classify ultrasonic signals from various objects. The analysis of the model's performance is divided into two stages: one without hyperparameter tuning and one with Grid Search, Randomized Search, Hyperband Search and for Bayesian Search hyperparameter tuning.

a) Performance Without Hyperparameter Tuning

In its default configuration, the classifier achieved an overall accuracy of 88%**.** The model performed exceptionally well for the aluminium class, with a precision and recall of 0.99, indicating near-perfect classification. Similarly, the plain wood and steel classes achieved high f1-scores of 0.94 and 0.93, respectively. However, the model had more difficulty classifying wood with sparse needle coverage and wood with dense needle coverage, where precision and recall dropped to 0.75-0.78**.**

Classification Report for Random Forest Classifier:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| aluminum | 0.99 | 0.99 | 0.99 | 200 |
| plain wood | 0.95 | 0.93 | 0.94 | 200 |
| steel | 0.91 | 0.94 | 0.93 | 200 |
| wood with lessneedles | 0.78 | 0.75 | 0.76 | 200 |
| wood with lotofneedles | 0.75 | 0.77 | 0.76 | 200 |
| accuracy | | | 0.88 | 1000 |
| macro avg | 0.88 | 0.88 | 0.88 | 1000 |
| weighted avg | 0.88 | 0.88 | 0.88 | 1000 |

Fig.29. Classification Report – Random Forest without hyperparameters

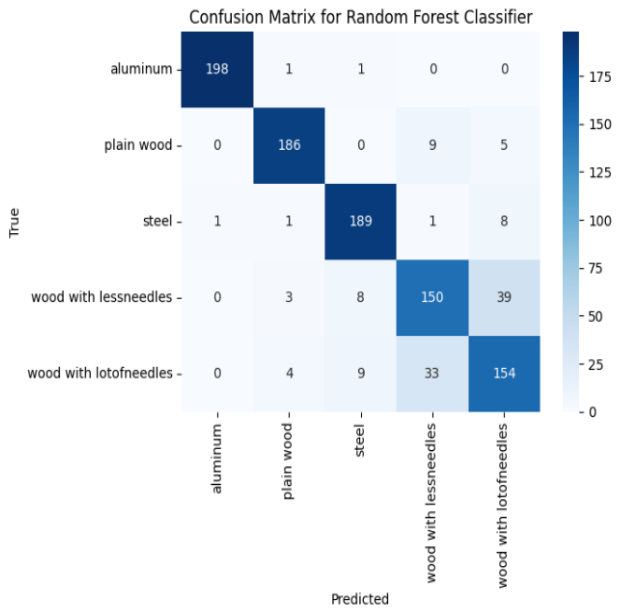


Fig.30. Confusion Matrix – Random Forest without hyperparameters

The confusion matrix provides further insights. For example, aluminium and steel were classified with almost no errors, but for wood with dense needle coverage and wood with sparse needle coverage, significant misclassifications occurred. Specifically, 39 samples of wood with sparse needle coverage were misclassified as wood with dense needle coverage, and 33 samples of wood with dense needle coverage were predicted as wood with sparse needle coverage.

b) Performance With Hyperparameter Tuning (Grid Search)
After tuning with Grid Search, the accuracy increased to 89%. The confusion matrix showed fewer misclassifications between the wood categories, and the precision for wood with sparse needle coverage improved slightly to 0.74, with recall rising to 0.80. The best hyperparameters identified by Grid Search were: {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 100}. The confusion matrix reveals that the number of misclassified wood with sparse needle coverage samples reduced from 39 to 26, showing improved differentiation between the wood categories.

Classification Report for Tuned Random Forest (Grid Search):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.96 | 0.94 | 0.95 | 200 |
| 2 | 0.91 | 0.96 | 0.94 | 200 |
| 3 | 0.82 | 0.72 | 0.77 | 200 |
| 4 | 0.74 | 0.80 | 0.77 | 200 |
| accuracy | | | 0.89 | 1000 |
| macro avg | 0.89 | 0.89 | 0.88 | 1000 |
| weighted avg | 0.89 | 0.89 | 0.88 | 1000 |

Fig.31. Classification Report – Random Forest (Grid Search)

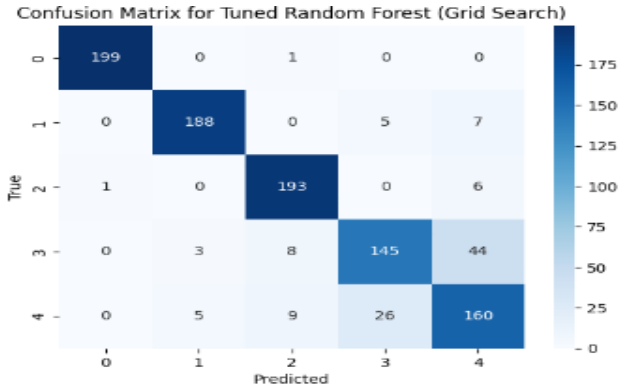


Fig.32. Confusion Matrix – Random Forest (Grid Search)

c) Performance With Hyperparameter Tuning (Randomized Search)

The Randomized Search approach yielded similar improvements, resulting in an overall accuracy of 88%. The model continued to perform well for aluminium and steel, but still faced some challenges with the wood categories. However, the precision and recall for wood with sparse needle coverage showed slight improvements, indicating enhanced robustness. The best hyperparameter configuration found through Random Search was: 'n_estimators': 50, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_features': 'auto', 'max_depth': None, 'criterion': 'gini', 'bootstrap': True. The confusion matrix revealed fewer misclassifications between wood with dense needle coverage and wood with sparse needle coverage, compared to the model without hyperparameter tuning. Specifically, the number of misclassified wood with dense needle coverage samples decreased from 33 to 26.

Classification Report for Tuned Random Forest (Random Search):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.96 | 0.94 | 0.95 | 200 |
| 2 | 0.91 | 0.96 | 0.94 | 200 |
| 3 | 0.83 | 0.72 | 0.77 | 200 |
| 4 | 0.73 | 0.80 | 0.77 | 200 |
| accuracy | | | 0.89 | 1000 |
| macro avg | 0.89 | 0.89 | 0.88 | 1000 |
| weighted avg | 0.89 | 0.89 | 0.88 | 1000 |

Fig.33. Classification Report – Random Forest (Random Search)

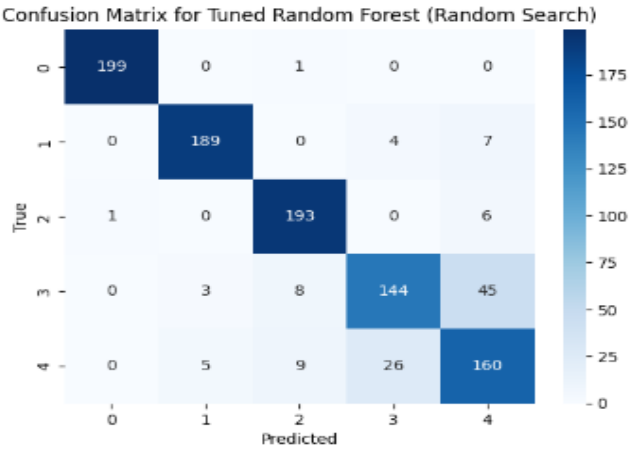


Fig.34. Confusion Matrix – Random Forest (Random Search)

d) Performance With Hyperparameter Tuning (Bayesian Search Tuning)

Classification Report for Tuned Random Forest (Bayesian Search):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.99 | 0.99 | 200 |
| 1 | 0.96 | 0.94 | 0.95 | 200 |
| 2 | 0.91 | 0.96 | 0.94 | 200 |
| 3 | 0.81 | 0.72 | 0.76 | 200 |
| 4 | 0.73 | 0.79 | 0.75 | 200 |
| accuracy | | | 0.88 | 1000 |
| macro avg | 0.88 | 0.88 | 0.88 | 1000 |
| weighted avg | 0.88 | 0.88 | 0.88 | 1000 |

Fig. 35. Classification Report – Random Forest (Bayesian Search)

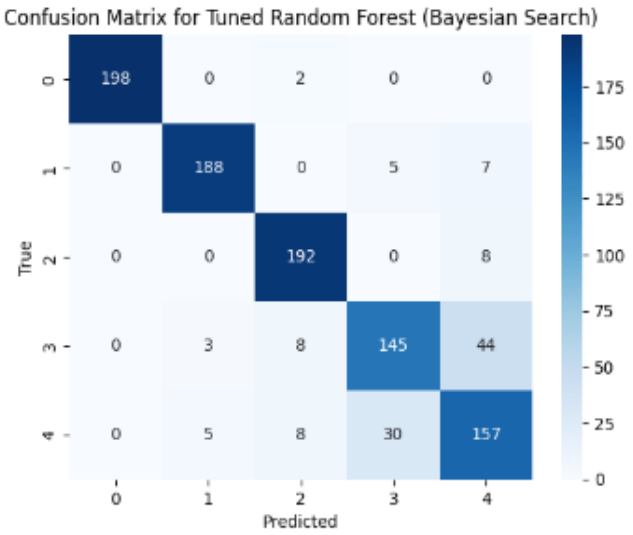


Fig. 36. Confusion Matrix – Random Forest (Bayesian Search)

The Bayesian Search method achieved an overall accuracy of 88%. While the precision and recall for aluminium, plain wood, and steel remained high, Bayesian Search provided slightly better recall in the wood categories compared to the untuned model. The optimal hyperparameters identified through Bayesian Optimization were 'bootstrap': True, 'criterion': 'entropy', 'max_depth': 50, 'min_samples_leaf': 4, 'min_samples_split': 12, 'n_estimators': 119.

The confusion matrix for Bayesian Search shows a further reduction in misclassifications between the wood categories, with fewer misclassified samples in the wood with sparse needle coverage and wood with dense needle coverage classes.

The Random Forest model achieved near-perfect precision and recall on the steel and aluminium classifications, performing well across all tuning approaches. Although the overall accuracy increased little (from 88% to 89%) after hyperparameter adjustment, the decrease in misclassifications for the wood categories showed that the model's robustness was strengthened, especially in difficult classification tasks.

e) Performance With Hyperparameter Tuning (Hyperband Search)

Classification Report for Tuned Random Forest (Hyperband Search):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.99 | 1.00 | 200 |
| 1 | 0.96 | 0.94 | 0.95 | 200 |
| 2 | 0.91 | 0.95 | 0.93 | 200 |
| 3 | 0.81 | 0.72 | 0.77 | 200 |
| 4 | 0.73 | 0.79 | 0.76 | 200 |
| accuracy | | | 0.88 | 1000 |
| macro avg | 0.88 | 0.88 | 0.88 | 1000 |
| weighted avg | 0.88 | 0.88 | 0.88 | 1000 |

Fig.37. Classification Report – Random Forest (Hyperband Search)

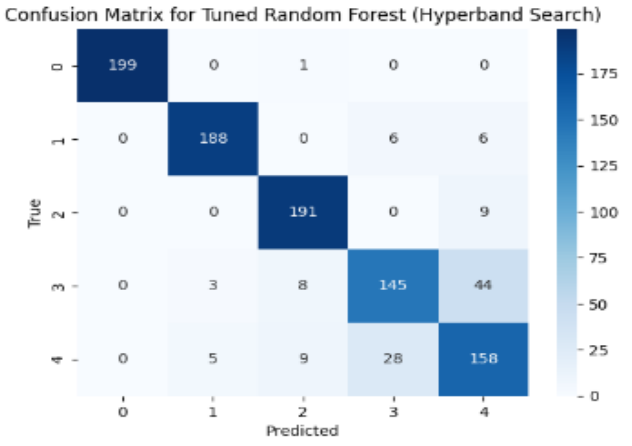


Fig.38. Confusion Matrix – Random Forest (Hyperband Search)

Hyperband Search also produced similar results, with an accuracy of 88%. The confusion matrix for this method showed a slight reduction in misclassification errors for the wood categories. The best hyperparameters found through Hyperband Search were: {'bootstrap': True, 'criterion': 'entropy', 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 300}.

The confusion matrix highlights the reduction in misclassification errors between wood with sparse needle coverage and wood with dense needle coverage, showing more accurate differentiation in these challenging categories.

*4. Multilayer Perceptron (MLP)*

The Multilayer Perceptron (MLP) model like other models, the performance analysis is divided into two stages: one without hyperparameter tuning and one with hyperparameter tuning using Randomized Search.

a) Performance Without Hyperparameter Tuning

In its default configuration, the MLP classifier achieved an overall accuracy of 86%. With precision and recall for the aluminium class at 0.99, which indicates almost perfect categorization, the model did remarkably well. The steel and plain wood classes performed very well as well, with f1-

scores of 0.92 and 0.96, respectively. However, the model struggled with wood-related categories. In particular, the f1-score of wood with dense needle coverage was 0.73, whereas the f1-score of wood with sparse needle coverage class was 0.72. The confusion matrix for the untuned MLP model reveals the misclassification trends. The decline in f1-scores, for instance, can be explained by the large number of samples (61) in the wood with dense needle coverage class that were incorrectly classified as having sparse needle coverage. Similarly, it was common to mistakenly classify wood with poor needle covering for wood with rich needle coverage. These findings suggest that the model had trouble differentiating between various wood kinds.

Classification Report for Multilayer Perceptron:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.97 | 0.94 | 0.96 | 200 |
| 2 | 0.90 | 0.94 | 0.92 | 200 |
| 3 | 0.83 | 0.64 | 0.72 | 200 |
| 4 | 0.67 | 0.81 | 0.73 | 200 |
| accuracy | | | 0.86 | 1000 |
| macro avg | 0.87 | 0.86 | 0.86 | 1000 |
| weighted avg | 0.87 | 0.86 | 0.86 | 1000 |

Fig. 39. Classification Report – MLP without hyperparameters

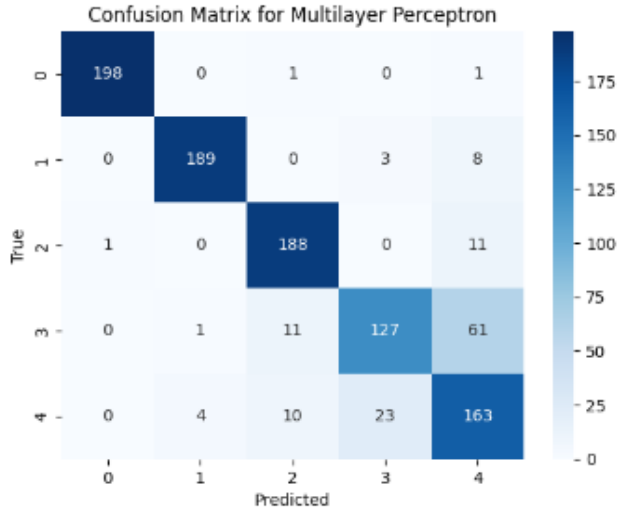


Fig. 40. Confusion Matrix – MLP without hyperparameters

b) Performance With Hyperparameter Tuning (Randomized Search)

Classification Report for Multi-Layer Perceptron (Randomized Search):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.95 | 0.95 | 0.95 | 200 |
| 2 | 0.91 | 0.94 | 0.93 | 200 |
| 3 | 0.73 | 0.80 | 0.76 | 200 |
| 4 | 0.75 | 0.67 | 0.71 | 200 |
| accuracy | | | 0.87 | 1000 |
| macro avg | 0.87 | 0.87 | 0.87 | 1000 |
| weighted avg | 0.87 | 0.87 | 0.87 | 1000 |

Fig. 41. Classification Report – MLP (Randomized Search)

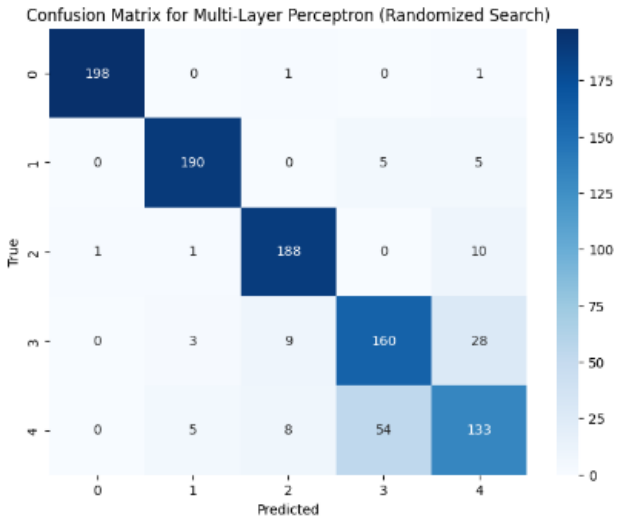


Fig. 42. Confusion Matrix – MLP (Randomized Search)

After hyperparameter tuning using Randomized Search the total accuracy increased marginally to 87%. The aluminium class continued to operate almost flawlessly, maintaining a high level of recall and precision at 0.99. For the wood classes, there were marginal gains in recall and precision. For instance, the f1-score increased from 0.72 to 0.76 for wood with scant needle coverage and from 0.73 to 0.71 for wood with high needle coverage. 'solver': 'adam', 'learning_rate': 'constant', 'hidden_layer_sizes': (100, 100), 'alpha': 0.0001, 'activation':'relu' were the best hyperparameters for MLP.

The confusion matrix for the tuned model illustrates a noticeable improvement in classifying the wood with dense needle coverage class. There was a decrease in the amount of samples that were incorrectly identified as belonging to the densely needled wood and the sparsely needled wood classes, suggesting improved separation between these related wood classes. This is the direct outcome of enhancing the learning process for these overlapping categories through hyperparameter adjustment.

Although the overall accuracy increase was only small (from 86% to 87%), the increases in f1-scores and decreased misclassifications for the wood classes demonstrate that hyperparameter adjustment contributed to enhanced model robustness.

*5. Convolutional Neural Networks (CNN)*

The Convolutional Neural Network (CNN) model ‘s performance of the model is evaluated in two stages: one without hyperparameter tuning and one with hyperparameter tuning.

a) Performance Without Hyperparameter Tuning

In the initial configuration, the CNN classifier achieved an overall accuracy of 77%. The aluminium class achieved recall of 0.99 and near-perfect precision, demonstrating outstanding performance. The class for simple wood did well as well, with a f1-score of 0.91. Nevertheless, the model had trouble differentiating between the wood with sparse needle coverage class and the wood with dense needle coverage

class, with the latter obtaining a comparatively low f1-score of 0.63.

The confusion matrix reveals that for the wood categories, there were significant misclassifications. For instance, 61 samples were mistakenly classified as steel and 33 samples from the category of wood with sparse needle coverage were mistakenly classified as wood with dense needle coverage. These incorrect classifications demonstrate how challenging it is to discern minute differences in ultrasonic signals between various wood groups.

Classification Report for Convolutional Neural Network (CNN):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.93 | 0.89 | 0.91 | 200 |
| 2 | 0.73 | 0.79 | 0.75 | 200 |
| 3 | 0.59 | 0.69 | 0.63 | 200 |
| 4 | 0.63 | 0.49 | 0.56 | 200 |
| accuracy | | | 0.77 | 1000 |
| macro avg | 0.77 | 0.77 | 0.77 | 1000 |
| weighted avg | 0.77 | 0.77 | 0.77 | 1000 |

Fig. 43. Classification Report – CNN without hyperparameters

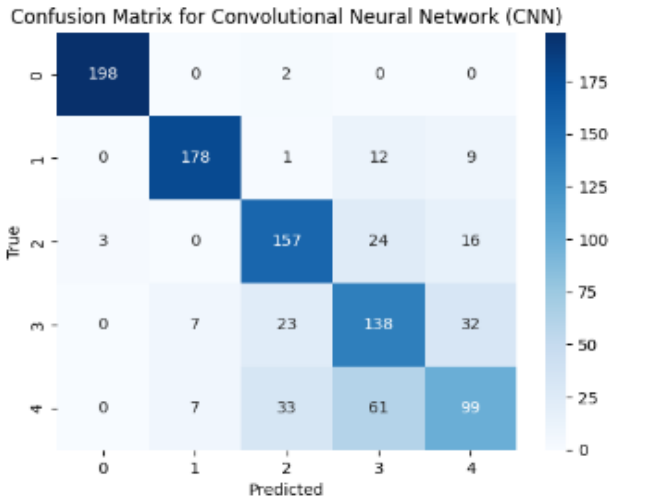


Fig. 44. Confusion Matrix – CNN without hyperparameters

b) Performance With Hyperparameter Tuning

Classification Report for CNN:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.93 | 0.94 | 0.93 | 200 |
| 2 | 0.88 | 0.95 | 0.91 | 200 |
| 3 | 0.86 | 0.59 | 0.70 | 200 |
| 4 | 0.67 | 0.81 | 0.73 | 200 |
| accuracy | | | 0.86 | 1000 |
| macro avg | 0.87 | 0.86 | 0.86 | 1000 |
| weighted avg | 0.87 | 0.86 | 0.86 | 1000 |

Fig. 45. Classification Report – CNN (Tuned)

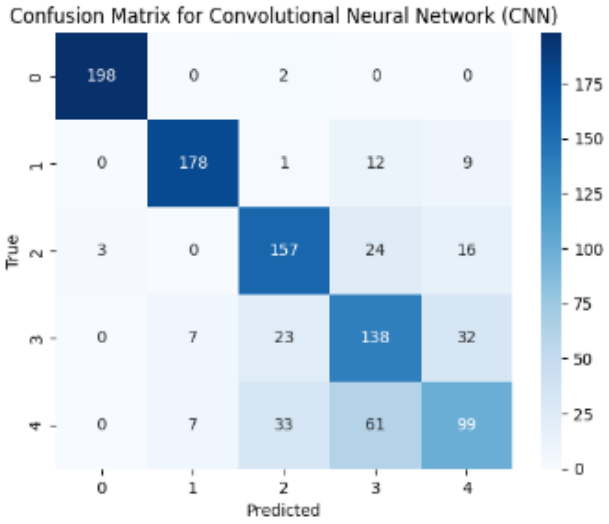


Fig. 46. Confusion Matrix – CNN (Tuned)

Following hyperparameter adjustment, the CNN model's accuracy increased dramatically to 86%. All categories saw improvements in precision and recall, with the wood categories seeing the biggest gains. For example, the f1-score went from 0.56 to 0.73 for wood with dense needle covering and from 0.63 to 0.70 for wood with sparse needle coverage.

The confusion matrix shows that misclassifications for the wood with dense needle coverage category were reduced, with fewer samples being misclassified as wood with sparse needle coverage or steel. Overall, there was a marked improvement in correctly identifying these wood categories compared to the initial model.

By comparing the results, it's evident that while the CNN model performed reasonably well without tuning, hyperparameter tuning was essential to enhance its robustness, particularly for the challenging wood categories. This improvement was reflected in the f1-scores and the reduction of misclassified samples in the confusion matrix.

*6. Long Short-Term Memory (LSTM)*

The Long Short-Term Memory (LSTM) model was evaluated in two stages: without hyperparameter tuning and with tuning using Randomized Search. Key performance indicators include precision, recall, f1-score, and the confusion matrix.

a) Performance Without Hyperparameter Tuning

The total accuracy of the LSTM classifier was 82% without the need for hyperparameter adjustment. With precision and recall scores of 0.99 and 0.95 for aluminium and plain wood, respectively, the model did well in classification tasks. Recall for the steel class was 0.91, and the f1-score was 0.87, suggesting that performance in recognizing steel was reasonable. But the model could not tell the difference between wood with lots of needles and wood with few needles. In particular, the precision and recall decreased to 0.63 and 0.70 for wood with fewer needles.

Classification Report for Long Short-Term Memory (LSTM):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.97 | 0.98 | 200 |
| 1 | 0.95 | 0.90 | 0.92 | 200 |
| 2 | 0.83 | 0.91 | 0.87 | 200 |
| 3 | 0.74 | 0.64 | 0.68 | 200 |
| 4 | 0.63 | 0.70 | 0.66 | 200 |
| accuracy | | | 0.82 | 1000 |
| macro avg | 0.83 | 0.82 | 0.82 | 1000 |
| weighted avg | 0.83 | 0.82 | 0.82 | 1000 |

Fig. 47. Classification Report – LSTM without hyperparameters

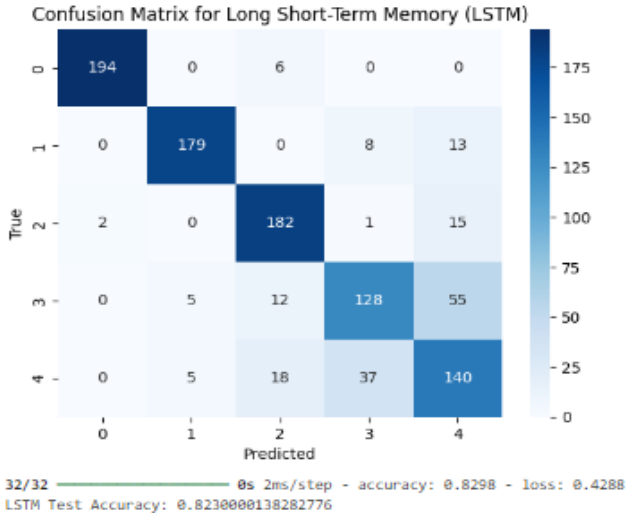


Fig. 48. Confusion Matrix – LSTM without hyperparameters

The confusion matrix revealed that for aluminium, six samples of aluminium were incorrectly identified as steel. There was a significant overlap in the forecasts for the wood with needles category, with wood with a lot of needles being incorrectly classified as other wood types. Without adjusting the hyperparameters, the confusion matrix shows that the model did a good job of differentiating between the aluminium and plain wood classes. It incorrectly identified 55 samples of wood with more needles as having fewer needles, nevertheless. This implies that a high level of classification confusion resulted from the LSTM model's inability to handle the minute variations in signal characteristics between different wood types when it was left untuned.

b) Performance With Hyperparameter Tuning

Classification Report for LSTM (Tuned):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 200 |
| 1 | 0.93 | 0.92 | 0.93 | 200 |
| 2 | 0.88 | 0.95 | 0.92 | 200 |
| 3 | 0.77 | 0.70 | 0.73 | 200 |
| 4 | 0.70 | 0.72 | 0.71 | 200 |
| accuracy | | | 0.86 | 1000 |
| macro avg | 0.86 | 0.86 | 0.86 | 1000 |
| weighted avg | 0.86 | 0.86 | 0.86 | 1000 |

Fig. 49. Classification Report – LSTM (Tuned)

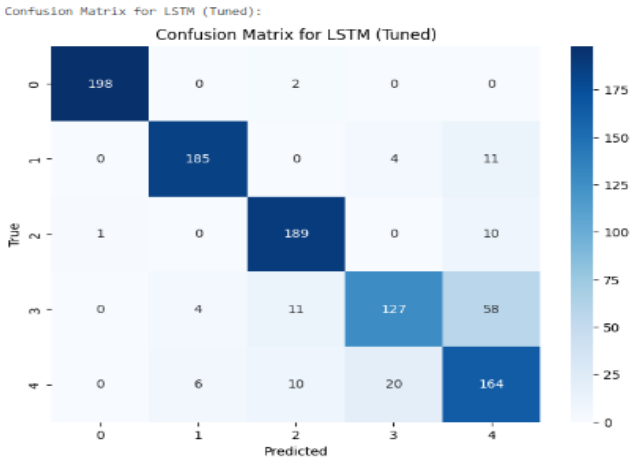


Fig. 50. Confusion Matrix – LSTM (Tuned)

After tuning the LSTM model using Randomized Search, the overall accuracy improved to 86%. Hyperparameter tuning positively impacted the classification of the wood classes, specifically reducing the misclassification rate between wood with more and fewer needles. Precision for wood with more needles increased to 0.67, with a recall improvement to 0.82.

The number of wood with more needles misclassified as wood with fewer needles decreased to 20 from 55. This improvement is indicative of how well hyperparameter optimization has worked to clear up any confusion among these closely related classes. Both tuned and untuned models performed consistently, achieving great recall and precision in differentiating between aluminium and plain wood. Differentiating between wood types with fewer and more needles presents the LSTM model with its biggest obstacle. Despite the fact that tweaking increased performance, there is still some overlap because these classes' signal patterns are similar. The adjusted LSTM model demonstrated a discernible enhancement in classification performance and overall accuracy for challenging classes, particularly in the wood categories.

## 7. Comparison of Model Performance Before and After Hyperparameter Tuning

Without hyperparameter tuning, the Random Forest classifier emerged as the top performance, achieving an accuracy of 87.7%. It showed efficient and fair classification in every category. With an accuracy of 86.5%, the Multilayer Perceptron trailed closely behind, performing well in easier categories but struggling in harder ones. Similar accuracy was obtained with SVM and K-Nearest Neighbours, with SVM receiving an 81.6% score and KNN receiving an 82.1%. While these models were effective at differentiating between discrete categories, they were less successful in classes where attributes overlapped, especially in the wood categories. With an accuracy of 77.4%, CNN demonstrated the lowest performance in this instance, whereas LSTM fared rather well, attaining 80.9%.

After applying hyperparameter tuning, all models saw improvements in performance. Using Grid Search and

Random Search, the Random Forest classifier continued to be the top model, achieving an accuracy of 89%. Among the many object categories, this model showed the highest stability and dependability. SVM, MLP, CNN, and LSTM all saw notable improvements, with the latter two achieving 86% accuracy. KNN witnessed a minor increase as well, adjusting to 83%. Among these models, the Convolutional Neural Network demonstrated the most noticeable increase, gaining over 9% in accuracy compared to its untuned performance.

In conclusion, the Random Forest classifier proved to be the most effective model overall, consistently delivering the highest accuracy both before and after hyperparameter tuning. While other models such as KNN, SVM, MLP, CNN, and LSTM performed well after tuning, Random Forest emerged as the most robust and reliable model for this classification task.

## V. CONCLUSION

The objective of this project was to investigate the classification of ultrasonic signals using various machine learning models, with a focus on the impact of hyperparameter tuning. The study utilized the Red Pitaya STEMLAB measurement board and the Ultrasonic sensor SRF02 to capture ultrasonic signals from different materials. We found that ultrasonic waves behave differently when interacting with various materials, as demonstrated by distinct patterns in the reflected signals. The Red Pitaya board was instrumental in collecting these signals, enabling us to extract meaningful features and understand the underlying behaviours of the waves.

To enhance the classification of these signals, feature extraction techniques such as Signal-to-Noise and Distortion Ratio (SINAD), peak position and peak count, and autocorrelation for periodicity detection were applied. These features provided key insights into the structure of the signals, which played a crucial role in improving the models' ability to classify materials based on their ultrasonic signal patterns. The classification behaviour observed through our experiments demonstrated noticeable variances, particularly when experimental setups were adjusted. This highlights the crucial role of feature engineering in improving model performance.

In our comparative analysis, each model demonstrated unique strengths in handling the data from the 1-meter ultrasonic sensor measurements. Without hyperparameter tuning, the Random Forest classifier emerged as the best performer with an accuracy of 87.7%, followed closely by the Multilayer Perceptron (MLP) with 86.5%. Other models, such as K-Nearest Neighbours (KNN), Support Vector Machine (SVM), CNN, and LSTM, performed reasonably well but showed room for improvement. After hyperparameter tuning, most models saw substantial performance gains. Random Forest remained a strong performer with accuracies of up to 89%, while models like SVM, MLP, CNN, and LSTM also achieved high accuracy scores of up to 86%, demonstrating the importance of optimizing model parameters.

In conclusion, this research underscores the importance of hyperparameter optimization and feature extraction in ultrasonic signal classification. The Red Pitaya STEMLAB measurement board, along with the SRF02 sensor, provided a robust platform for signal collection and analysis. Using advanced machine learning models and feature extraction techniques, we demonstrated significant improvements in classification accuracy across various materials. Future work could focus on expanding the dataset and further refining model architectures, particularly by incorporating pre-trained models, to further improve classification precision in real-world applications.

## VI. ACKNOWLEDGMENT

I would like to thank Prof. Dr. Andreas Pech for providing me an opportunity to work on this topic and provided me with the proper guidance to write this paper.

## VII. REFERENCES

[1] S. Shi, S. Jin, D. Zhang, J. Liao, D. Fu and L. Lin, “Improving Ultrasonic Testing by Using Machine Learning Framework Based on Model Interpretation Strategy,” *Chinese Journal of Mechanical Engineering*, vol. 36, no. 1, p. 127 , 2023.

[2] H. G. R. Z. P. M. Mohamed-Elamir Mohamed, “A machine learning approach for ultrasonic noise classification and suppression,” 2020.

[3] M. BAHY, “Classification of ultrasonic signals using machine learning to identify optimal frequency for elongation control,” Stockholm, Sweden, 2022.

[4] A. a. N. F. Moshrefi, “Advanced Industrial Fault Detection: A Comparative Analysis of Ultrasonic Signal Processing and Ensemble Machine Learning Techniques,” *Applied Sciences*, vol. 6397, p. 15, 2024.

[5] Richards, Mike, Pi Updates and Red Pitaya, Radio User. Bournemouth, UK: PW Publishing Ltd. 11, 2016.

[6] 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..

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

[8] P. N. T. Wells, “Ultrasonic attenuation measurements in liquids and solids,” *Physical Acoustics*, vol. 3, pp. 219-294, 1966.

[9] S. G. Kim, “Ultrasonic reflection and refraction at material boundaries,” *Journal of the Acoustical Society of America*, vol. 39, pp. 600-607, 1966.

[10] M. P. B. M. A. R. J. H. Arnold, “Multipath effects in ultrasonic waveforms,” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 38, pp. 314-323, 1991.

[11] A. H. a. N. P. M. a. M. R. Pech, “A new Approach for Pedestrian Detection in Vehicles by Ultrasonic Signal Analysis,” pp. 1-5, 2019.

[12] K. J. Piczak, “ENVIRONMENTAL SOUND CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS,” *IEEE 25th*

*International Workshop on Machine Learning for Signal Processing (MLSP)*, p. 6, 2015.

[13] Z. C. L. R. W. John R. Hershey, "Deep clustering: Discriminative embeddings for segmentation and separation," 2015.

[14] T. M. a. P. E. H. Cover, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, p. 21–27, 1967.

[15] A. H. C. A. Ozcan I, " Comparison of Classification Success Rates of Different Machine Learning Algorithms in the Diagnosis of Breast Cancer," *Asian Pac J Cancer Prev*, vol. 23, no. 10, pp. 3287-3297, 2022.

[16] K. C. R. L. S. H. Y. G. Fatemehalsadat Madaeni, "Convolutional neural network and long short-term memory models for ice-jam predictions," *The Cryosphere*, vol. 16, p. 1447–1468, 2022.

[17] C. a. V. V. Cortes, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.

[18] B. A. S. a. K.-R. M. Scholkopf, "Kernel principal component analysis," *Advances in Kernel Methods-Support Vector Learning*, pp. 327-352, 1999.

[19] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[20] A. a. M. W. Liaw, "Classification and regression by randomForest.," *R News*, vol. 2, no. 3, pp. 18-22, 2002.

[21] C. M. Bishop, "Pattern Recognition and Machine Learning," *Springer*, 2006.

[22] Y. B. a. G. H. Y. LeCun, "Deep learning," *Nature*, vol. 521, no. 7553 , pp. 436-444, 2015.

[23] I. S. G. E. H. A. Krizhevsky, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, Lake Tahoe, Nevada, 2012.

[24] S. a. J. S. Hochreiter, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[25] F. A. J. S. a. F. C. Gers, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451-2471, 2000.