Number Recognition by Listening—Traditional Acoustic Feature Analysis and Machine Learning Method for Estimating the Number of Balls in a Black Box

Zixuan Peng*, Xiaoxi Zhou*, Jifan Zhang
Nanjing Foreign Language School, Nanjing 210000, China
E-mail: zxpeng2007@gmail.com, xiaoxizhou27@gmail.com, 3421364813@qq.com

Abstract

The use of non-invasive methods to accurately estimate the number of objects in sealed containers remains a challenging problem in various fields of application. This paper presents a series of novel and practical methods for calculating the ball numbers in a black box based on acoustic analysis. Firstly, in order to obtain the sound of ball impact in black box quantitatively, this paper designs an experimental system that can oscillate a box left and right according to the set frequency and amplitude. On this basis, we proposed a multi-modal physics simulation model to solve for the sound energy produced by the collision system during a given time period. Secondly, acoustic feature extraction methods such as fast Fourier transform (FFT) are used to analyze sound signals collected and therefore derive sound energy from the experiments. Furthermore, a data-driven method based on a Convolutional Neural Network (CNN) is used to train the end-to-end prediction model for the Mel frequency cepstrum coefficient (MFCC), constant Q transform (CQT) and wavelet transform of audio data. This paper generates synthetic data through Python simulation, and constructs an experimentally derived data set of ball collision sound energies, verifies the experimental results and theoretical model, and explores the relationship between physical parameters and acoustic characteristics. The combined approach of classical acoustic analysis and machine learning employed in this paper allows us to leverage physical insight and data-driven capabilities to achieve accurate and reliable ball count predictions. The results demonstrate the effectiveness of combining theoretical modeling, simulation, traditional acoustic feature extraction and machine learning for non-invasive object counting, opening up new possibilities for applications in various fields such as baggage screening, automated production lines, and the pharmaceutical industry.

Keywords: Non-invasive object counting, Acoustic feature extraction, Energy estimation, Machine learning

*Z.P., X.Z.: These authors contributed equally to this work.

CONTENTS

I. Introduction

Recent advancements in artificial intelligence, particularly in the realm of machine learning, have revolutionized many areas of science and technology [1]. One such area is the development of non-invasive methods for object counting and detection. The ability to accurately estimate the number of objects within a sealed container without opening it has wide-ranging applications in fields such as industrial automation, logistics, healthcare, and security.

Traditional methods for object counting often rely on visual inspection or invasive techniques, which can be time-consuming, expensive, or even impractical in certain scenarios. Acoustic analysis, on the other hand, offers a promising non-invasive alternative for estimating object counts based on the sound generated by the objects within the container. As this is a relatively new topic, there are few works directly related. Here highlighted are three of th ‘e most relevant studies.

First, the paper (2015) [2] titled Multiple Sound Source Location Estimation and Counting in a Wireless Acoustic Sensor Network presents a method based on inferring a location estimate for each frequency of the captured signals. A clustering approach—where the number of clusters (i.e., sound sources) is also an unknown parameter—is then employed to determine the number of sources and their locations.

Similarly, Diep et al. (2013) [3] in their work Acoustic Counting and Monitoring of Shad Fish Populations propose a method for analyzing acoustic signals during fish spawning. This method relies on short-term spectral analysis combined with a Gaussian mixture model for detecting and counting shad spawning acts. In our study, spectrum analysis method is also used, but unlike their work, the purpose of the method is to determine the ball numbers by detecting the sound of collisions. Additionally, only a single microphone was utilized, as opposed to analyzing data from multiple microphones.

Third, the study by Amft et al. (2009) [4] titled Bite Weight Prediction from Acoustic Recognition of Chewing explores predicting the weight of food consumed by analyzing the sound of chewing. While their work is similar to ours in that both employ an acoustic method to estimate the quantity of an item in a confined space, there are notable differences. In their case, the sound is produced by compressing food, causing it to fracture and emit sound as a means of energy release. In contrast, our study focuses on sounds generated solely by collisions.

Despite these differences, the methodologies share a common goal of using sound to measure quantities in confined environments, highlighting the potential of acoustic analysis in various fields.

This paper presents a series of novel and practical methods for non-invasive object counting using acoustic analysis. We focus on the specific problem of estimating the ball numbers within a black box. Our approach combines theoretical modeling based on energy methods with a data-driven approach utilizing a Convolutional Neural Network (CNN). A multi-modal physics simulation model was developed using python to predict the sound energy produced by the system.

The key research objectives of this study are:

- Construct the theoretical model of 3D collision cloud between balls in the box and between balls and the wall of the box.
- Develop a multi-modal physics simulation model that can predict the sound energy produced by different numbers of balls within the black box.
- Design and train a CNN to accurately estimate the ball numbers based on the acoustic features extracted from the audio recordings.
- Utilize the signal energy formula to validate results from the theoretical model.
- Evaluate the effectiveness of our combined theoretical and data-driven approach for non-invasive object counting using acoustic analysis.

By successfully addressing these objectives, this paper aims to demonstrate the potential of the approach for various real-world applications, paving the way for more efficient, cost-effective, and non-invasive object counting solutions [5].

II. EXPERIMENTAL SYSTEM SETUP

A. Mecahnical system design

As shown in Figure 1, the experimental setup comprises a mechanical system designed to hold and release a box filled with spherical balls. The primary components include a metallic platform with a perforated surface that provides a stable base for the experiment. Attached to this platform is a hinged arm mechanism, which controls the tilt and movement of the box.

The green components are made using a 3D printer to secure the boxes and hinges. In order to facilitate theoretical analysis and simulation calculation, the sealed container is designed as a cube structure. The box is customized based on transparent acrylic material, allowing for easy and intuitive qualitative observation of the ball movement inside the box.

A stepper motor is used to drive the system. It is controlled by a Raspberry Pi outputting PWM signals to achieve higher control accuracy than regular motors as shown in Figure 2.

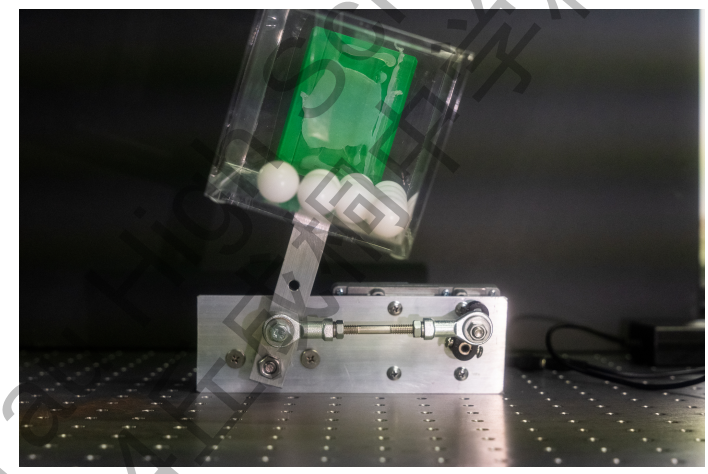

Fig. 1. Experimental setup front.

During each experimental run, the box holds a varying ball numbers. By adjusting the angle of the hinged arm, the box can be tilted, causing the balls to move and collide with the internal surfaces, thereby producing sound. The speed of rotation has been calculated in detail in the theoretical section.

This setup collects acoustic data as the balls impact the box's internal surfaces. The mechanical design ensures consistent and reproducible ball movements, which is critical for accurate sound data collection.

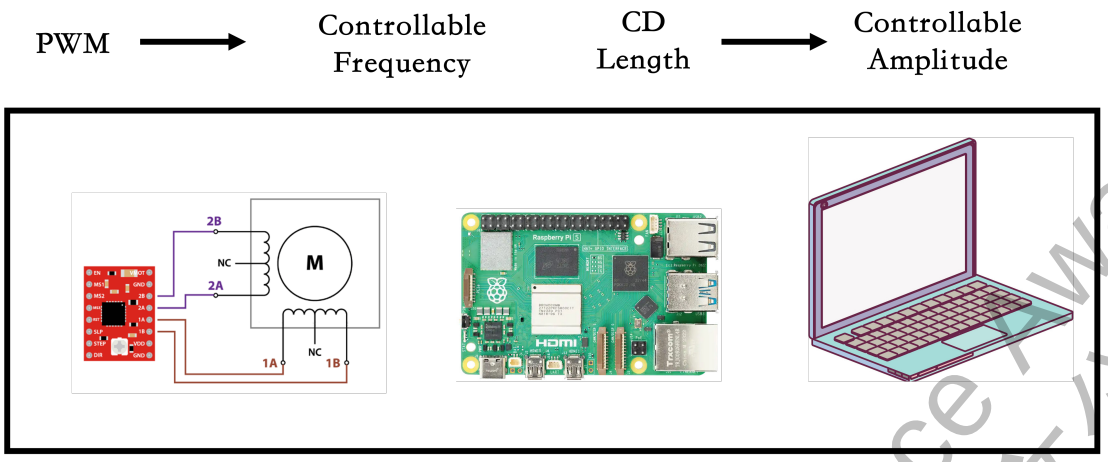

Fig. 2. Experimental setup

B. *Illustration of experimental system*

To illustrate the machine's operation more clearly, a schematic is presented in Figure 3. Let the horizontal direction be the X-axis and the vertical direction be the Z-axis.

Let the coordinates of point C at time t be $(x_C(t), z_C(t))$. The coordinates of point D, $(x_D(t), z_D(t))$, correspond to $(x_C(t), z_C(t))$.

Point C rotates in a circular path with a frequency of 2 Hz and a radius r, while the rod AD (with length R) rotates around point A. The rod CD is rigid, with a fixed length L.

Given the coordinates of point A as (x_A, z_A) and point B as (x_B, z_B), the following system of equations is established to describe the relationships between these points:

$$(x_C(t) - x_B)^2 + (z_C(t) - z_B)^2 = r^2, \tag{1}$$

$$(x_D(t) - x_A)^2 + (z_D(t) - z_A)^2 = R^2, \tag{2}$$

$$(x_D(t) - x_C(t))^2 + (z_D(t) - z_C(t))^2 = L^2. \tag{3}$$

Since point C follows a circular path, its coordinates can be expressed as:

$$x_C(t) = r\cos(4\pi t), \tag{4}$$

$$z_C(t) = r\sin(4\pi t). \tag{5}$$

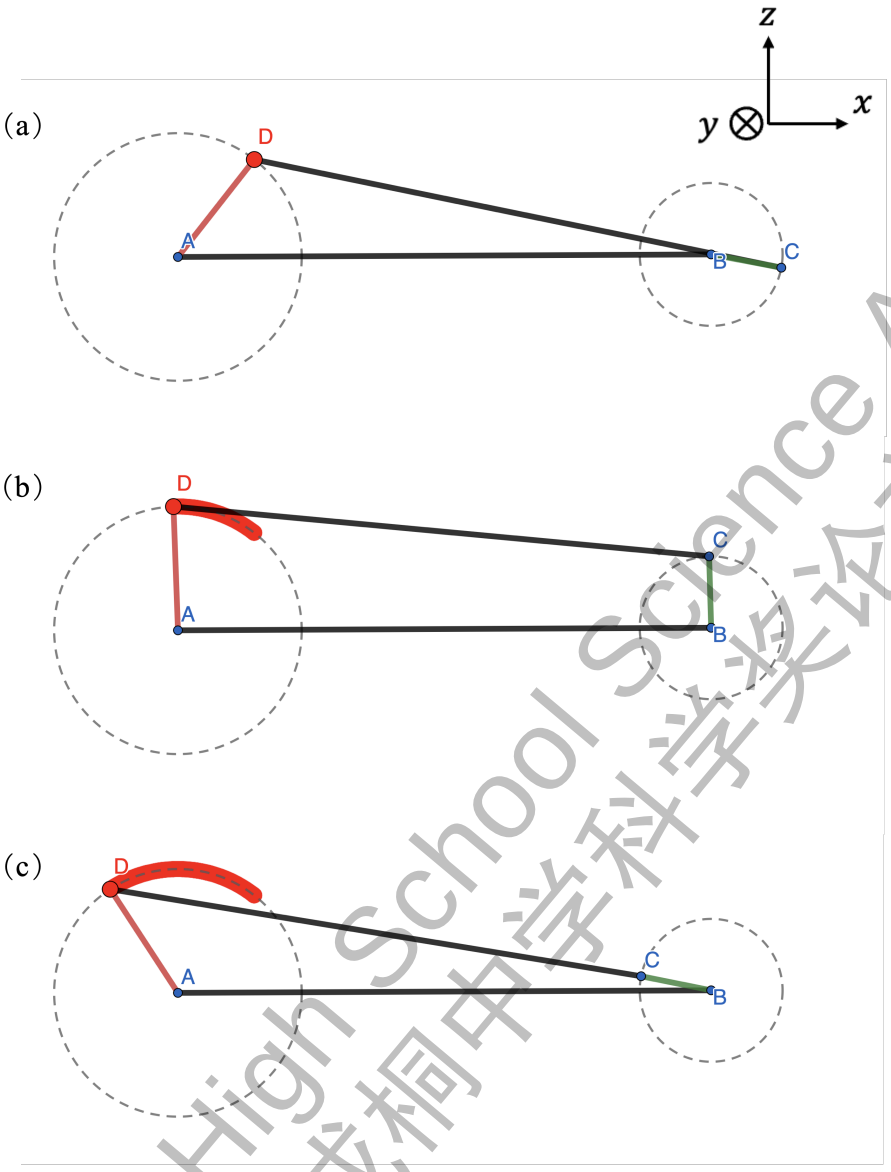

Fig. 3. Schematic of the mechanics in motion.

Subsequently, there is:

$$z_C(t) = z_B \pm \sqrt{z_B^2 - (z_B^2 - r^2 + (x_C(t) - x_B)^2)}, \tag{6}$$

$$z_D(t) = y_A \pm \sqrt{z_A^2 - (z_A^2 - R^2 + (x_C(t) - x_A)^2)}. \tag{7}$$

To find the accelerations a_x and a_z of point D with respect to point C, the second derivatives of $x_D(t)$ and $z_D(t)$ is needed. First, these equations were differentiated with respect to time to obtain the

velocities of point D, and then differentiate again to find the accelerations.

By substituting the expressions for $x_C(t)$ and $z_C(t)$, the motion of point D can be fully described. The box's acceleration is calculated by finding the second derivatives of $x_D(t)$ and $z_D(t)$.

This experimental setup provides a method to collect acoustic data as the balls collide with the internal surfaces of the box.

C. Acoustic data preparation

*1) **Data collection**:*

For acoustic data collection, the experiment was repeated multiple times with varying numbers of balls placed inside the box. Each time, the box was shaken using our mechanism to induce motion of the balls, and the resulting sounds were recorded.

The ball numbers was systematically altered between runs to observe the effect on the acoustic signal. The collected sound data from these experiments were then used for further analysis, particularly in the noise reduction process described in subsequent sections.

Thus data of different ball numbers ranging from 0-10, 15, 20, and 30 balls were collected. Data of the ball numbers were each collected for 20 times. The subsequent files are named as i-j where i is the ball numbers in the box, j is the ball number ranging from 1 to 20.

*2) **Data preprocessing**:*

The first 20 rounds of noise data were collected by running the machine without placing any balls in the box. After acquiring the noise data, the mean noise energy and spectrum is calculated.

Next, spectral subtraction is applied to remove the noise from the data collected with balls in the box. The basic principle of spectral subtraction is to subtract the estimated noise spectrum from the spectrum of the noisy signal, thereby resulting in a relatively clean signal.

III. THEORETICAL ANALYSIS

In this paper, the theoretical analysis is carried out from three aspects. Firstly, a dynamic multi-modal physics model of ball collision and sound pressure in a box is established. Secondly, classical acoustic feature extraction methods such as Fourier transform are used to convert the time domain sound data into frequency domain data, and a practical method based on energy estimation over time domain and frequency domain is proposed to distinguish the ball numbers. Finally, a CNN model is proposed for training based on various acoustic analysis methods to achieve a quantitative estimation of the ball number.

To better demonstrate the mechanical aspect of the physics simulation model, a flow chart is drawn, as shown in Figure 4.

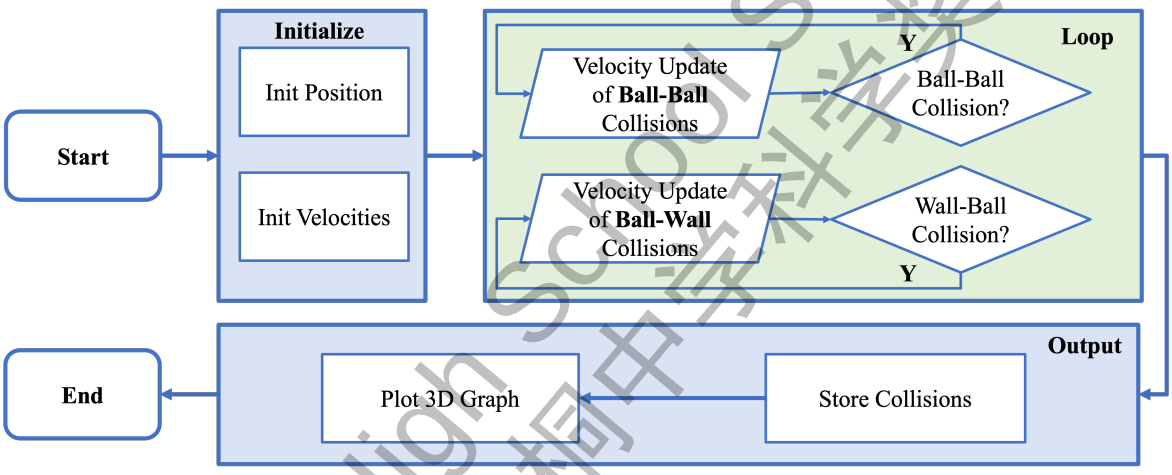

Fig. 4. This flowchart illustrates a collision simulation process.

The process begins with initialization, where the positions and velocities of the balls are set. Following this, it enters the collision update phase, where two types of collisions checked each time-step. First, the velocities are updated for ball-to-ball and ball-to-wall collisions if a collision is detected. After handling these collision updates, the process proceeds to the final stage, where a 3D graph is plotted to visualize the simulation, and collision data is stored. This marks the end of the simulation process.

A. Simulation based collision dynamics

It is very difficult to strictly derive the exactly positions of the collisions since they happen so frequently. So a simulation and probabilistic approach is employed to this problem. The physics equations

represented by the code describe the motion of balls in a 3D space under the influence of a time-varying acceleration field, including elastic collisions with walls and other balls. The key physics equations are:

1) ***Equations of Motion****:*

For each ball:

$$x(t+\Delta t) = x(t) + v_x(t)\Delta t + \frac{1}{2}a_x(t)\Delta t^2 \tag{8}$$

$$y(t+\Delta t) = y(t) + v_y(t)\Delta t + \frac{1}{2}a_y(t)\Delta t^2 \tag{9}$$

$$z(t+\Delta t) = z(t) + v_z(t)\Delta t + \frac{1}{2}a_z(t)\Delta t^2 \tag{10}$$

$$v_x(t+\Delta t) = v_x(t) + a_x(t)\Delta t \tag{11}$$

$$v_y(t+\Delta t) = v_y(t) + a_y(t)\Delta t \tag{12}$$

$$v_z(t+\Delta t) = v_z(t) + a_z(t)\Delta t \tag{13}$$

Where:

$$a_x(t) = (\frac{d^2 z_D(t)}{dt^2} + g)\cos\theta + \frac{d^2 x_D(t)}{dt^2}\sin\theta \tag{14}$$

$$a_y(t) = 0 \tag{15}$$

$$a_z(t) = (\frac{d^2 z_D(t)}{dt^2} + g)\sin\theta + \frac{d^2 x_D(t)}{dt^2}\cos\theta \tag{16}$$

a) ***Ball-Wall Collision****:*

When a ball collides with a wall, its velocity component perpendicular to the wall is reversed and scaled by the coefficient of restitution:

$$v'_\perp = -e_2 v_\perp \tag{17}$$

Where $v_\perp$ is the velocity component perpendicular to the wall and $e_2 = 0.95$.

b) ***Ball-Ball Collision****:*

For two colliding balls (1 and 2), the velocity changes are given by:

$$\Delta\vec{v}_1 = (v_{2n} - v_{1n})\hat{n}e_1 \tag{18}$$

$$\Delta\vec{v}_2 = (v_{1n} - v_{2n})\hat{n}e_1 \tag{19}$$

Where:

- v_{1n} and v_{2n} are the normal components of the velocities along the line of centers
- $\hat{n}$ is the unit vector along the line of centers
- $e_1 = 0.9$ is the coefficient of restitution for ball-ball collisions

*c) **Collision Detection**:*

- Wall collision occurs when:

$$|x| \geq \frac{L}{2} - r, \quad |y| \geq \frac{L}{2} - r, \quad \text{or} \quad |z| \geq \frac{L}{2} - r \tag{20}$$

where L is the box size and r is the ball radius.

- Ball-ball collision occurs when the distance between ball centers is $\leq 2r$:

$$|\vec{r}_2 - \vec{r}_1| \leq 2r \tag{21}$$

These equations collectively describe the dynamics of the balls, including their motion under acceleration, elastic collisions with walls and other balls, and the criteria for recording significant collision events in the simulation.

*2) **Refined detection parameters**:*

Python is used to simulate this system, however, normal code fails to differentiate between nearby collisions, resulting in a 'spaghetti string' of collisions which are physically impossible.

To address the issue of excessive and unnecessary collision recordings in a ball-wall collision simulation, a refined approach that incorporates a speed threshold for collision detection is introduced. This method ensures that only significant collisions, based on a predefined relative speed, are recorded and processed. Below, the modifications made to the existing simulation code to implement this feature are outlined.

*a) **Relative Speed for Ball Collisions**:*

$$v_{rel} = |v_{2n} - v_{1n}| \tag{22}$$

*b) **Minimum Speed Threshold for Recording Collisions**:*

$$|\vec{v}| > 0.1 \tag{23}$$

*3) **Simulation constants and parameters**:*

The simulation is defined with the following constants:

- Coefficient of restitution for ball-ball collisions (e_1): 0.9
- Coefficient of restitution for ball-wall collisions (e_2): 0.95
- Radius of each ball (r): 1.0
- Side length of the cubic box (l): 10.0
- Acceleration field components (a_1, a_2, a_3): (0.0, 0.0, -9.81), representing gravity acting in the z-direction
- ball numbers (`NUM_BALLS`): 10
- Simulation time (`simulation_time`): 100.0 seconds

A list named `collision_points` is used to store details of each collision, including the position (x, y, z), type of collision (ball or wall), and the relative speed at the point of collision.

*4) **Collision detection functions**:*

In the simulation system, 2 functions are designed, one is Update Positions and Velocities and the other is Check Wall Collision.

The function `update_positions_and_velocities` iteratively updates the velocity and position of each ball based on the acceleration and checks for collisions with walls and other balls. Wall collisions are processed to reflect off the walls with a restitution factor, and ball collisions are handled using simple elastic collision physics.

The function `check_wall_collision` is specifically designed to handle wall collisions more accurately:

- **Position and Velocity**: It retrieves the current position and velocity along the specified axis (x, y, or z).
- **Collision Detection**: It checks if the ball's position exceeds the boundaries of the box, considering the ball's radius.
- **Speed Threshold**: A minimum speed of 0.1 units is set to filter out insignificant collisions. Only collisions where the relative speed exceeds this threshold are recorded.
- **Collision Recording**: If the collision is significant, the ball's position and velocity are corrected, and the collision details are logged.

The simulation initializes a set of balls with random positions and velocities and simulates their motion over the specified time. Collisions are recorded and processed as described.

After the simulation, the collision points are plotted in a 3D scatter plot using Matplotlib, with different colors representing wall and ball collisions. The relative speed of ball collisions is visualized using a color gradient. Additionally, collision details are saved to a text file for further analysis.

This refined approach to collision detection in simulations not only enhances the accuracy of the simulation by preventing the recording of trivial collisions but also provides a clearer visualization of the dynamics involved. By implementing a speed threshold, the simulation focuses on significant interactions, thereby improving both the performance and the utility of the simulation results.

*5) **3D Collision cloud**:*

A 3D Graph of collisions within the box offer a visual representation of the potential interactions between rigid balls.

Below are the 3D Graphs of boxes with 2, 5 and 10 balls, shown in Figure 5, Figure 6 and Figure 7.

It can be derived from the graphs that the larger the ball number, the more collisions will be observed during a given time period. This can also further supported through energy analysis and acoustic data collected from experiments as shown further down in the paper.

In the end, a relationship between ball numbers and the number of ball-ball collisions and ball-wall collisions can be demonstrated in Figure 8.

Through stored collisions, collision count as well as sound energy can be acquired. As $E_{collision} = \int udV$, where $u = (\frac{1}{2}\rho)p(x,t)^2$, and $E_{total} = \sum_{i=1}^{n} E_i$ total energy was calculated through simulation of pressure fields in Figure 9.

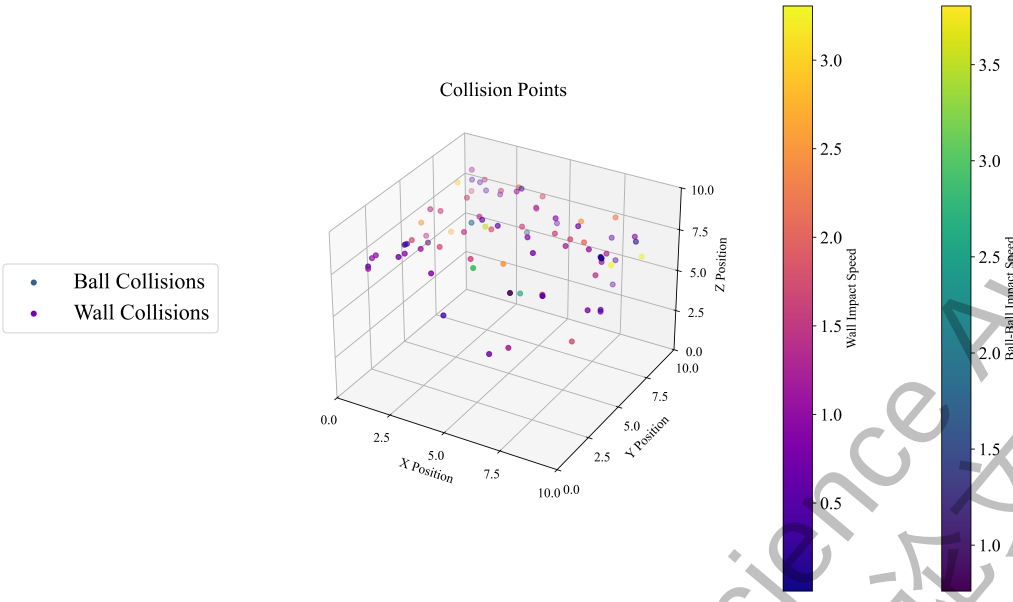

Fig. 5. Collision clound of 2 balls

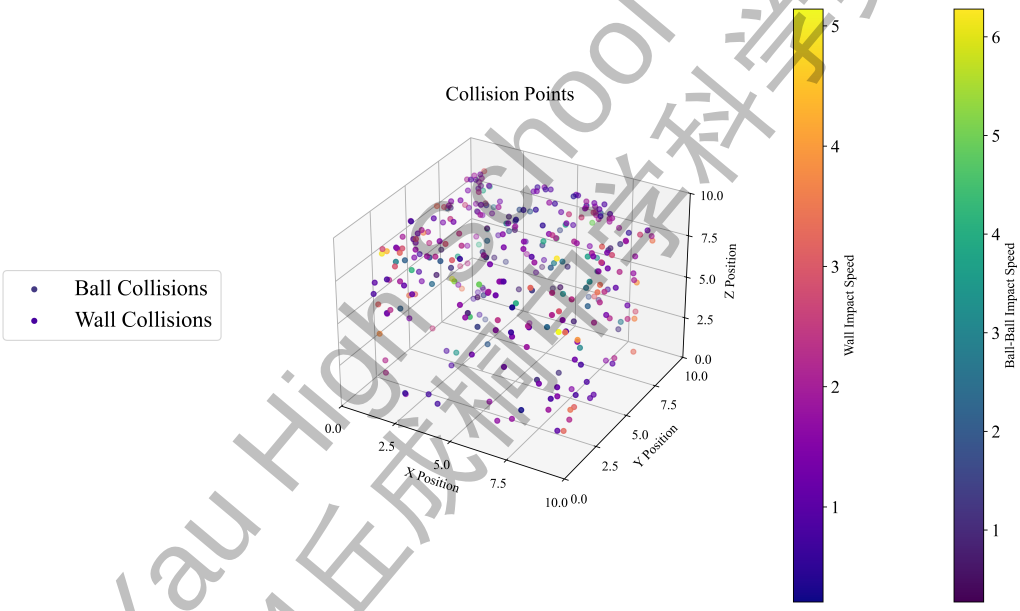

Fig. 6. Collision clound of 5 balls

B. *Traditional acoustic feature analysis*

Initially, the input time domain sound signal $x(t)$ is transformed into the frequency domain using the Short-Time Fourier Transform (STFT) to obtain the spectrum for each frame of the signal:

$$X(\omega, t) = \text{STFT}\{x(t)\} \tag{24}$$

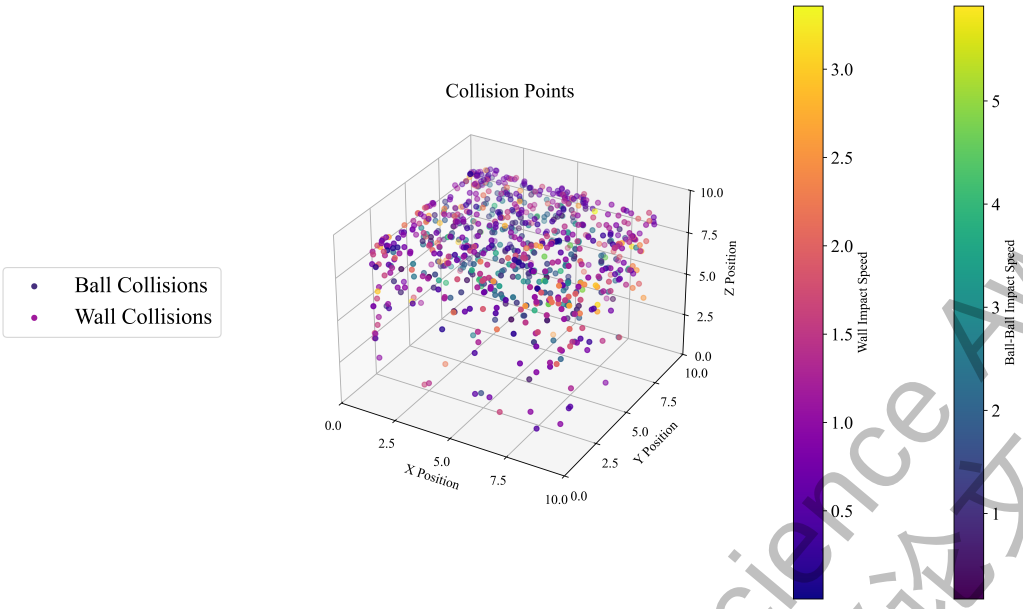

Fig. 7. Collision clound of 10 balls

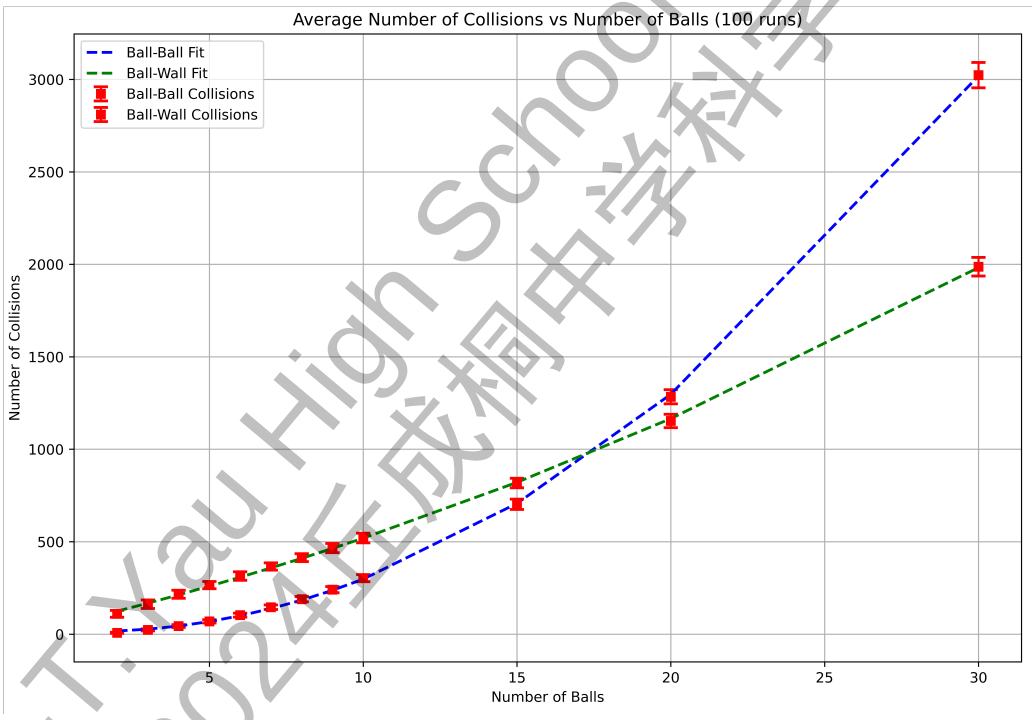

Fig. 8. Relationship between ball numbers and the number of ball-ball collisions and ball-wall collisions.

where $X(\omega, t)$ represents the frequency domain signal at time t and frequency ω [6].

The noise spectrum $N(\omega, t)$ can be estimated by analyzing segments of pure noise or using statistical methods [7]. The average noise spectrum, assumed to be stationary, is shown in Figure 11.

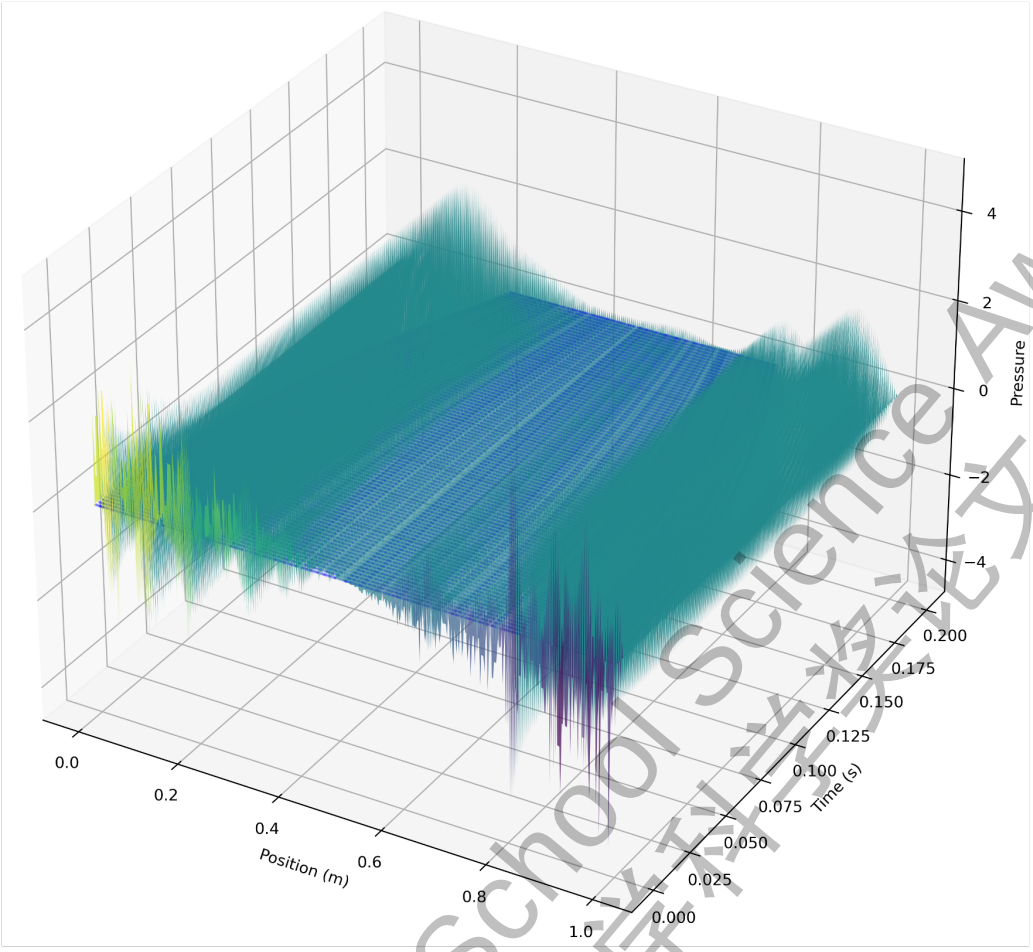

Fig. 9. Relationship between pressure, collision position and time.

$$\hat{N}(\omega, t) = \text{Estimate of } N(\omega, t) \tag{25}$$

*1) **Dynamic noise estimation**:*

To enhance the noise estimation process, a dynamic noise estimation technique that adapts to variations in the noise environment is employed. The formula used for noise estimation updates the noise spectrum based on the observed signal.

The noise spectrum $N(\omega, t)$ at a given time t can be dynamically estimated by averaging the noise over several frames where the signal is assumed to be inactive [8]. The updated noise estimate $\hat{N}(\omega, t)$ is calculated as:

$$\hat{N}(\omega, t) = \alpha \cdot \hat{N}(\omega, t-1) + (1-\alpha) \cdot |X(\omega, t)|^2 \tag{26}$$

where:

- $\hat{N}(\omega, t)$ is the estimated noise spectrum at time t.
- α is a smoothing factor, typically between 0.9 and 1, which determines the weight given to past noise estimates.
- $|X(\omega, t)|^2$ is the power spectrum of the noisy signal at time t.

This formula effectively averages the noise spectrum over time, assuming that noise characteristics change slowly. By continuously updating $\hat{N}(\omega, t)$, the noise estimate adapts to slow variations in the noise environment, providing a more accurate basis for spectral subtraction.

This adaptive noise estimation approach is particularly beneficial in scenarios where the noise is not perfectly stationary, ensuring that the noise subtraction process remains effective even when noise characteristics evolve over time.

The basic formula for spectral subtraction is given by:

$$\hat{S}(\omega, t) = |X(\omega, t)| - |\hat{N}(\omega, t)| \tag{27}$$

where $\hat{S}(\omega, t)$ represents the estimated clean signal spectrum, $|X(\omega, t)|$ denotes the magnitude spectrum of the noisy signal, and $|\hat{N}(\omega, t)|$ is the magnitude spectrum of the estimated noise [9].

*2) **Analysis using Fourier Transform**:*

In this analysis, acoustic data recorded from various experimental setups is processed and examined. The primary objective was to extract meaningful features from the sound data, with a particular focus on the frequency domain characteristics. Additionally, polynomial fitting was applied to the smoothed frequency spectra to better understand the underlying trends.

For each category i,all corresponding audio files are labeled as $i-j$, where j ranges from 1 to 20. If necessary, the audio data is converted to a single channel (mono) by averaging the stereo channels. One spectrum diagram out of our m4a files is shown in Figure 10.

There is also average noise spectrum in Figure 11.

Using spectrum subtraction method, the clean spectrum was acquired in Figure 12.

To derive a representative signal for each category, the waveforms and their corresponding Fourier transforms are averaged across all 20 files within each category. Due to variations in the lengths of the

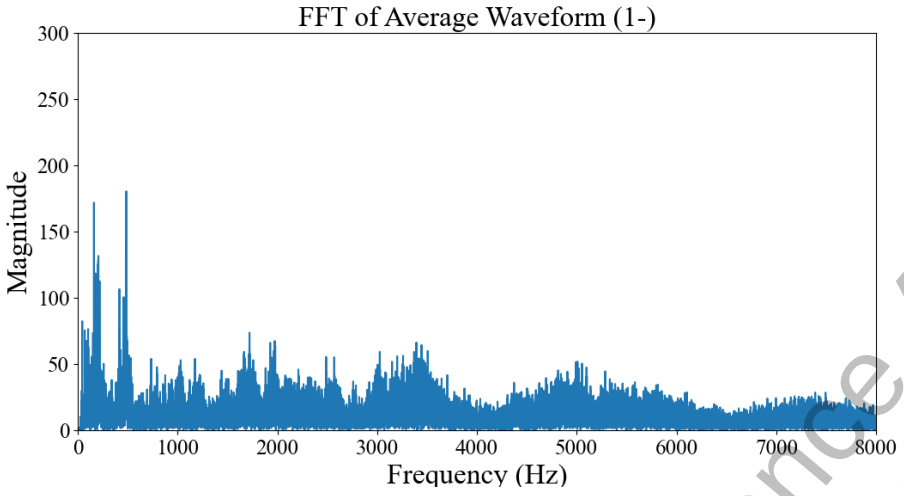

Fig. 10. This is the spectrum when the box have one ball in it.

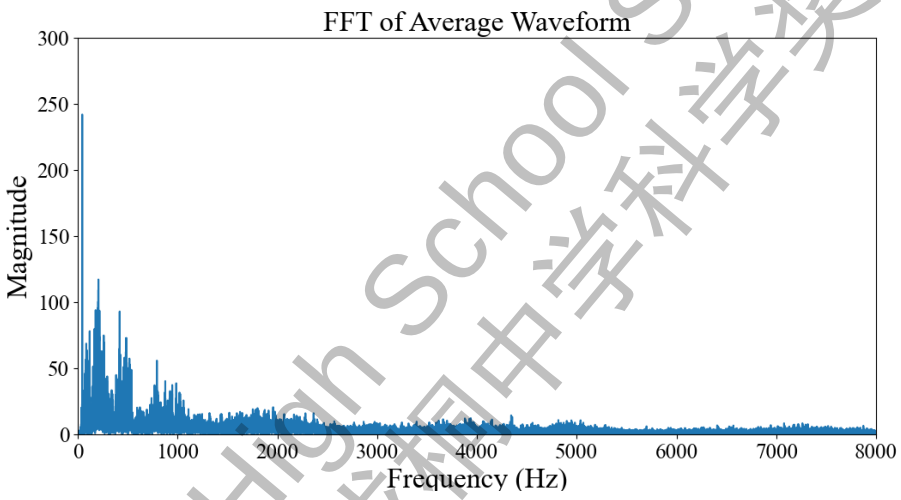

Fig. 11. This is spectrum for average noise

recordings, the data is aligned by trimming all waveforms and spectra to match the length of the shortest file in each category.

3) ***Energy estimation over frequency spectrum****:*

The averaged Fourier spectrum for each category was further processed to obtain a smooth envelope that would more clearly reveal the underlying frequency trends. This was achieved through the iterative application of a moving average filter. Specifically, a moving average filter with a window size of 50 was applied to the averaged Fourier spectrum, and the smoothing process was repeated 50 times to enhance the smoothness of the spectrum.

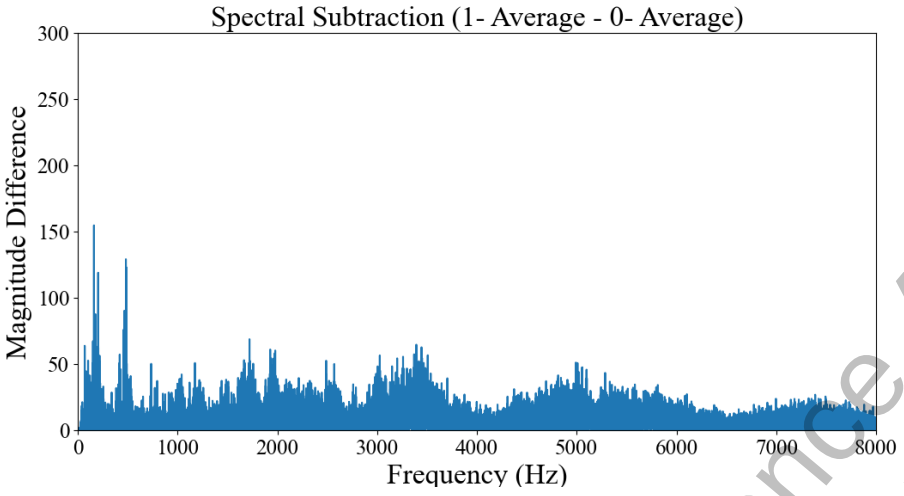

Fig. 12. This is clean spectrum.

After smoothing the frequency spectrum, polynomial fitting was applied to the data. A third-order polynomial was chosen to fit the smoothed spectra, providing a compact representation of the frequency characteristics. The resulting polynomial coefficients were used to generate a formula representing the spectral envelope for each category. The general form of the polynomial fit is given by:

$$P(x) = c_3x^3 + c_2x^2 + c_1x + c_0 \tag{28}$$

where $P(x)$ is the polynomial fitted to the frequency spectrum, and c_3, c_2, c_1, and c_0 are the coefficients determined by the fitting process.

As shown in the Figure 13, there is a noticeable increase in the magnitude of the frequency as the ball numbers increases. This trend underscores the relationship between the ball numbers in the experimental setup and the resulting frequency characteristics of the sound produced.

Thus, the number of different balls can be distinguished by calculating the area below the spectral envelope over the frequency domain. On the other hand, the energy produced by the impact of different numbers of balls should also be different, although it is not easy to see intuitively. Based on the above inferences, a new method was proposed to distinguish the number of pellets in frequency domain energy estimation.

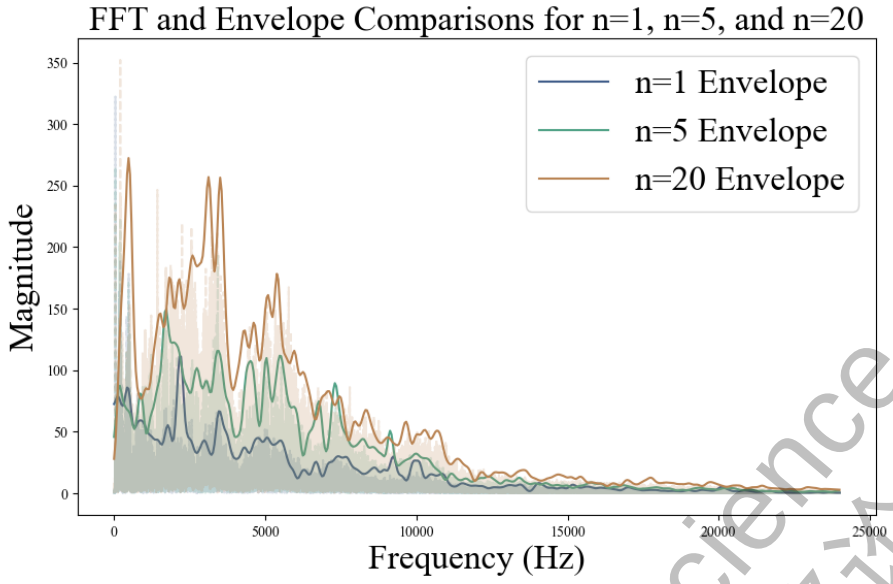

Fig. 13. Smoothed frequency spectrum envelopes for when ball number is 1, 10 and 20.

C. *Prediction using machine learning method*

Although there is a difference between different numbers of balls through their energy as well as the area of maximum envelope area of the Fourier spectrum plot, the accuracy is still under doubts. Thus apply CNN machine learning method was applied to increase accuracy. The sturcture of CNN model is shown in Figure 14.

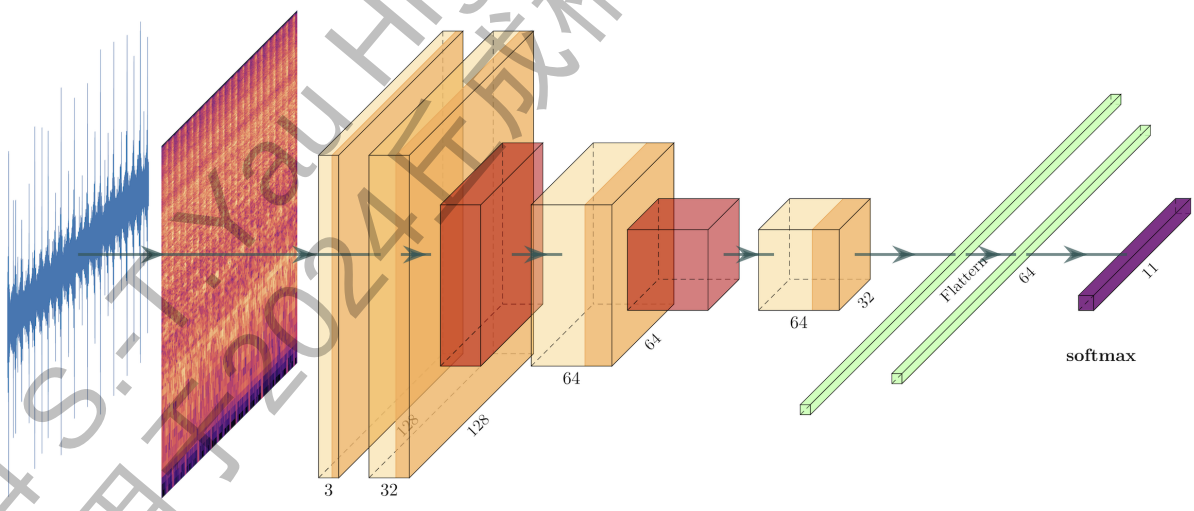

Fig. 14. CNN machine learning model.

*1) **Input data processing**:*

To use a CNN machine learning model, our audio data was first converted into visual representations. Mel Frequency Cepstral Coefficients (MFCC), constant-Q transform (CQT), spectrograms, and wavelet scalograms were utilized to effectively classify the clean data.

*a) **Waveform**:*

Waveform representations are another critical tool in audio signal processing. A waveform provides a visual representation of an audio signal in the time domain, showing how the amplitude of the signal varies over time.

The process of generating a waveform begins by loading an audio file, such as a `.m4a` file, using Python libraries like `librosa`. The function `librosa.load` reads the audio file and returns both the audio time series data, which is a one-dimensional array, and the sample rate, which defines the number of samples of audio carried per second.

Once the audio signal is loaded, the waveform can be plotted directly using `matplotlib`. The time axis is generated by dividing the total number of samples by the sample rate, giving the time in seconds corresponding to each point in the time series data. The amplitude of the waveform at each time point represents the intensity of the audio signal at that instant shown in Figure 10.

To improve the visualization, it is common to normalize the waveform by scaling the amplitude values between -1 and 1. This allows for a clearer display of the relative changes in the signal's intensity without distortion from large values. The waveform is then plotted, with time on the x-axis and amplitude on the y-axis.

The waveform provides important insights into the temporal structure of the audio signal. For example, in speech analysis, distinct speech sounds like vowels and consonants can be identified visually based on the signal's amplitude variation over time. Similarly, in music, waveforms reveal patterns corresponding to notes, chords, and rhythmic structures.

However, waveforms do not provide much information about the frequency content of the signal, making them less useful for tasks that require understanding the signal's spectral characteristics. For this reason, waveforms are often used in combination with other representations, such as the spectrogram or MFCC, which provide detailed information about the frequency components of the signal.

*b) **MFCC**:*

Mel Frequency Cepstral Coefficients (MFCC) are widely used in audio signal processing, particularly for speech and music analysis, as they provide a compact representation of the power spectrum of an audio signal. The transformation process involves several key steps.

The first step is to apply a pre-emphasis filter to the audio signal, which emphasizes higher frequencies [10]. This filter is applied using the following formula:

$$y(t) = x(t) - \alpha x(t-1) \tag{29}$$

where $y(t)$ is the output signal, $x(t)$ is the input signal, and α is a coefficient, typically around 0.97.

Next, the audio signal is divided into short frames, typically 20–40 ms in length, since speech and other audio signals are non-stationary. A window function, such as the Hamming window, is applied to each frame to reduce spectral leakage [11]. The Hamming window is defined as:

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \tag{30}$$

where N is the length of the window and n is the sample index.

After applying the window, the Fast Fourier Transform (FFT) is performed on each frame to convert the signal from the time domain to the frequency domain [12]. The power spectrum is then computed as the square of the magnitude of the FFT:

$$P(f) = |X(f)|^2 \tag{31}$$

where $X(f)$ is the Fourier-transformed signal.

The power spectrum is mapped onto the Mel scale, which approximates how humans perceive sound frequencies [13]. The Mel scale is defined as:

$$M(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{32}$$

A filterbank of triangular filters is then applied to the power spectrum, with each filter emphasizing a different range of frequencies on the Mel scale. The output is the log energy of each filter's response.

Finally, the log energies are transformed using the Discrete Cosine Transform (DCT) to obtain the MFCC. The DCT decorrelates the filterbank energies, and the resulting coefficients represent the cepstral features of the signal [14]. The n-th MFCC is computed as:

$$c_n = \sum_{k=1}^{K} \log(E_k) \cos\left(\frac{n(k-0.5)\pi}{K}\right) \tag{33}$$

where E_k is the log energy of the k-th Mel filter, and K is the total number of filters.

In this process, `.m4a` audio files are converted into spectrogram images by applying the Short-Time Fourier Transform (STFT) and saving the resulting data in PNG format. First, the audio files are loaded using `librosa`, a Python library for analyzing music and audio. The function `librosa.load` reads the `.m4a` file and returns the audio time series data and the sample rate. The STFT is then computed using `librosa.stft`, which divides the audio signal into overlapping windows and applies the Fourier Transform to each window, producing a frequency-time representation of the audio.

The magnitude of the STFT is calculated and converted to a decibel (dB) scale using `librosa.amplitude_to_db`, which allows for better visualization. This transformation provides a clearer representation of the signal's frequency content over time. A spectrogram is generated by plotting the decibel-scaled STFT using `librosa.display.specshow`. For a cleaner visual representation, axis labels and ticks are removed.

Finally, the generated spectrogram is saved as a PNG image of the specified size using `matplotlib`, and the process is repeated for all `.m4a` files in the input folder. Each spectrogram is saved in the output folder, allowing for efficient batch processing of multiple audio files. This method provides a compact visual representation of the frequency content in the audio files, useful for audio analysis and machine learning tasks.

The MFCC spectrogram provides a visual representation of the cepstral features over time. Each coefficient in the MFCC captures different aspects of the signal's spectral envelope, with the lower coefficients representing the coarse spectral shape and the higher coefficients capturing finer details. The MFCC spectrogram is widely used in speech recognition, speaker identification, and music analysis, as it captures important features that are invariant to changes in pitch, volume, and noise.

By visualizing the MFCC spectrogram, one can observe how the cepstral coefficients vary over time,

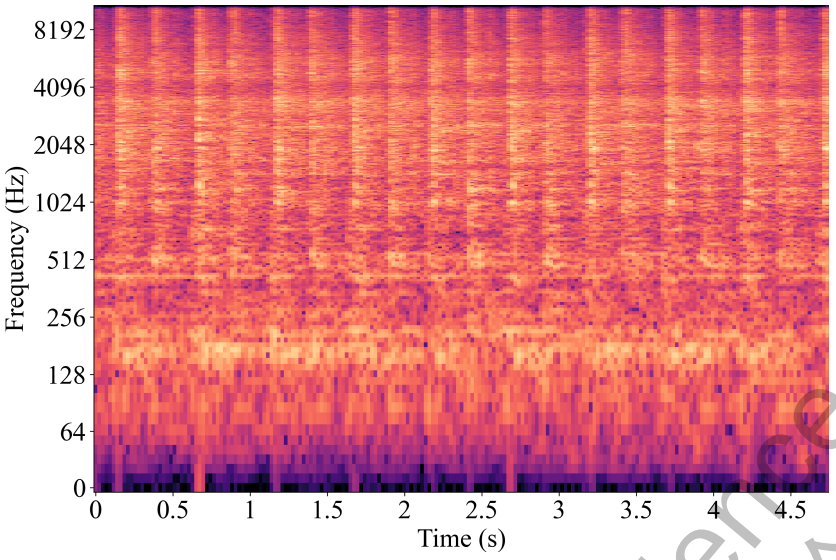

Fig. 15. MFCC diagram for one ball in the box; this is the 10th data.

offering insights into the structure and characteristics of the audio signal, which can be used for machine learning tasks and audio classification.

c) ***CQT****:*

The Constant-Q Transform (CQT) is another widely used technique in audio signal processing, particularly in music analysis [15]. Unlike the Short-Time Fourier Transform (STFT), which uses a fixed window size and frequency resolution, the CQT provides a logarithmic frequency resolution [16].

The CQT represents the frequency content of a signal using a series of filters with geometrically spaced center frequencies. The filters are spaced such that each filter's center frequency is a constant ratio, Q, of its bandwidth [17]. The transform is computed as:

$$X(k) = \sum_{n=0}^{N-1} x(n) \frac{1}{N_k} e^{-j2\pi \frac{kn}{N_k}} w(n) \tag{34}$$

where $X(k)$ is the transformed signal, N_k is the length of the window for the k-th frequency bin, and $w(n)$ is the window function. The window length varies for different frequency bins, providing better resolution at lower frequencies.

In practice, the CQT can be computed using the `librosa.cqt` function, which takes the audio signal as input and returns the CQT representation. The resulting CQT spectrogram can be visualized similarly to the STFT-based spectrogram by converting the amplitude to a decibel scale using

`librosa.amplitude_to_db`. This allows for better visualization of the signal's frequency content.

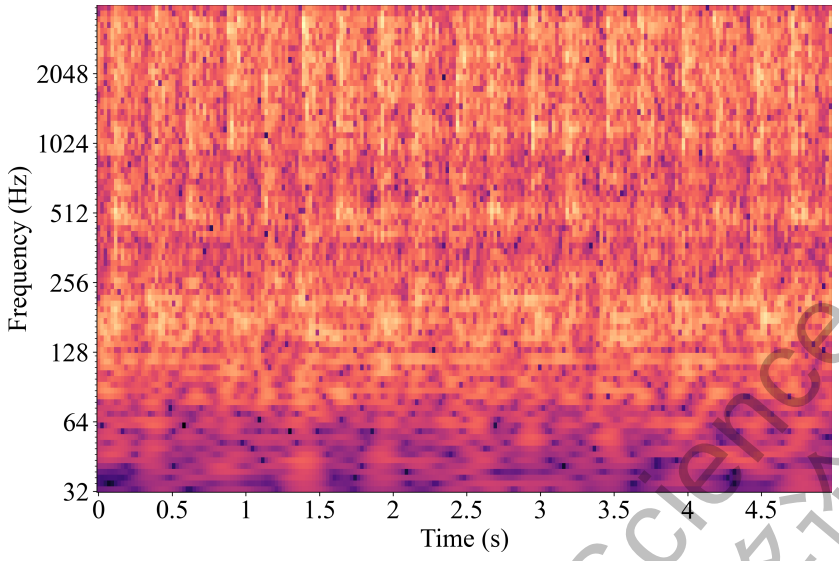

Fig. 16. CQT diagram for one ball in the box; this is the 10th data.

d) ***Spectrum****:*

The Short-Time Fourier Transform (STFT) provides a time-frequency representation of the signal by applying the Fourier Transform to overlapping short segments of the signal [18]. This allows for tracking how the signal's frequency content evolves over time, providing a detailed spectrum of the signal. The STFT is computed as:

$$STFT\{x[n]\}(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{35}$$

where $x[n]$ is the input signal, $w[n]$ is a window function (such as the Hamming window), and ω is the angular frequency. The STFT is applied by shifting the window across the signal and applying the Fourier Transform to each windowed segment. The result is a two-dimensional representation, where one axis represents time and the other represents frequency, as illustrated in Figure 17.

In practice, the STFT can be computed using libraries such as `librosa` or `scipy`. The magnitude of the STFT is often converted to a decibel scale for better visualization. The resulting spectrogram can highlight how the energy of different frequency components changes over time.

The STFT spectrogram is useful for analyzing non-stationary signals, like speech and music, as it captures both time and frequency information. It can be used for applications such as speech recognition,

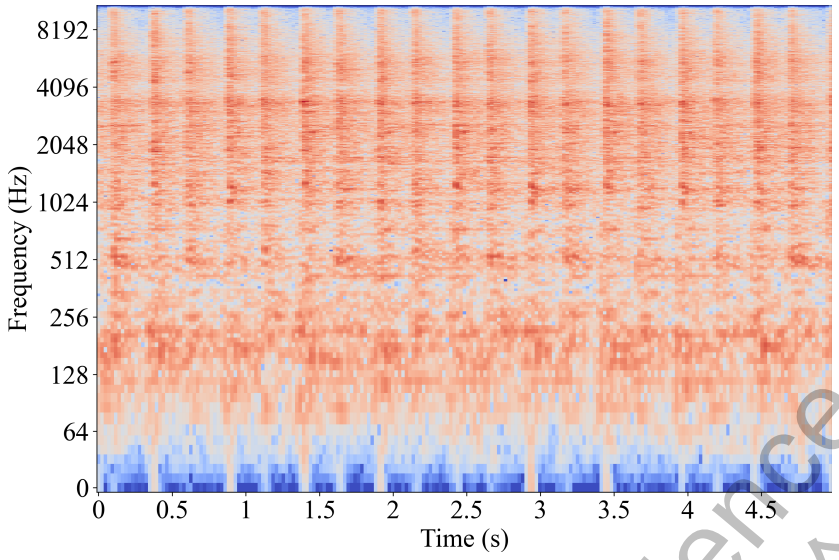

Fig. 17. STFT spectrum diagram for one ball in the box; this is the first data.

music transcription, and more.

*e) **Wavelet Scalograms**:*

The Wavelet Transform provides an alternative time-frequency representation by decomposing a signal into a set of wavelets, which are localized in both time and frequency [19]. This makes it particularly suitable for analyzing signals with transient features.

The Continuous Wavelet Transform (CWT) is given by:

$$CWT_x(a,b) = \int_{-\infty}^{\infty} x(t)\psi^*\left(\frac{t-b}{a}\right)dt \tag{36}$$

where ψ is the wavelet function, a is the scale parameter (related to frequency), and b is the time shift. The result is a two-dimensional scalogram, where one axis represents time and the other represents scale (or frequency) [20].

Wavelet transforms can be performed using Python libraries such as `PyWavelets` or `scipy`. The scalogram offers a more flexible representation compared to the STFT, making it ideal for analyzing signals with rapidly changing features.

The wavelet scalogram like in Figure 18 can provide insights into the time-varying characteristics of a signal, especially for signals with sharp transitions or localized events.

*2) **CNN model design**:*

Convolutional Neural Networks (CNN) are a class of deep neural networks primarily used for image

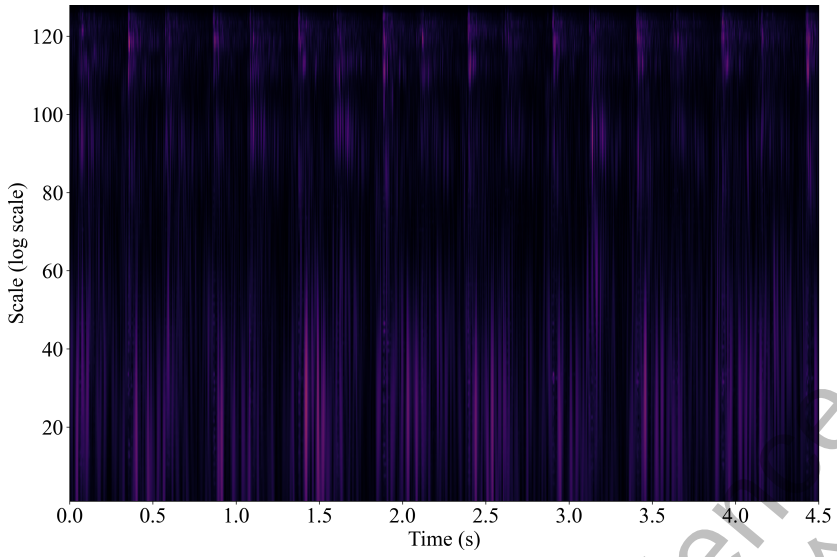

Fig. 18. Wavelet scalogram for one ball in the box; this is the first data.

recognition and classification tasks. The key advantage of CNN over traditional neural networks is their ability to capture spatial hierarchies in images through the use of convolutional layers. The following section outlines the CNN classification process used in our model, which is designed to classify images into one of 11 classes.

The data was first labelled by i-j, where i is the ball numbers in the box and j is the number of experiments. In CNN model, the data with the same i was classified into a group.

The input to the CNN is a wavelet-scalogram image of size 128×128 with three color channels (RGB). Each image is represented as a 3D tensor of shape $(128, 128, 3)$, where the first two dimensions represent the spatial size of the image, and the third dimension represents the color channels.

The CNN begins by applying a series of convolution operations to extract features from the input image. Each convolutional layer consists of a set of learnable filters (or kernels) that are convolved with the input image to produce feature maps. A filter is a small matrix (e.g., 3×3) that slides over the input image, computing dot products between the filter values and the corresponding image values.

Mathematically, the convolution operation can be expressed as:

$$(f * I)(x, y) = \sum_{i=1}^{k} \sum_{j=1}^{k} f(i, j) I(x + i, y + j) \tag{37}$$

where f is the filter of size $k \times k$, and $I(x, y)$ is the pixel value of the input image at position (x, y)

[21]. The result is a feature map that highlights specific features, such as edges or textures, learned by the filter.

After each convolution operation, a non-linear activation function, such as the Rectified Linear Unit (ReLU), is applied element-wise to the feature map [22]. The ReLU function is defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{38}$$

The ReLU function introduces non-linearity into the model, allowing it to learn more complex patterns in the data. It ensures that only positive feature values are passed to the next layer while setting negative values to zero.

Following the convolution and activation operations, a pooling layer is applied to reduce the spatial dimensions of the feature maps. The most common pooling operation is max pooling, which selects the maximum value within a small window (e.g., 2×2) of the feature map. This downsampling operation reduces the computational complexity and provides translation invariance to small shifts in the input image [23].

Mathematically, max pooling can be expressed as:

$$P(x, y) = \max\{I(x + i, y + j) \mid i, j \in [0, p - 1]\} \tag{39}$$

where $P(x, y)$ is the pooled feature map, and $p \times p$ is the size of the pooling window.

After several convolutional and pooling layers, the resulting feature maps are flattened into a 1D vector and passed to fully connected (dense) layers. In this stage, each neuron in the fully connected layer is connected to all neurons in the previous layer, allowing the network to learn non-linear combinations of the high-level features extracted by the convolutional layers [24].

The output of the fully connected layer is given by:

$$y_i = \sigma\left(\sum_j w_{ij} x_j + b_i\right) \tag{40}$$

where x_j are the inputs from the previous layer, w_{ij} are the learned weights, b_i is the bias term, and σ is an activation function (e.g., ReLU for hidden layers or softmax for the output layer).

The final layer in the CNN is a fully connected layer with 11 neurons, corresponding to the 11 classes (labels 0-10). A softmax activation function is applied to the output of this layer to produce class probabilities. The softmax function is defined as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{41}$$

where z_i is the input to the i-th neuron, and K is the total number of classes (in this case, 11) [25]. The softmax function ensures that the output is a probability distribution, with the sum of all probabilities equal to 1.

The class with the highest probability is selected as the predicted class for the input image.

The model is trained using the Sparse Categorical Cross-Entropy loss function, which is defined as:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \log(\hat{y}_{i,y_i}) \tag{42}$$

where y is the true label, $\hat{y}$ is the predicted probability for the correct class, and N is the number of samples in the batch. This loss function encourages the model to assign high probabilities to the correct class.

The model is optimized using the Adam optimizer, which adjusts the model parameters (weights and biases) to minimize the loss function by computing the gradient of the loss with respect to the parameters and updating them using backpropagation.

Accuracy is one of the primary metrics used to evaluate a neural network model during both training and testing phases.

Training accuracy represents the proportion of correctly classified instances in the training dataset. It is calculated at the end of each epoch and helps track the learning progress of the model. Formally, it is defined as:

$$\text{Training Accuracy} = \frac{N_{\text{Correct on Training}}}{N_{\text{Total on Training}}} \tag{43}$$

Validation accuracy is the accuracy of the model on the validation dataset, which typically consists of `test_images` and `test_labels`. It measures the model's performance on unseen data and is

commonly used to detect overfitting. The formula is:

$$\text{Validation Accuracy} = \frac{N_{\text{Correct on Validation}}}{N_{\text{Total of Validation}}} \tag{44}$$

Both training and validation accuracy are automatically computed during the training process by the `model.fit()` function, and the results are stored in `history.history['accuracy']` and `history.history['val_accuracy']` respectively.

In the provided code, a Custom loss metric is defined as:

$$\text{Custom loss} = \frac{n_{\text{predicted}} - n_{\text{true}}}{n_{\text{predicted}} + n_{\text{true}}} \tag{45}$$

Here, N stands for the number of predictions and n stands for the number of balls in the box. This Custom loss does not follow the conventional definition of classification accuracy. Instead, it reflects the relative difference between the predicted label and the true label, normalized by their sum.

If predicted_label = true_label, the Custom loss is zero, indicating a perfect match. As the discrepancy between predicted and true labels increases, the Custom loss deviates from zero, with the sign indicating the direction of the error. However, this metric is not a conventional measure of classification accuracy but rather a bias measurement.

Training loss quantifies the discrepancy between the predicted outputs and the true labels in the training dataset. It is typically computed using the cross-entropy loss function, which is defined as:

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{46}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability for the i-th training sample [26].

Validation loss is computed similarly to training loss but on the validation dataset. It measures how well the model generalizes to unseen data and is key for monitoring overfitting. Both training and validation loss values are automatically computed by the `model.fit()` function and stored in `history.history['loss']` and `history.history['val_loss']` respectively.

In summary, the standard accuracy and loss metrics serve as the main indicators of model performance during training, while the Custom loss provides additional insights into prediction biases, albeit with a

different interpretation than conventional classification accuracy.

*3) **Direct energy estimation**:*

The total energy E of a discrete signal $x[n]$ is calculated using the following formula:

$$E = \sum_{n=0}^{N-1} |x[n]|^2 \tag{47}$$

where $x[n]$ represents the amplitude of the signal at time n, and N is the total length of the signal [1]. This relationship is illustrated in Figure 19.

Figure 19 illustrates the relationship between the ball numbers and the average short-term energy, showing a clear positive correlation between the two variables. As the ball numbers increases, the average short-term energy rises consistently, suggesting a quadratic trend. This implies that experiment results matches well with theoretical calculation.

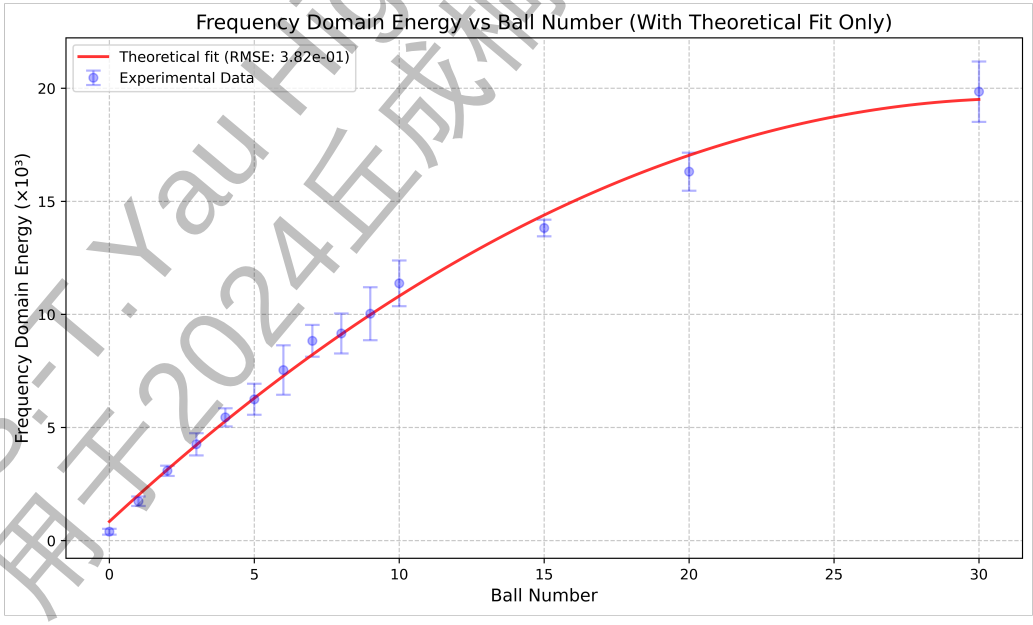

Fig. 19. Total energies of the ball numbers measured in experiment.

TABLE I
TEST SET SAMPLES: TRUE LABELS AND PREDICTED LABELS

True Label	Predicted Label	Custom loss
10	11	0.0476
2	2	0.0
1	1	0.0
7	8	0.0667
7	7	0.0
1	1	0.0
1	1	0.0
5	5	0.0
2	2	0.0
7	7	0.0
4	5	0.1111
10	10	0.0
2	2	0.0
5	5	0.0
6	7	0.0769
6	7	0.0769
7	8	0.0667
2	2	0.0
4	5	0.1111
7	7	0.0
1	1	0.0
8	8	0.0

Then, we utilize the theoretical formula conducted in the previous section $E = -0.01875125n^2 + 1.18463695n + 0.83882017$ to predict the number of balls. Since we have experimental results of energy, the inversion of the function provides an approach in predicting ball numbers. This method demonstrates an accuracy of **70.50%**, where detialed prediction is shown in the Table I.

D. Machine learning method results

1) Waveform:

In this section, we analyze the results of training a model directly on waveform data. Table II shows the true labels, predicted labels, and Custom loss for each sample in the test set. The Custom loss is calculated based on the relative difference between the predicted and true labels, as described in previous Formula 45. A positive value indicates that the predicted label is greater than the true label, while a negative value suggests the opposite.

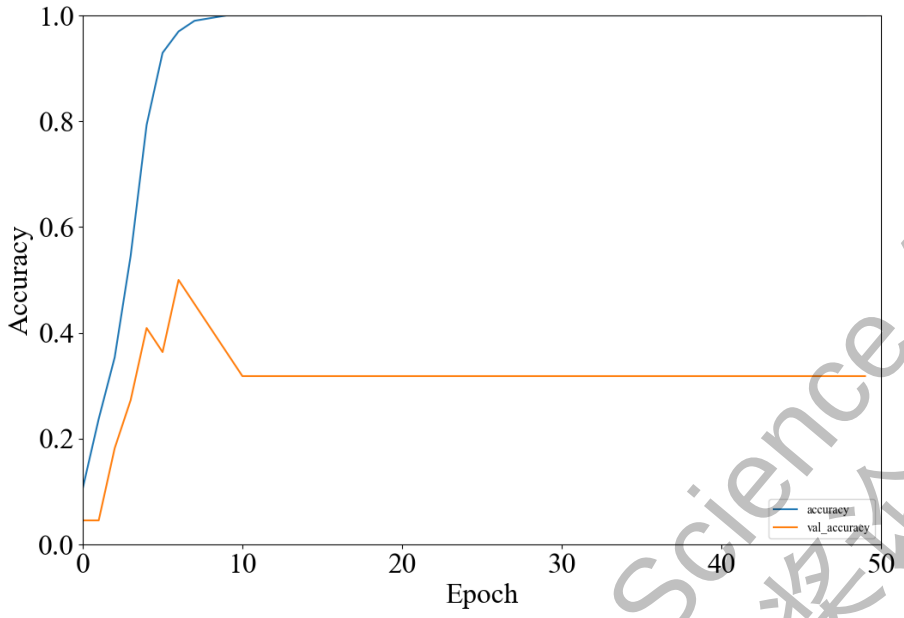

Fig. 20. Accuracy trends across training epochs for Waveform. Validation accuracy plateaued around 30.91%, suggesting signs of overfitting.

TABLE II
TEST SET SAMPLES: TRUE LABELS AND PREDICTED LABELS

True Label	Predicted Label	Custom loss
10	8	-0.1111
10	10	0.0
6	8	0.1429
8	9	0.0588
9	9	0.0
8	10	0.1111
9	5	-0.2857
1	1	0.0
10	9	-0.0526
10	10	0.0
7	5	-0.1667
7	5	-0.1667
3	4	0.1429
3	5	0.25
0	0	0.0
3	4	0.1429
3	2	-0.2
3	3	0.0
1	1	0.0
0	10	1.0
4	3	-0.1429
2	1	-0.3333

The standard deviation of the custom accuracies across the test set is approximately 0.2573, which reflects a moderate variability in prediction performance for different samples. This suggests that while some predictions are relatively close to the true values, others show significant deviation, both positively and negatively.

From the training results, we observe a final test accuracy of 0.3091 and a loss value of 7.2172. Although the model achieves an accuracy of approximately **40.91%**, this is relatively low and indicates that the model struggles to generalize well to the test data.

Figure 20 illustrates the accuracy trends across training epochs. The validation accuracy plateaus after several epochs, suggesting that the model may be starting to overfit the training data. This could be due to insufficient regularization or a mismatch in model complexity for the task at hand.

*2) **MFCC**:*

The model was trained for 100 epochs, achieving a training accuracy of 100%. However, the validation accuracy plateaued around **91.88%**, shown in Figure 21, indicating potential overfitting as training progressed. While the training loss consistently decreased, a slight increase in the validation loss further reinforces the signs of overfitting.

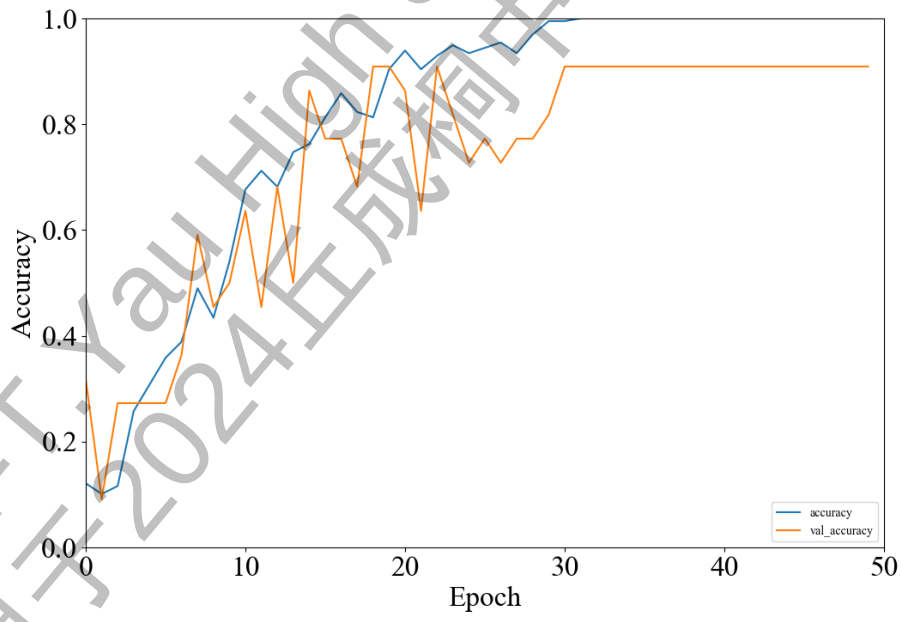

Fig. 21. Accuracy trends across training epochs for MFCC. Validation accuracy plateaued around 91.88%, suggesting signs of overfitting.

The final test accuracy achieved by the model was **95.45%**. Despite the model's exceptional

performance on the training data, the gap between training and validation accuracy suggests the model may have overfit, limiting its ability to generalize effectively to unseen data.

These minor variations in Custom loss imply that when the model does misclassify, shown in Table III, the deviation from the true label is often small.

TABLE III
TEST SET SAMPLES: TRUE LABELS AND PREDICTED LABELS FOR MFCC

True Label	Predicted Label	Custom loss
10	10	0.0
10	10	0.0
6	6	0.0
8	8	0.0
9	9	0.0
8	8	0.0
9	9	0.0
1	1	0.0
10	10	0.0
10	10	0.0
7	7	0.0
7	7	0.0
3	3	0.0
3	3	0.0
0	0	0.0
3	3	0.0
3	4	0.1429
3	3	0.0
1	1	0.0
0	0	0.0
4	4	0.0
2	2	0.0

The standard deviation of the custom accuracies ($\sigma = 0.0298$) demonstrates that the model's predictions were consistent across test samples, with minimal variation. However, the gap between training and validation performance suggests a need for improved generalization.

To address overfitting, several strategies could be considered:

- **Regularization techniques**: Implementing L2 regularization or adding dropout layers can help reduce overfitting.
- **Data augmentation**: Augmenting the training dataset can enhance the model's robustness.
- **Model complexity**: Simplifying the model architecture by reducing the number of layers or neurons can aid in preventing overfitting.

*3) **CQT-spectrograms**:*

In this section, we present the results of applying the CQT (Constant-Q Transform) approach for spectrogram analysis. Table IV summarizes the true labels, predicted labels, and the Custom loss calculated for each sample in the test set.

The CQT-based model achieved perfect accuracy on the training set, reaching 100%. However, the validation accuracy plateaued at **77.27%**, shown in Figure 22, suggesting the model may have started to overfit the training data. Despite this, the test set accuracy remained respectable at **72.73%**.

As shown in Table III, the low standard deviation ($\sigma = 0.0948$) of custom accuracies indicates that the model's predictions were relatively consistent across the test set, with minimal variability in performance between samples. This consistency suggests that the model is fairly robust, but improvements in generalization may still be necessary to close the gap between training and validation performance.

TABLE IV
CUSTOM LOSS RESULTS FOR TEST SET SAMPLES USING THE CQT APPROACH.

True Label	Predicted Label	Custom loss
10	10	0.0
10	10	0.0
6	6	0.0
8	6	-0.1429
9	9	0.0
8	8	0.0
9	9	0.0
1	1	0.0
10	10	0.0
10	10	0.0
7	6	-0.0769
7	7	0.0
3	6	0.3333
3	3	0.0
0	0	0.0
3	3	0.0
3	4	0.1429
3	2	-0.2
1	1	0.0
0	0	0.0
4	4	0.0
2	2	0.0

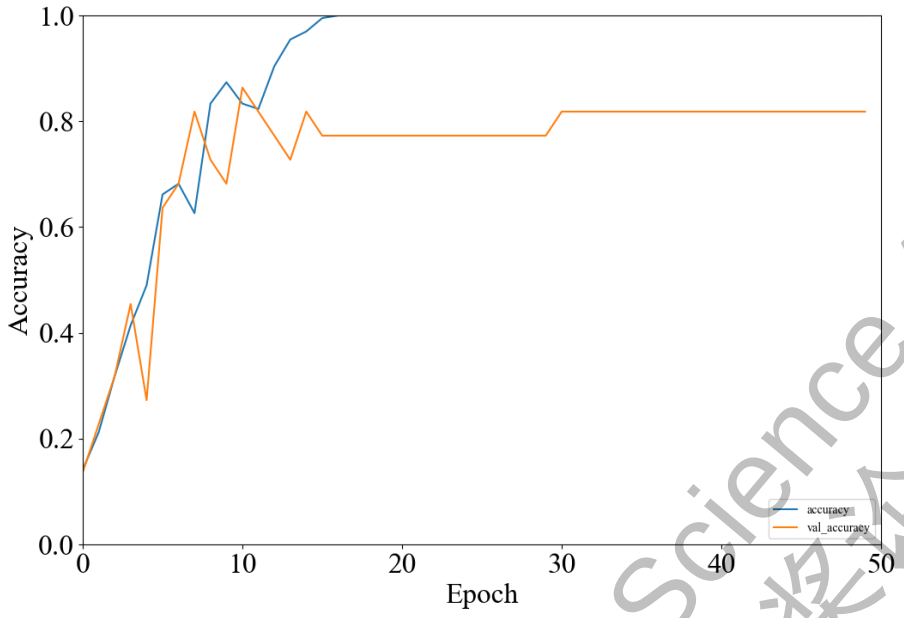

Fig. 22. Accuracy trends across training epochs for CQT. Validation accuracy plateaued around 77.27%, suggesting signs of overfitting.

4) ***Spectrogram****:*

In this section, we present the results of using the Spectrogram approach for model training. Table V summarizes the true labels, predicted labels, and the Custom loss for each test set sample.

The Spectrogram-based model achieved perfect training accuracy (100%), but its validation accuracy dropped significantly, plateauing at just **30.91%**, shown in Figure 23. This sharp contrast between training and validation performance suggests severe overfitting, where the model performs exceptionally well on the training data but struggles to generalize to unseen data.

The high standard deviation of custom accuracies ($\sigma = 0.1210$) further emphasizes the variability in the model's predictions, shown in Table IV, indicating inconsistent performance across different test samples. Ultimately, the model's final test accuracy was **50.00%**, underscoring its limited generalization capability.

TABLE V
CUSTOM LOSS RESULTS FOR TEST SET SAMPLES USING THE SPECTROGRAM APPROACH.

True Label	Predicted Label	Custom loss
10	10	0.0
10	10	0.0
6	6	0.0
8	4	-0.3333
9	8	-0.0588
8	8	0.0
9	7	-0.125
1	1	0.0
10	9	-0.0526
10	10	0.0
7	6	-0.0769
7	5	-0.1667
3	3	0.0
3	5	0.25
0	0	0.0
3	4	0.1429
3	4	0.1429
3	3	0.0
1	1	0.0
0	0	0.0
4	5	0.1111
2	3	0.2

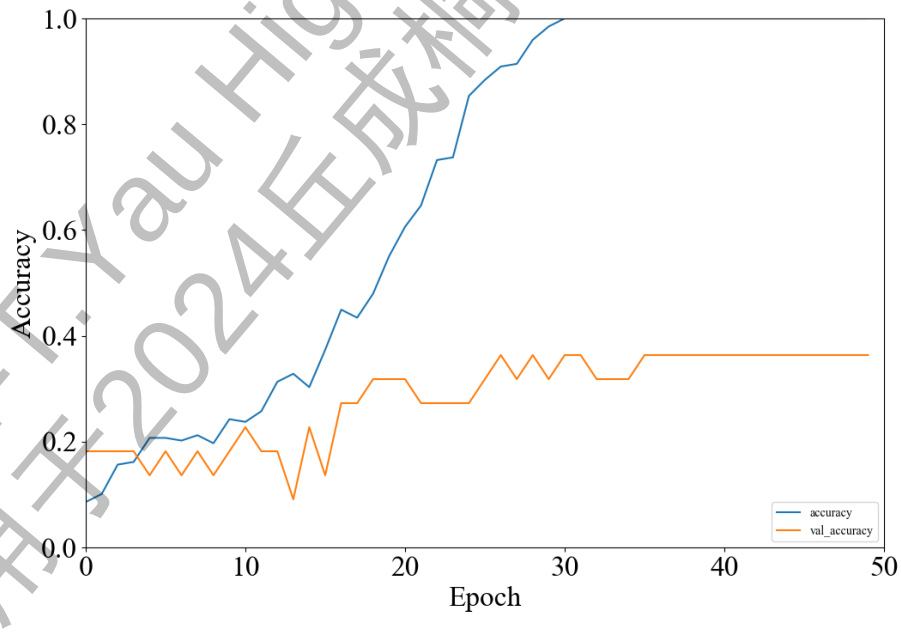

Fig. 23. Accuracy trends across training epochs for Spectrogram. Validation accuracy plateaued at 30.91%, highlighting overfitting.

*5) **Wavelet Scalograms**:*

This section presents the results of using wavelet scalograms for model training. Table VI provides a summary of the true labels, predicted labels, and the Custom loss for each test set sample.

TABLE VI
CUSTOM LOSS RESULTS FOR TEST SET SAMPLES USING WAVELET SCALOGRAMS.

True Label	Predicted Label	Custom loss
10	10	0.0
10	6	-0.25
6	4	-0.2
8	5	-0.2308
9	9	0.0
8	7	-0.0667
9	8	-0.0588
1	3	0.5
10	8	-0.1111
10	10	0.0
7	7	0.0
7	7	0.0
3	4	0.1429
3	3	0.0
0	0	0.0
3	1	-0.5
3	2	-0.2
3	3	0.0
1	1	0.0
0	0	0.0
4	5	0.1111
2	1	-0.3333

The wavelet scalograms-based model achieved perfect training accuracy at 100%, but the validation accuracy plateaued at **40.45%**, shown in Figure 24, which indicates that the model is overfitting to the training data. This performance gap suggests the model struggles to generalize well on the validation set.

As shown in Table V, the standard deviation ($\sigma = 0.1901$) of the custom accuracies suggests a moderate level of variability in the model's predictions across different test samples. The final test accuracy of **36.36%** highlights that the model's performance on the test set remains limited.

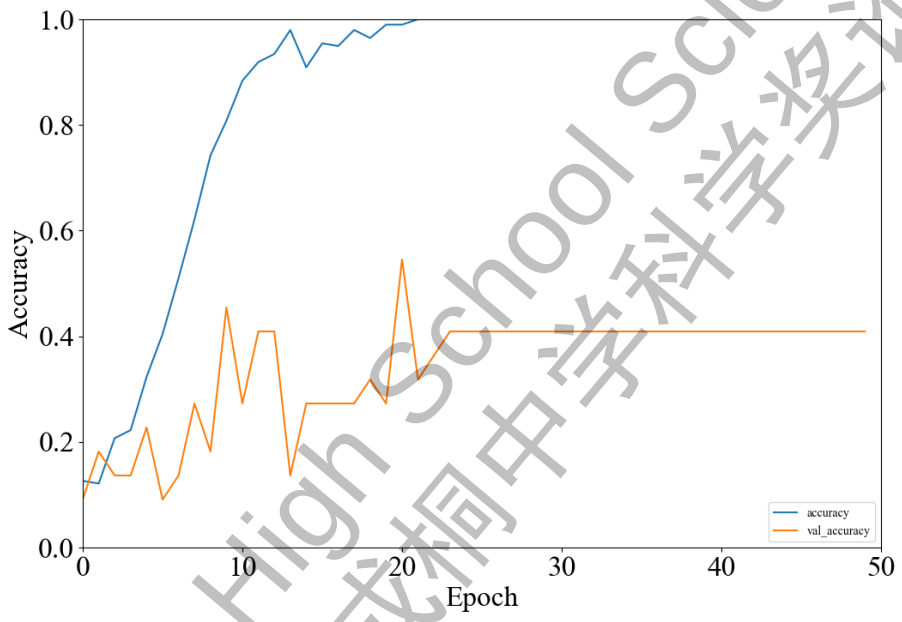

Fig. 24. Accuracy trends across training epochs for Wavelet Scalograms. Validation accuracy plateaued around 40.45%, indicating overfitting.

IV. Discussion

In acoustic analysis, the overlapping error bars in the maximum envelope area suggest that the ball numbers cannot be determined solely by examining the maximum envelope area, as shown in Figure 13. However, the sound energy shown in Figure 19 presents a different scenario, where the error bars do not overlap. This allows us to leverage the sound energy for ball estimation. By performing a polynomial fit, the energy of the balls inside the box can be calculated and compared with the test data. This comparison enables us to estimate the ball numbers. However, due to data variations, the polynomial changes, making this approach unreliable for precise estimation. Additionally, this method does not allow us to quantify accuracy effectively. To overcome these limitations, machine learning method was applied to quantify the accuracy of predictions.

The traditional model accuracy was 72.91%, where precision and recall of each catagory was calculated, shown in Figure 25.

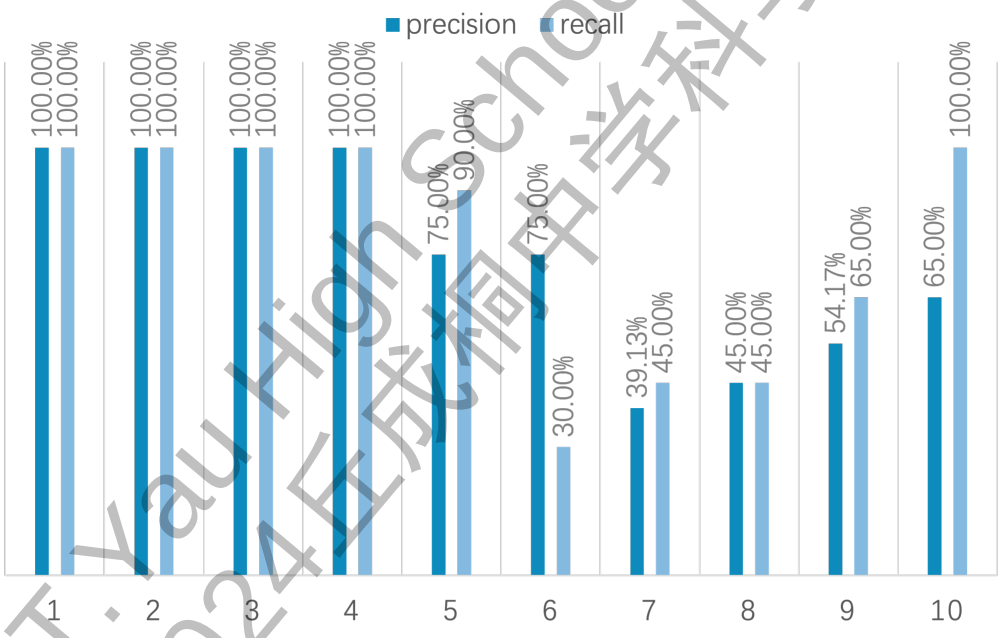

Fig. 25. Accuracy trends across categories.

The confusion matrix was acquired as well in Figure 26

The best result from our machine learning models was achieved using MFCC spectrograms, where the validation accuracy reached 0.91, indicating a 91% chance that the predictions are correct. The test accuracy further improved to 94%, meaning the error between the predicted and true labels was below

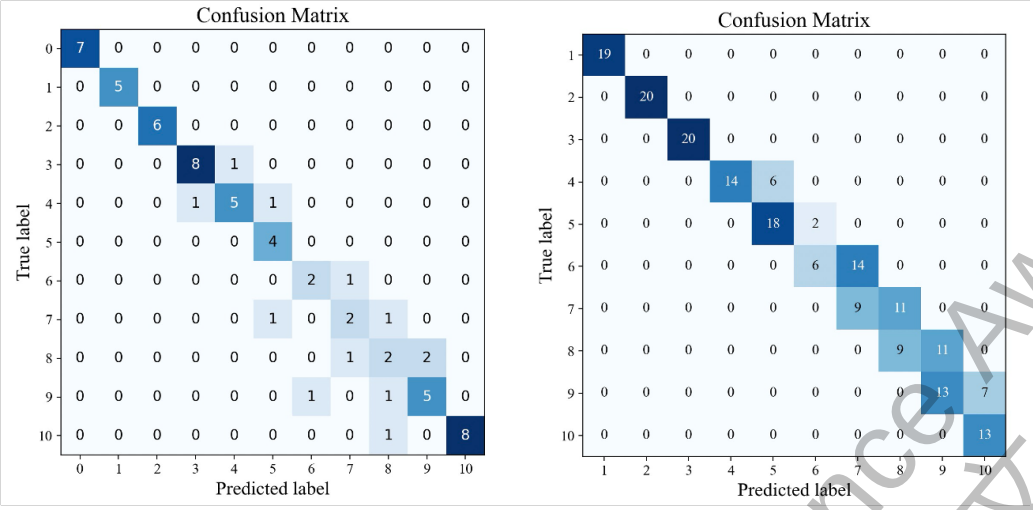

Fig. 26. The matrix on the left is the confusion matrix for machine learning method. The matrix on the right is the confusion matrix for traditional method. Through these matrices, recall and precision can be calculated.

6%. Shown in Figure 27, precision and recall of each category was calculated for this method.

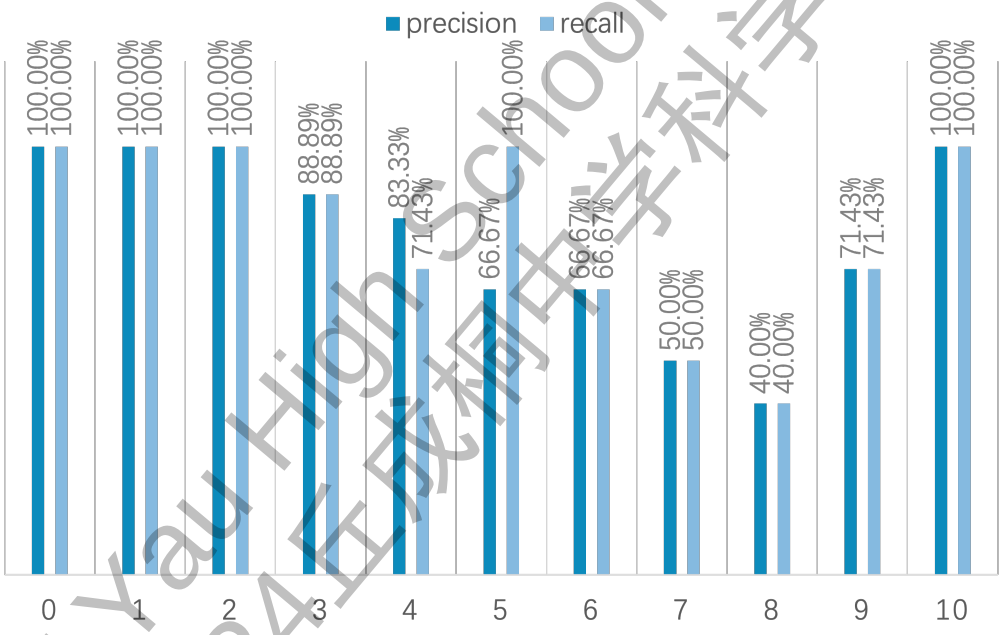

Fig. 27. Accuracy trends across categories.

In waveform analysis, directly using waveform data for machine learning resulted in moderate performance, with a test accuracy of 40.91%. The relatively high standard deviation of the custom accuracies indicates variability in the predictions, suggesting inconsistent performance across samples. This could be improved by tuning the model or incorporating additional preprocessing steps. The plateau in validation accuracy also indicates overfitting, which could be mitigated with further experimentation,

including the use of regularization techniques to improve generalization.

The CQT-spectrogram model demonstrated strong performance with a test accuracy of 72.73%. However, the plateau in validation accuracy and the gap between training and validation accuracy indicate potential overfitting. Addressing this issue could involve incorporating regularization techniques or expanding the dataset. The model's consistency, reflected in the low standard deviation of custom accuracies, is promising, but further refinement of the model architecture may be needed to enhance its generalization.

The spectrogram model exhibited classic signs of overfitting, with a significant gap between its training accuracy (100%) and validation accuracy (22.73%). Additionally, the high standard deviation in custom accuracies suggests that the model struggles to generalize effectively. To address these issues, optimization strategies such as regularization or data augmentation are necessary to improve the model's robustness and minimize overfitting.

Similarly, the wavelet scalogram model also showed clear signs of overfitting, with a large discrepancy between the training accuracy (100%) and validation accuracy (45.45%). The high variability in custom accuracies, combined with a test accuracy of just 36.36%, underscores the need for regularization techniques or more diverse training data to improve generalization and reduce overfitting.

To conclude, we acquired quantitative comparison between different prediction method, shown in Figure 28, where the advantage of machine learning method in MFCC was obviously demonstrated.

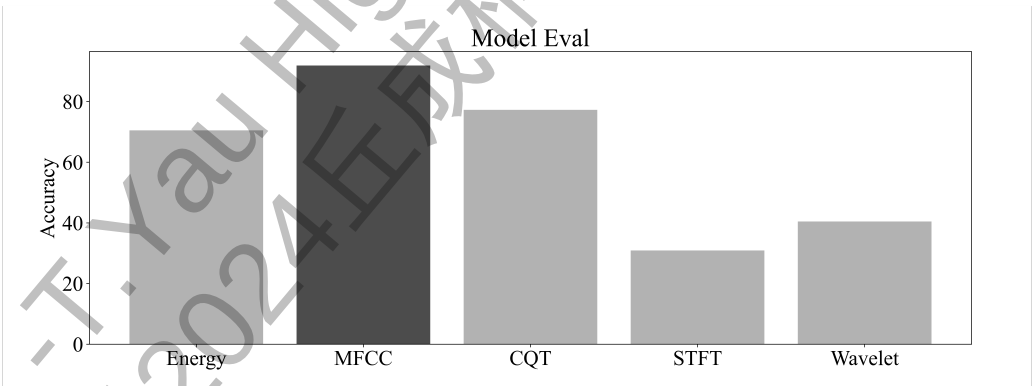

Fig. 28. Comparison between different methods.

V. CONCLUSION

In this study, a novel approach was presented for non-invasive object counting using acoustic analysis, focusing specifically on the challenge of estimating the ball numbers in a closed container based on sound recordings. By combining a theoretical model based on energy methods with a data-driven machine learning approach using CNN, accurate predictions of the ball numbers can be made.

To delve deeper into the dynamics of the collision system and enhance the understanding of the relationship between physical parameters and acoustic data, Python simulations were employed. By modeling the ball collisions within the box, a probabilistic cloud representing the potential system outcomes was generated. This simulation-based approach allowed for a comparison between the simulated probabilistic cloud and actual acoustic measurements, leading to further insights into the limitations and potential of acoustic-based ball estimation. This framework also provided a valuable understanding of inherent system uncertainties, which informed the development of more robust and accurate estimation methods.

Our analysis demonstrated that the acoustic energy generated by the balls is positively correlated with the ball numbers in the container, as evidenced by our experimental results and theoretical visualizations. Using MFCCs, CQT-spectrograms, wavelet scalograms, and other signal processing techniques, audio data was converted into diagrams suitable for CNN input, allowing the model to classify the ball numbers with reasonable accuracy.

Despite achieving 100% training accuracy, the validation results revealed overfitting in several models, indicating the need for further improvements in generalization. Potential strategies to mitigate overfitting, such as regularization, data augmentation, and model complexity reduction, were proposed. Additionally, the Custom loss metric highlighted the small deviations between predicted and true values, offering further insight into model performance.

Our approach successfully merges theoretical insights with the power of machine learning to address the object counting problem. The use of Python simulation to validate theoretical models and explore the relationships between physical parameters and acoustic features further strengthened the study. While our method shows promise, future work will focus on enhancing the model's generalization capabilities and extending the approach to larger number of objects and different types of environments.

Overall, this research opens new possibilities for non-invasive counting methods using acoustic

analysis, with potential applications in various fields such as the number of items in luggage security inspection, the number of components in automated production lines, and the number of drugs in the pharmaceutical and health industry.

REFERENCES

[1] D. StackExchange, "Continuous vs discrete signal energy," https://dsp.stackexchange.com/questions/67067/continuous-vs-discrete-signal-energy, 2024, accessed: 2024-09-10.

[2] A. Alexandridis and A. Mouchtaris, "Multiple sound source location estimation and counting in a wireless acoustic sensor network," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2015, pp. 1–5.

[3] G. L. D'Spain, "Improving the navy's passive underwater acoustic monitoring of marine mammal populations," Defense Technical Information Center, Tech. Rep., 2013. [Online]. Available: https://doi.org/10.21236/ada598800

[4] O. Amft, M. Kusserow, and G. Tröster, "Bite weight prediction from acoustic recognition of chewing," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 6, pp. 1663–1672, 2009.

[5] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, pp. 1–21, 2021.

[6] C. Mateo and J. A. Talavera, "Bridging the gap between the short-time fourier transform (stft), wavelets, the constant-q transform and multi-resolution stft," *Signal, Image and Video Processing*, vol. 14, no. 8, pp. 1535–1543, 2020.

[7] X.-L. Hu, P.-H. Ho, and L. Peng, "Statistical properties of energy detection for spectrum sensing by using estimated noise variance," *Journal of Sensor and Actuator Networks*, vol. 8, no. 2, p. 28, 2019.

[8] *Multiple time averaging and drift*. CRC Press, 1990, ch. 2.

[9] S. P. Gupta, "Speech enhancement using modified spectral subtraction algorithm," *International Journal of Engineering and Computer Science*, vol. 4, no. 12, pp. 14 849–14 852, 2016. [Online]. Available: https://doi.org/10.18535/ijecs/v4i12.35

[10] "Reference carrier frequencies, pre-emphasis characteristic and audio and control signals for 3/4-in type e helical-scan television tape cassette recording," n.d., accessed: 2024-09-10. [Online]. Available: https://doi.org/10.5594/smpte.rp87.1999

[11] M. Mottaghi-Kashtiban and M. G. Shayesteh, "New efficient window function, replacement for the hamming window," *IET Signal Processing*, vol. 5, no. 5, pp. 499–505, 2011. [Online]. Available: https://doi.org/10.1049/iet-spr.2010.0272

[12] W. L. Gans and N. S. Nahman, "Fast fourier transform implementation for the calculation of network frequency domain transfer functions from time domain waveforms," National Bureau of Standards (NBS), Tech. Rep., 1972. [Online]. Available: https://doi.org/10.6028/nbs.ir.73-303

[13] "Tuning, timbre, spectrum, scale," in *Sound on Sound*, n.d., pp. 39–50, accessed: 2024-09-10. [Online]. Available: https://doi.org/10.1007/1-84628-113-x_3

[14] M. McLaren and Y. Lei, "Improved speaker recognition using dct coefficients as features," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4430–4434.

[15] L. Bahatti, M. Zazoui, O. Bouattane, and A. Rebbani, "Short-term sinusoidal modeling of an oriental music signal by using cqt transform," *Journal of Signal and Information Processing*, vol. 4, no. 1, pp. 51–56, 2013. [Online]. Available: https://doi.org/10.4236/jsip.2013.41006

[16] J. P. Keradec and X. Margueron, "Improved frequency resolution dft eases teaching fft analysis and provides better amplitude accuracy," in *2005 IEEE Instrumentation and Measurement Technology Conference Proceedings*, vol. 2, 2005, pp. 1149–1154, n.d. [Online]. Available: https://doi.org/10.1109/imtc.2005.1604324

[17] S. Koshita, Y. Kumamoto, M. Abe, and M. Kawamata, "High-order center-frequency adaptive filters using block-diagram-based frequency transformation," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 4284–4287.

[18] A. Jalali and M. Bagheri, "Gas reservoir detection using mixed components short time fourier transform (mc-stft) as a new attribute," 2023, preprint. [Online]. Available: https://doi.org/10.21203/rs.3.rs-3445403/v1

[19] "Fast algorithms for wavelet transform computation," in *Time Frequency and Wavelets in Biomedical Signal Processing*, 2010, accessed: 2024-09-10. [Online]. Available: https://doi.org/10.1109/9780470546697.ch8

[20] *The Continuous Wavelet Transform*, n.d., accessed: 2024-09-10. [Online]. Available: https://doi.org/10.1887/0750306920/b833c2

[21] X. Jia, "Cnn input layer and convolution layer," in *Field Guide to Hyperspectral/Multispectral Image Processing*, 2022. [Online]. Available: https://doi.org/10.1117/3.2625662.ch83

[22] B. Pradhan and M. I. Sameen, "Manifestation of svm-based rectified linear unit (relu) kernel function in landslide modelling," in *Space Science and Communication for Sustainability*, 2017, pp. 185–195. [Online]. Available: https://doi.org/10.1007/978-981-10-6574-3_16

[23] A. F. Nurjannah, A. S. D. Kurniasari, Z. Sari, and Y. Azhar, "Pneumonia image classification using cnn with max pooling and average pooling," *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, vol. 6, no. 2, pp. 330–338, 2022. [Online]. Available: https://doi.org/10.29207/resti.v6i2.4001

[24] H. Nakahara, T. Fujii, and S. Sato, "A fully connected layer elimination for a binarized convolutional neural network on an fpga," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–4. [Online]. Available: https://doi.org/10.23919/fpl.2017.8056771

[25] D. B. Prakash, K. A. Kumar, and M. Rishitha, "Deep cnn: Classification of known classes and unknown classes based on softmax prediction probabilities," in *2023 4th International Conference for Emerging Technology (INCET)*. IEEE, 2023, pp. 1–11. [Online]. Available: https://doi.org/10.1109/incet57972.2023.10170405

[26] S. E. Golann, J. M. Von Felsinger, and R. Beavais, "The training gap: Discrepancy between training and practice in community psychology," 1969. [Online]. Available: https://doi.org/10.1037/e463452008-353

VI. ACKNOWLEDGMENTS

We would like to express our deep gratitude to our advisor, Professor Dong Zhang from Nanjing University. He not only gave us valuable guidance in theory and data analysis, but also instructed us in the discipline of essay writing. His invaluable guidance, insightful feedback, and unwavering support have been the cornerstone of this study. In addition, we would like to thank Professor Juan Tu, Professor Jing Lu and Professor Xiasheng Guo from Professor Zhang's team for their suggestions on the structure and writing of the paper, which helped us to complete this paper with high quality. Their expertise and encouragement have helped us navigate countless challenges and significantly shaped both the direction and quality of this research. Their commitment to excellence has both enriched this work and fostered our personal and professional growth. We, as a team, are especially thankful for their patience and understanding during the more difficult phases of this journey.

We would also like to extend our heartfelt thanks to our family and friends, whose unwavering support provided us with a sanctuary of rest and emotional strength during times of exhaustion. Their belief in us has been a constant source of motivation, and their companionship and occasional bursts of creativity have guided us through the challenges of this research with resilience and optimism.

We are truly grateful to the Yau committee for providing us with this wonderful opportunity. It was through their platform that we came together to tackle this problem as a team, demonstrating the power of unity and cooperation. This experience not only allowed us to grow as individuals but also left us with lasting memories.

Finally, we would like to thank our team members for their dedication—staying up late to complete tasks, offering help when needed, facing difficulties with courage, and persevering throughout the research process. Each member played an indispensable role in the success of this project.

This project stems from Problem 1 of the 2024 International Young Physicists' Tournament, where it is required to distinguish the ball number in a closed container by recording its acoustic data.

A. Author introduction

Zixuan Peng:

Grade: 11

Gender: Male

School: Nanjing Foreign Language School

Contribution: Zixuan designed the complete experiment, collected the data using a DIY robotic mechanism and finished all data recording in a laboratory. He is also in charge of the ball collision simulation. He wrote the abstract, introduction, theoretical analysis and conclusion part of the paper. He also greatly contributed to the construction of the CNN model.

Awards:

- MYPT 2024 No.1 (Team), Best Opponent (Individual)
- BPhO Gold 2023
- Physics Bowl Gold 2024
- AIME 11/15
- AMC 12B Perfect Score
- Harvard-MIT Math Tournament 2024 February Individual round top 15 (Geo)
- Duke Math Meet China No.1(Team), Best Individual (No.1 in Individual Round)

Xiaoxi Zhou:

Grade: 12

Gender: Female

School: Nanjing Foreign Language School

Contribution: Xiaoxi is in charge of data analysis and CNN machine learning model construction. In this paper, she wrote the acoustic data preparation, analysis of acoustic data, energy of different balls, prediction of balls using CNN model, and results and discussion.

Awards:

- S.-T Yau High School Honorable Award 2023 Math
- IYPT 2024 Best Reviewer (Individual)
- BPhO Gold 2023
- Physics Bowl Gold 2023
- AIME 12/15

Jifan Zhang:

Grade: 11

Gender: Male

School: Nanjing Foreign Language School

Contribution: Jifan is responsible for the literature research. He was involved in setting up the experimental environment, the ball collison simulation, and wrote the related work and part of introduction and conclusion.

Awards:

- BPhO Gold 2023
- AIME 13/15
- MYPT Best Reporter (Individual)

VII. Data Availability

The datasets established for the research work presented in this paper, as well as the associated algorithm codes, are available from the authors upon reasonable request. In order to facilitate more peers to work on this research together, the datasets and related code of this paper have been publicly available in the Github repository at the following URL: https://github.com/GlitchyPeng/AcousticCountingDatabase. The code used in the simulation is also available here in this GitHub repository at the following URL: https://github.com/GlitchyPeng/Ball_Collision_Simulation.

Researchers interested in utilizing the data for their own studies are encouraged to appropriately reference this work. Detailed instructions on how to access and use the data and the code are provided on the repository homepage.

VIII. MODIFICATIONS

We added comparisons between the energy calculated using the multi-modal simulation model and the energy acquired in our experiment. Using the inverse function of our theoretical solution, ball numbers can be predicted and compared with the original simulation ball number. However, this method does not perform as well as machine learning method.

We revised the figures in spectrum subtraction because waveforms could not clearly demonstrate the subtracted spectrum.

We revised the figure for FFT envelope as before envelopes overlap and could not clearly demonstrate the increasing trend with ball numbers.

We added confusion matrices to further demonstrate performances of each method.