International Journal of Precision Engineering and Manufacturing
https://doi.org/10.1007/s12541-024-01018-3

Online ISSN 2005-4602
Print ISSN 2234-7593

REGULAR PAPER

Operation and Productivity Monitoring from Sound Signal of Legacy Pipe Bending Machine via Convolutional Neural Network (CNN)

Eunseob Kim[1,2] · Daeseong Mun[1] · Martin B. G. Jun[1,2] · Huitaek Yun[3]

Received: 22 January 2024 / Revised: 1 April 2024 / Accepted: 1 April 2024

Abstract
This study introduces a non-invasive approach to monitor operation and productivity of a legacy pipe bending machine in real-time based on a lightweight convolutional neural network (CNN) model and internal sound as input data. Various sensors were deployed to determine the optimal sensor type and placement, and labels for training and testing the CNN model were generated through the meticulous collection of sound data in conjunction with webcam videos. The CNN model, which was optimized through hyperparameter tuning via grid search and utilized feature extraction using Log-Mel spectrogram, demonstrated notable prediction accuracies in the test. However, when applied in a real-world manufacturing scenario, the model encountered a significant number of errors in predicting productivity. To navigate through this challenge and enhance the predictive accuracy of the system, a buffer algorithm using the inferences of CNN models was proposed. This algorithm employs a queuing method for continuous sound monitoring securing robust predictions, refines the interpretation of the CNN model inferences, and enhances prediction outcomes in actual implementation where accuracy of monitoring productivity information is crucial. The proposed lightweight CNN model alongside the buffer algorithm was successfully deployed on an edge computer, enabling real-time remote monitoring.

Keywords Machine sound monitoring · Sound recognition · Convolutional neural network · Remote monitoring

1 Introduction

Machine monitoring is crucial to provide visibility into manufacturing operations that drives improvement from the shop floor. With the advent of manufacturing equipment equipped with Information and Communication Technology (ICT) and Internet of Things (IoT) capabilities, effective monitoring and the utilization of Artificial Intelligence (AI) have become more feasible [1, 2]. These advancements are allowing manufacturing companies to progressively depend more on the collection, analysis, and exchange of data within interconnected production systems [3]. Manufacturing companies are progressively depending more on the collection, analysis, and exchange of data within interconnected production systems. With the adoption of Industry 4.0 principles, manufacturing processes have become more efficient, cost-optimized, and have introduced new business models, resulting in heightened competitiveness in the market [4]. Despite these advancements, industry still relies on a significant amount of legacy equipment, which remains pivotal to industrial operations [5]. A "legacy machine" is characterized as manufacturing equipment that inherently lacks the capacity for external communication or the existence of an application programming interface (API) that would facilitate data exchange. The "legacy" pertains to the inherent functionalities of the device rather than the chronological age of the equipment [6]. It poses challenges in modern manufacturing environments because of incompatibility with the state-of-the-art technology. In other words, legacy machines are not able to communicate with or connect to newer machines or systems [7, 8]. Due to this downside,

Eunseob Kim and Daeseong Mun have contributed equally to this work and designated as co-first authors.

✉ Huitaek Yun
htyun@kaist.ac.kr

[1] School of Mechanical Engineering, Purdue University, West Lafayette, IN 47906, USA

[2] Indiana Next Generation Manufacturing Competitiveness Center (IN-MaC), Purdue University, West Lafayette, IN 47906, USA

[3] Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, Republic of Korea

Published online: 15 April 2024

integrating legacy machines into a modern manufacturing environment entails high costs. Despite the challenge, manufacturers continue using legacy machines because these are still in good working order in terms of producing parts. Over time, expertise in using these legacy machines has accumulated, and they have been optimized for stable operation, making it difficult to simply replace them. Moreover, a tremendous amount of budget is spent to replace the legacy machines with the latest machines that are equipped with exchanging data. This decision turns in reduced efficiency, increased maintenance costs, and limited scalability [6]. Maintaining and upgrading legacy machines are an important consideration for companies who expect to stay competitive in the global marketplace, especially for small manufacturing enterprises (SMEs) [9, 10]. Retrofitting is defined as a process that involves altering or adding devices to traditional machinery, aiming to improve their efficiency and functionality while minimizing financial and time expenses and risks without the need for replacing equipment [11]. However, retrofitting the legacy machines is challenging to most SMEs. The shortage of relevant skilled personnel [3, 12] as well as the lack of awareness of advanced techniques to upgrade the machines [13, 14] make retrofitting the legacy machines difficult in SMEs.

To overcome these challenges, low-cost methods have been studied to enable legacy machines to be monitored. Extracting data directly from programmable logic controller (PLC) [5, 15, 16] or installing sensor in the electrical circuit [10], hard wiring [7], or power supply [6, 17, 18] are intrusive ways that involves unwanted downtime to deploy the sensors and then stabilize the machine. Furthermore, when it comes to the lack of cybersecurity capability in SMEs [19, 20], the intrusive approach is threatened because manipulation of the deployed solution can potentially affect the operation of the machine and even disable it. This makes SMEs hesitate to adopt direct connections to PLCs or electrical circuits of the machines. Computer vision, vibration monitoring, and sound recognition are typical methods as non-intrusive monitoring for machine state and operation in the context of retrofitting legacy machines. Computer vision techniques with affordable cameras such as webcam were applied for monitoring of legacy computer numerical control (CNC) machine tools [21–23], manual production and assembly process [24], and various shop floor artifacts such as equipment control panel, analog gauge, and so on [21]. Nonetheless, the presence of cameras during machine operation can impede the operator's line of sight. Conversely, there is a potential for operators or other objects to obstruct the camera's view, a common occurrence on a real factory floor, which causes trouble with vision detection. Moreover, in any vision system, object detection is sensitive to light conditions of the environment [25, 26], which can pose challenges when implementing it on real factory floor. On the other hand, vibration and sound monitoring offers an alternative non-intrusive approach as machines generate vibrations and emits sounds, producing distinct vibrational and acoustic behaviors at different operational states. Accelerometers and acoustic emission (AE) sensors were utilized for machine operation and prognostic monitoring [27–29]. Although vibration monitoring using accelerometer and AE sensor is known for its high accuracy and has been extensively researched, its practical implementation is challenging due to the requirement for additional apparatus such as a data acquisition (DAQ) system and amplifier, which contributes to high costs. To address this issue, low-cost accelerometers such as micro-electromechanical systems (MEMS) and piezoelectric-based sensors were also employed to monitor the operational states of the machine [30–33]. These low-cost accelerometers have inherent limitations including drifting [34, 35], narrow frequency band and low resolution [31], and phase lag [32], which hinders the factory to utilize them. In the meantime, as sound sensors are becoming more affordable, sound recognition is a practical technique for monitoring machine operations. However, the adaptability of sound signals for machine and process monitoring is poor [29], especially in a real shop floor environment, mainly due to noise from neighboring machines [36] and difficulty in localizing sound signals [37].

To address the challenges in sound monitoring, the adoption of AI techniques and the development of a new sound sensor to reduce noise have been applied to make sound recognition a more feasible solution for monitoring machine operations. Previously in our group, a stethoscope-based internal sound sensor has been developed [38]. To capture internal sound, it consists of a stethoscope and a USB microphone attached to the end of a rubber tube. The details of configuration and system identification are shown in [38]. It showed better prediction accuracy than a microphone when the same signal processing and model were applied to running state prediction of CNC machine tools and their subcomponents [39]. It was also utilized to identify anomalies caused by heavy lifting of robot arm [40] and predicting cutting state and productivity of CNC tube cutting machine [41]. Sound recognition framework based on MTConnect to stream multiple sound streams was suggested and evaluated in predicting accuracy and response time [41]. Moreover, other recent studies showed that machine learning (ML) and deep learning (DL) techniques for sound recognition are able to predict manufacturing process and machine operation. Dynamic time wrapping (DTW) was evaluated to predict operational status of legacy machines, showing that feature extraction affects speed and accuracy of prediction [36]. A study [42] introduced a learning-based acoustic defect detection (LearnADD) method for automating bottle inspection and compared different ML and DL techniques, revealing that LSTM exhibited the highest accuracy performance.

Implementing DL models on edge computers is pivotal in sound monitoring applications, where real-time data processing with minimum latency is vital [43, 44]. Conducting sound

classification directly at the source not only enhances the responsiveness and accuracy of event detection but also preserves bandwidth and ensures data privacy by limiting the transmission of sound data through networks. Convolutional neural network (CNN) architecture using sound signals was successfully applied for real-time prediction of the operational state of multiple machines simultaneously [39, 45, 46]. In the study [46], the entire processing time for machine operational sound monitoring was 8 seconds even the efforts to reduce the inference time using a lightweight CNN architecture. For environmental sound classifications using a shallow CNN architecture [47], they achieved a short inference time averaging 0.255 seconds. In our previous study [41] concluded that supplementary algorithms on edge computers were necessary for robust prediction based on the CNN inferences. Furthermore, most prior research on edge computing implementation focused on simple binary classification tasks such as determining the ON or OFF status of machines and identifying cutting operations [36, 39, 41, 46].

Based on our previous work, this study was extended to apply the internal sound sensors for monitoring multiple operational states and productivity of a legacy manufacturing machine. This research suggests a workflow to create a lightweight CNN model aiming to reduce computation load considering deployment to edge computer and an algorithm utilizing sequential CNN inferences to improve the performance of productivity prediction.

2 Monitoring System

The target system is a legacy pipe bending machine (SB-22X8A-MR-V-U, SOCO) on a real shop floor. Other machines such as CNC mills, band saws, and welding stations are located near the pipe bending machine, causing a noisy environment. The pipe bending machine has eight electric servo control axes, which enables this machine to draw, rotate, and bend the pipe. The bending amount, direction, and number of cuts depend on the final shape of the part. This machine bends the raw straight metal pipe and yields the bent pipe as an intermediate product. The operator manually feeds the raw pipe into the machine in a timely manner so that the machine bends and cuts the pipe as it is pre-programmed. The schematic of the monitoring system for the pipe bending machine and part example are shown in Fig. 1. The details of sensor deployment and data collection are as follows.

2.1 Sensor Deployment

Four sound sensors were installed at different locations of the pipe-bending machine. All sensors were connected to a single edge device (Raspberry Pi 4B). Aside from sound sensors, three webcams (CyberTrack H4 web camera, ADESSO) were also installed at the high points around the pipe bending machine where each camera monitors the operation, and the captured images represent the context of sound data. The four sound sensors used in the study are denoted from 1 to 4. Sensor 1 is a USB microphone (K053, Fifine Microphone) to capture ambient sound, which was affixed to the backside of Sensor 2. Sensors 2, 3, and 4 are the internal sound sensor [38], the combination of a stethoscope (Littmann Classic III, 3M) and a microphone (the same USB microphone as Sensor 1). Sensor 2 was installed on the surface of the main bed which acts as the base for the rest of the components. Sensor 3 was installed on the surface of the front bed, which is the most frequently moving component to bend a pipe. Within the front bed, there are clamps capable of securing raw pipes of different sizes, along with a shear cutting blade used to trim each part after bending. Sensor 4 was installed on the hydraulic

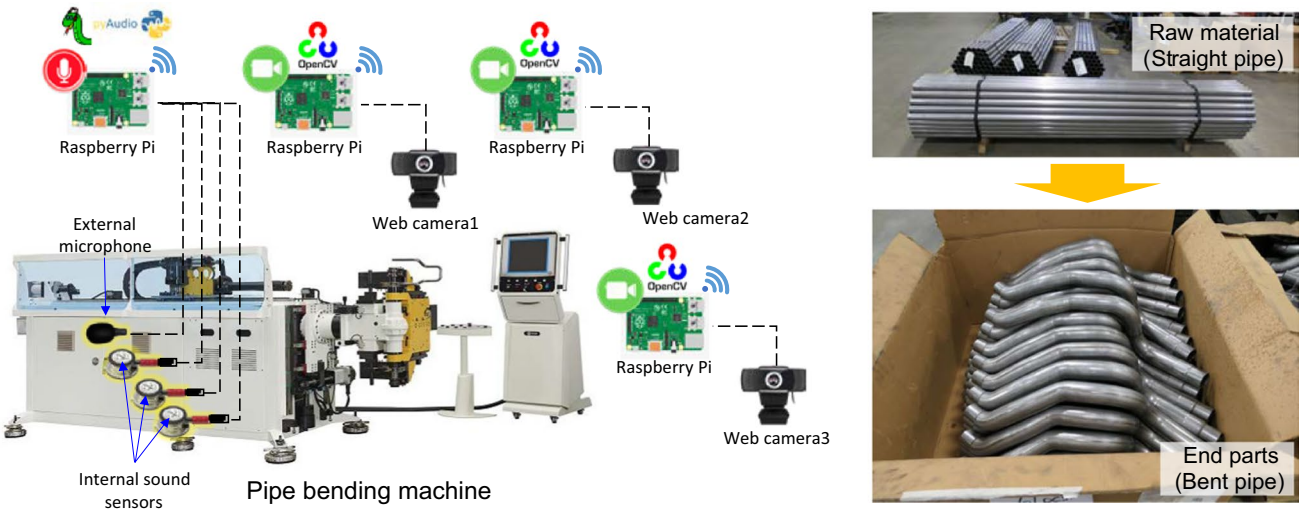

Fig. 1 Outline of pipe bending machine monitoring system (left) and raw material and end part (right)

Table 1 Type and location of sound sensor

Number	Type	Location
Sensor 1	External microphone	Main bed
Sensor 2	Internal sound sensor	Main bed
Sensor 3	Internal sound sensor	Front bed
Sensor 4	Internal sound sensor	Hydraulic pump

pump which provides the power to clamp the pipe and closes the cutting blade. The initial locations for the sound sensors were determined through discussions with the machine operator and shop floor manager, aiming to choose spots that would not interfere with the machine's operations or production processes while still effectively capturing the machine's sounds. The types and locations of the sound sensors are summarized in Table 1. The locations of the sound sensors and webcams are shown in Fig. 2, respectively. The stationary sample frame image from the video of each webcam is shown in Fig. 3.

2.2 Data Collection and Labeling

After the sensors were deployed, both sound and vision data were collected. A customized sound collection program to capture sound signals from all sensors simultaneously was written in Python using advanced Linux sound architecture (ALSA) and PyAudio module because Raspberry Pi OS supports ALSA for hardware and software interfaces of sound devices. In Windows OS, WASAPI can be utilized to ensure compatibility and efficient sound data collection. A webcam image capturing program was also developed by Python and OpenCV library. In daily data collection, programs for webcam video and sound were started at the same time. From the timestamp printed on each frame of video as on the left top of each frame in Fig. 3, the vision data and sound data were synchronized correctly by using a network time protocol (NTP) server for all the embedded computers. The specifications of the collected data according to sensors are summarized

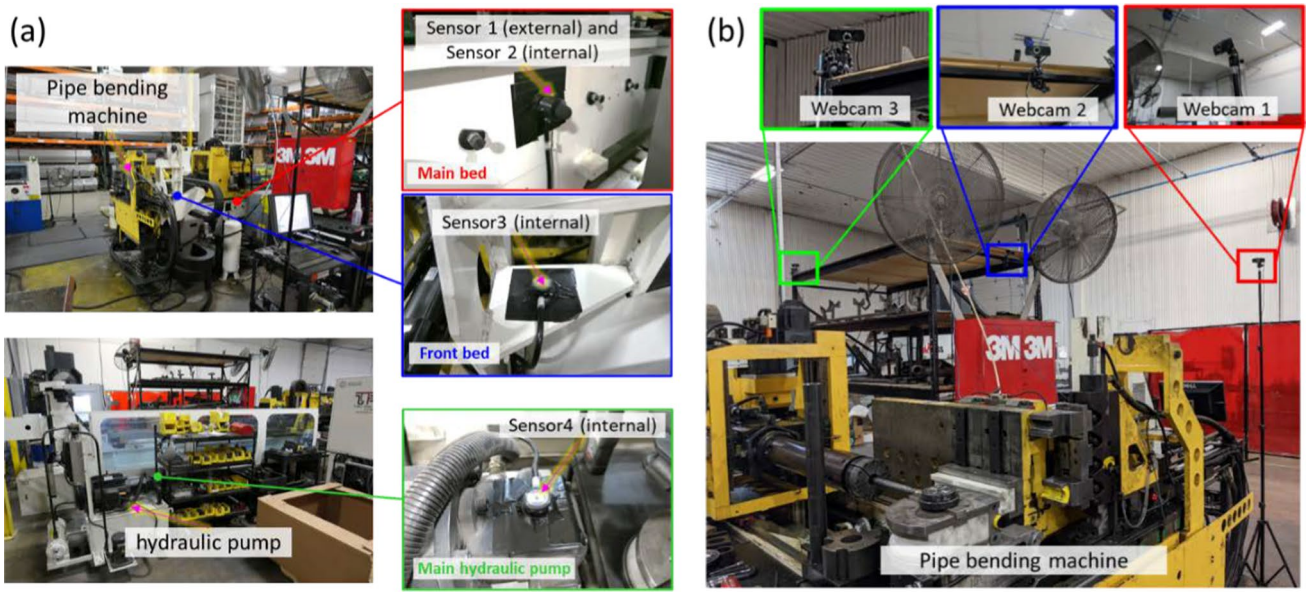

Fig. 2 Sound sensor (**a**) and webcam (**b**) placement for pipe bending machine

Fig. 3 Captured images of webcam videos: webcams 1 (left), 2 (middle), and 3 (right)

Table 2 Specification of collected data

Data type	Category	Specification
Sound	Format	WAV
	Duration per day	1 h
	Sampling rate	48 kHz
	Channel	1
	Audio bit depth (resolution)	16-bit
	Number of sensors	4
Vision	Format	AVI
	Duration per day	1 h
	Frames per second	30 fps
	Frame size	640×480 pixels
	Number of sensors	3

Table 3 Definition of operational state for labeling

Label	Definition
Off	Machine is powered off
Idle	Machine is ready to run without any axis movements
Load	Operator feeds a straight pipe into the machine
Executing	The machine bends, rotates, or draws the pipe
Cut	The cutting blade trims the pipe and yields a product or a by-product

in Table 2. After creating the files of the collected data from the sensors, embedded computers automatically uploaded the files to cloud storage to secure the disk space.

The collected data was labeled with five different classes according to operational states: 'Off', 'Idle', 'Loading', 'Executing', and 'Cut'. The definition of each label is described in Table 3. When an operator runs the pipe bending machine, the operator first feeds the straight pipe into the machine and then it draws the pipe in the axial direction. The machine moves to bend and cut the pipe to produce parts. The sum of 'Loading', 'Executing', and 'Cut' time represents the run time of the machine. Therefore, machine utilization can be estimated according to batch, part, shift, and time, which can contribute to production scheduling and optimization. The operational state and sound signals of a single cycle for 90 s are shown in Fig. 4. Figure 4a is the operational state event plot, Fig. 4b is raw sound signals in the time domain, and Fig. 4c is sound spectrograms by short-time Fourier transform (STFT). In Fig. 4b, c, Sensor 1 to 4 signals are located from top to bottom, respectively. In Fig. 4b, the y-axis represents normalized amplitude of sound. The time range from 0 to 90 seconds is the same in all the plots and each time axis is aligned vertically. The first 'Executing' and 'Cut' produced a byproduct and the following three 'Executing' events produced three products. As in Fig. 4, each operational state generates unique sounds. When we listened to the recorded sound after analyzing webcam videos, it was able to figure out which operational state generated corresponding sound. Moreover, according to sensor type and placement, different sounds were captured despite the same machine being monitored. Sensor 4 as in the bottom plots of Fig. 4b, c showed distinct sound signals when cutting occurred whereas it could not be able to capture other operational states.

The labels and accordance timestamps were created from the collected webcam videos as illustrated in Fig. 5. The sound signal example in Fig. 5 is from Sensor 1. The data and time range of Fig. 5 are the same as Fig. 4, which means data from 0 to 20 s of Fig. 4 was used to describe labeling procedure in Fig. 5. First, webcam videos were analyzed to define the operational state with the start and end timestamps of each state. Second, the operational state data and the sound signals were synchronized using the given timestamps. Third, the entire sound signal was scanned with a frame size of 0.98 s. The chunk length when reading the audio signal from the USB microphone is 2 to the power of n where n is a positive integer. In this monitoring system, the sound chunk size was 2^{11} sample points. This chunk translates to approximately 42.67 ms of sound at a sampling rate f_s of 48 kHz. Considering further real-time implementation using the same sound data flow framework, the frame size, which is 23 chunks, 47,104 sample points, and 0.981 s, of the nearest 1 s was chosen. The frames were eventually used to input data for training models. The interval of each scanning frame was the chunk length. Therefore, the first frame starts at 0 s as in Fig. 5, the second frame starts at 43.67 ms (a chunk), and the second frame starts at 85.33 ms (two chunks). The start, center time, and end time of the frame were defined as $t_{f,s}$, $t_{f,c}$, and $t_{f,e}$, respectively in Fig. 5. The superscript means the frame number. In case the operational state is changed in a frame, the label for the frame was determined based on the center time of the frame. For example, if a frame's center time $t_{f,c}$ falls within an operational state, it is labeled as such. Thus, labels for all the frames were determined by scanning the entire sound signals. All example sound signals according to the sensor and operational states are plotted in Appendix.

2.3 Sound Feature Extraction

Upon segmenting and labeling the sound signals from the sensors into input frames, the log-Mel spectrogram was adopted to extract features. These features were subsequently used to train the CNN models. The log-Mel spectrogram is the time–frequency representation of the sound signal by applying Mel scale on the short-time Fourier transformation (STFT) [48]. The Mel scale refines frequency data using Mel filter banks on the STFT, reducing the feature size for CNN models. This ensures training efficiency while preserving high accuracy in sound recognition. The Mel spectrogram has been widely employed for musical and speech recognition [49, 50] as well as machine sound monitoring [39, 41,

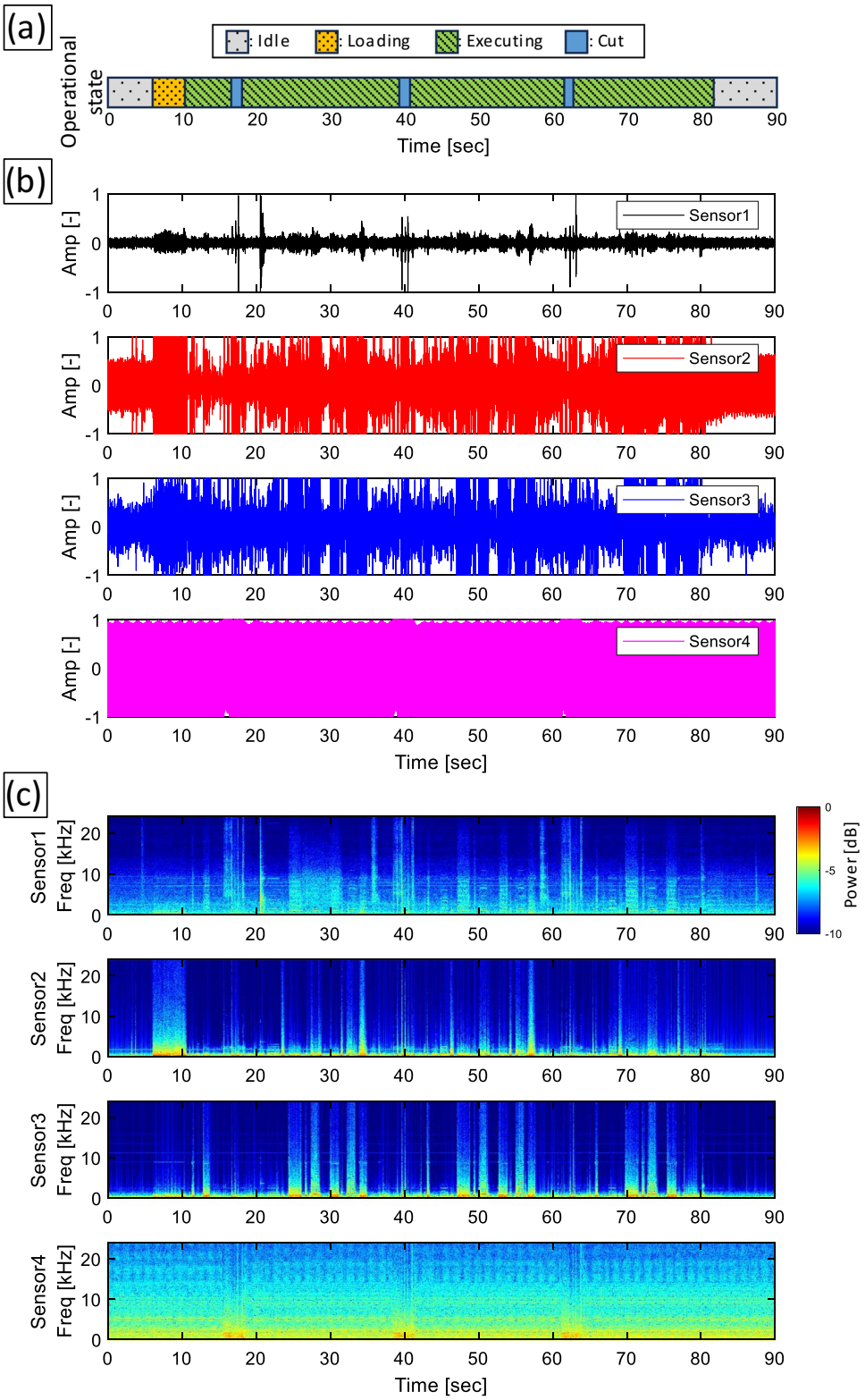

Fig. 4 A single cycle data of pipe bending machine for 90 s: **a** operational state, **b** time domain, and **c** sound spectrogram

46]. The Mel scale indicates how humans perceive the frequency of a pure tone compared to its objectively measured frequency. The relationship between the Mel and frequency is expressed in Eq. (1).

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

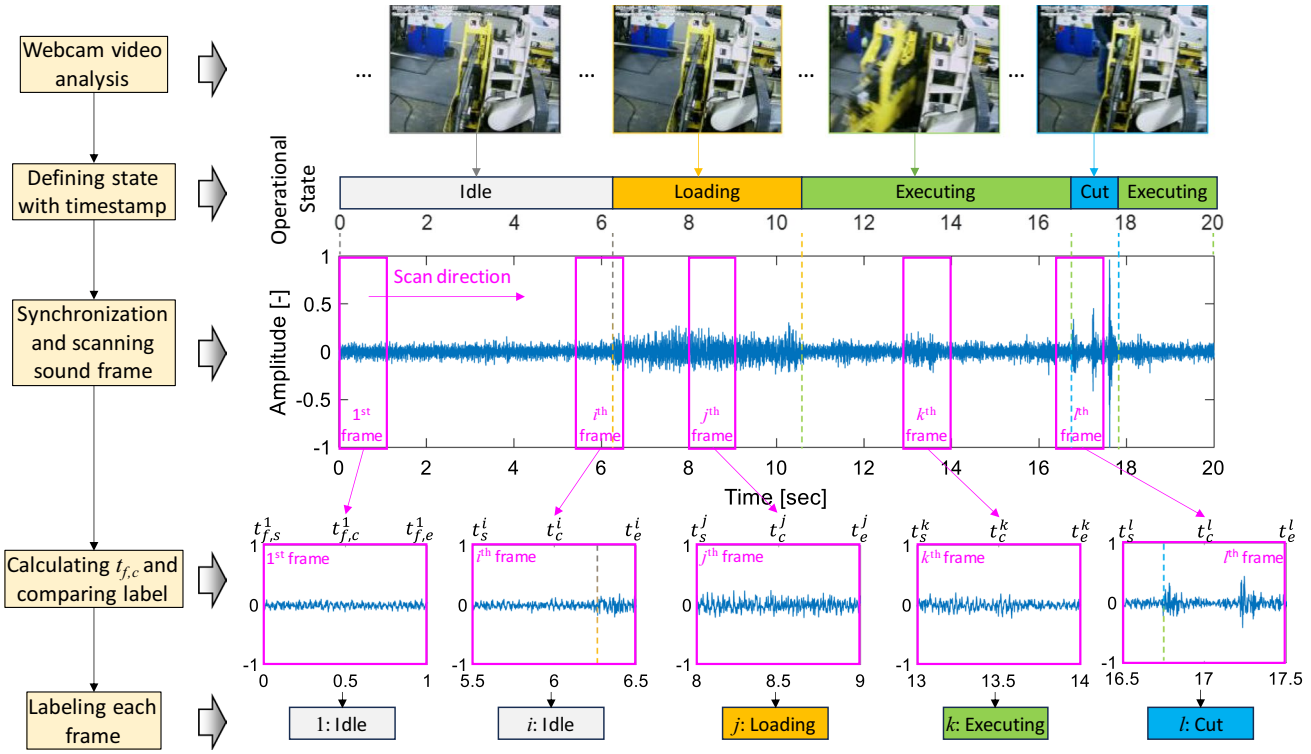

Fig. 5 Operational state labeling procedure

where M and f represent Mel and frequency, respectively. This expression is calibrated so that a frequency of 1000 Hz corresponds to 1000 Mels. Below 1000 Hz, the relationship between Mel and frequency is nearly proportional. For higher frequencies, however, the relationship is logarithm. In this study, 120 Mel bands within a frequency range of 20 Hz to 24 kHz were chosen for feature extraction in training the CNN model. The lower frequency limit (20 Hz) was set based on the minimum frequency range of the USB microphone, while the upper limit (24 kHz), half of the sampling rate, was chosen as the maximum frequency. When transforming the time domain information to the STFT, A Hamming window with 2048 points of FFT (fast Fourier transform) was applied as a windows function to reduce the spectral leakage enabled by smooth tapering at the edges. Due to the shape of the window function in the time domain, there is information loss near the edge. To counteract this loss, a 50% window overlap was employed. The input feature dimension of the Log-Mel spectrogram is 120×47 where the number of Mel bands is 120 and the number of windows is 47.

3 CNN Training and Prediction Algorithm

3.1 Dataset

Datasets were collected on various production days for different parts to train and test the CNN models. Figure 6 illustrates the structure of dataset utilized for both training and testing phases. The CNN model was trained using one hour of actual production data from a single part production (Day 1, Part A). Test datasets were curated to assess the model's performance on identical as well as different parts. Different parts have various shapes, materials, cycle time, and so on. Arbitrary identifiers, such as Part A, B, and Day 1, 2, and so on, were used for parts and days.

Figure 7 summarizes the distribution of label count for each dataset. The 'Loading' label consistently registered the shortest duration, indicating the least frequent occurrence among all labels. Table 4 provides a summary of the cycle time and operational states' statistics. Each cell in the table represents averaged value and standard deviation. Cycle time was defined as the working duration for a single raw material. Notably, the cycle time and operational states exhibited variations depending on the parts involved.

3.2 CNN Model Training

Convolutional neural network (CNNs) is one of the widely used Deep learning (DL) models in audio classification tasks for their ability to learn complex patterns [51]. While the complexity of a CNN model is crucial for accuracy, it is important to avoid excessive complexity in edge computing with limited resources. Minimizing prediction time is also a key for real-time monitoring. Therefore, it is advantageous

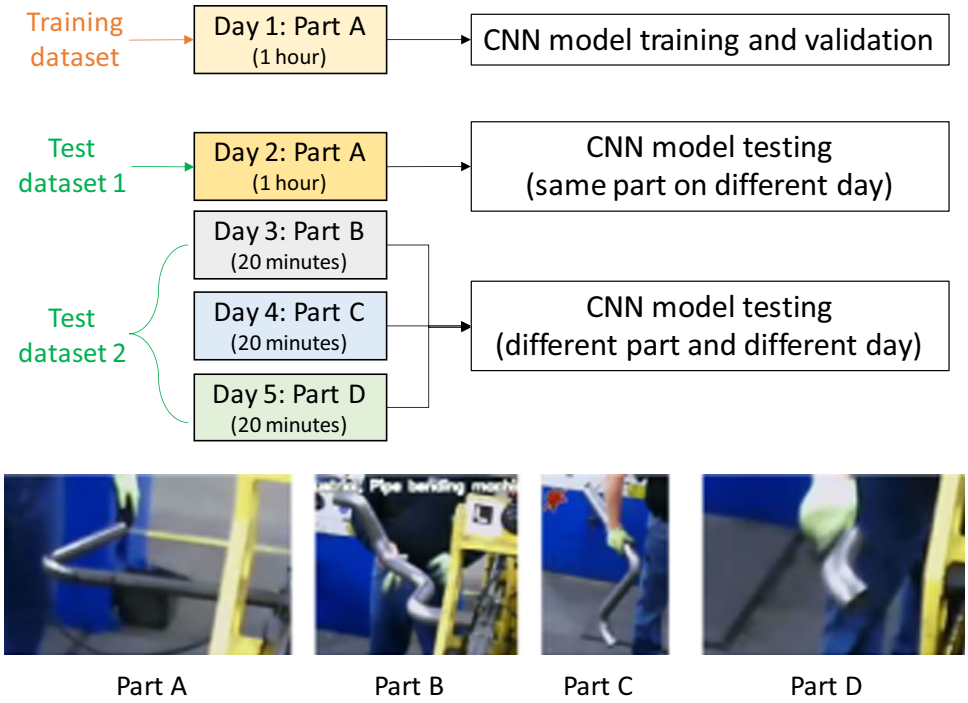

Fig. 6 Configuration of datasets for CNN model training and testing on the top and the captured images of parts on the bottom

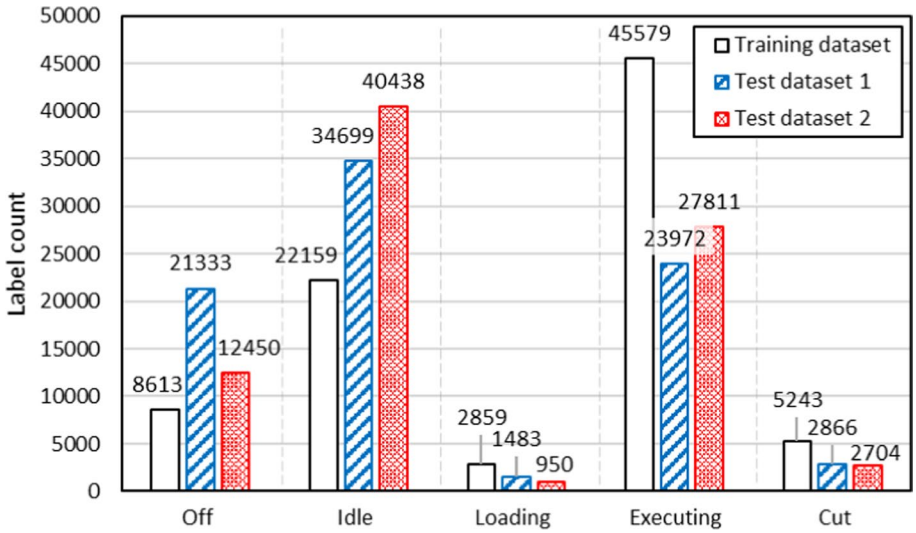

Fig. 7 Label count distribution according to dataset

Table 4 Statistics of cycle time and operational states

Part	Cycle time (s)	Loading (s)	Executing (s)	Cut (s)
A	76.56 ± 0.43	4.51 ± 0.12	14.59 ± 8.94	2.83 ± 0.05
B	128.01 ± 0.14	3.16 ± 0.21	10.85 ± 4.23	2.73 ± 0.03
C	83.40 ± 1.10	2.53 ± 0.17	22.62 ± 10.10	2.51 ± 0.04
D	159.76 ± 0.85	5.39 ± 0.09	25.88 ± 20.2	2.76 ± 0.02

for the CNN model to balance between simplicity and performance. Among various hyperparameter optimization strategies [52–56], grid search was employed in this study to fine-tune the CNN model.

The CNN model architecture, illustrated in Fig. 8, incorporated two convolutional layers, interposed by 2×2 max pooling layers, followed by a fully connected layer, two hidden layers, and an output layer, with the latter being configured to classify five task states. The ReLU (Rectified Linear Unit) activation function was appointed for both the convolutional and hidden layers, while the output layer utilized the softmax activation function, defined in Eq. (2).

$$S(\hat{y}_c) = \frac{\exp(\hat{y}_c)}{\sum_{k=1}^{C} \exp(\hat{y}_k)} \quad (2)$$

where C represents the total number of classes and $\hat{y}_c$ and $\hat{y}_k$ indicate the unnormalized outputs for classes c and k, respectively. The softmax function converts the raw outputs from the preceding layer into probabilities by exponentiating and normalizing them, ensuring them to sum up to one across all classes, thus forming a valid probability distribution.

As in Table 5, a grid search strategy was employed to optimize various hyperparameters across predefined search spaces: the number of filters, kernel sizes for the convolutional layers, and the neuron count for the hidden layers. The minimum and maximum denote the lower and upper bound of hyperparameters, and step size indicates the interval between each step. The steps show the total number of steps. The total number of combinations for grid search was 1024. In the convolutional operations, a stride of 1 and zero-padding were consistently applied. Training was conducted utilizing the categorical cross-entropy loss function L. Training utilized the Adam optimizer, adopting a learning rate of 10^{-4}, with data processed in batches of 64. Initial training was confined to 10 epochs per configuration during the grid search, and upon identifying the most propitious configuration, further training was conducted for an additional 100 epochs to obtain the best model for each sensor case.

Figure 9 illustrates the entire sequence of training and testing CNN model including the grid search for hyperparameter tuning. The dataset necessitated a mindful approach to splitting, considering the pronounced label imbalance prevalent throughout the training data (Day1, Part A) as shown in Fig. 7. To address this, each label category was first subsampled to mirror the count of the least populous label, ensuring equitable representation across all classes. Subsequently, the curated dataset was partitioned into training and validation sets, adhering to an 80–20 percent division

Table 5 Search space range of hyperparameter for grid search

Hyperparameter	Minimum	Maximum	Step size	Steps
Number of filters in 2D Conv 1	8	32	8	4
Kernel size in 2D Conv 1	2×2	3×3	1	2
Number of filters in 2D Conv 2	8	32	8	4
Kernel size in 2D Conv 2	2×2	3×3	1	2
Number of neurons in Hidden 1	16	64	16	4
Number of neurons in Hidden 2	16	64	16	4

scheme, which was randomly implemented. The best model was consequently tested using the full test datasets of the test dataset 1 and 2, ensuring a comprehensive and unbiased evaluation of its predictive capabilities.

Moreover, a comparative analysis of prediction performance against established CNN architectures was conducted to validate the effectiveness of the proposed CNN with the top-performing sensor. Specifically, VGG16, VGG19 [57], YAMNet [58], and ResNet-50 [59] were chosen for comparison. To facilitate training these CNN models, the size of input feature and output were modified to accommodate the data and labels, and the identical training dataset was utilized.

3.3 Prediction Algorithm for Real-Time Implementation

After evaluating the trained models and selecting the best sensor, MTConnect was adopted as middleware to generate sound signal stream and implement the CNN model on a Raspberry Pi in real-time [41]. An MTConnect adapter for a sensor transmits a chunk (2^{11} data points) of sound signals in the space-delimited format continuously to MTConnect agent in *Displacement* representation and *TimeSeries* type of the MTConnect standard [60]. The example of the sound

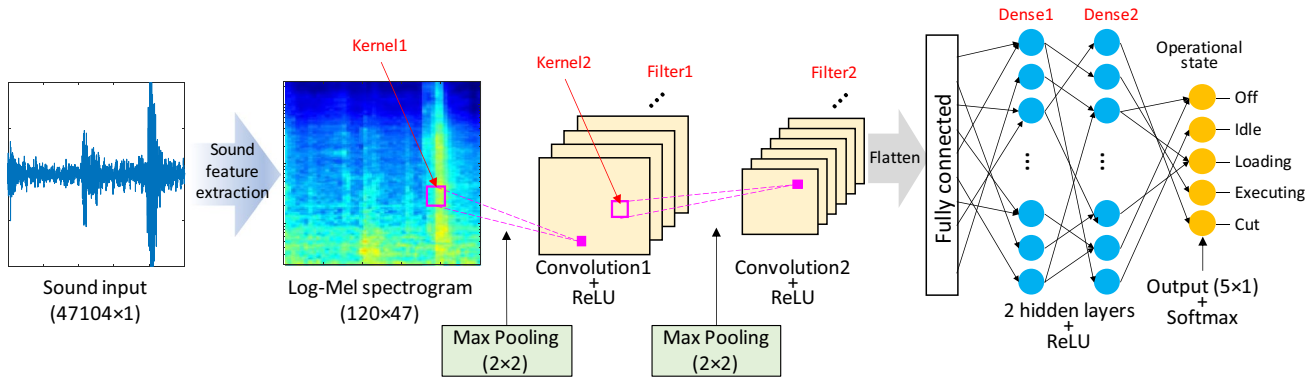

Fig. 8 CNN architecture for operational state sound classification of pipe bending machine

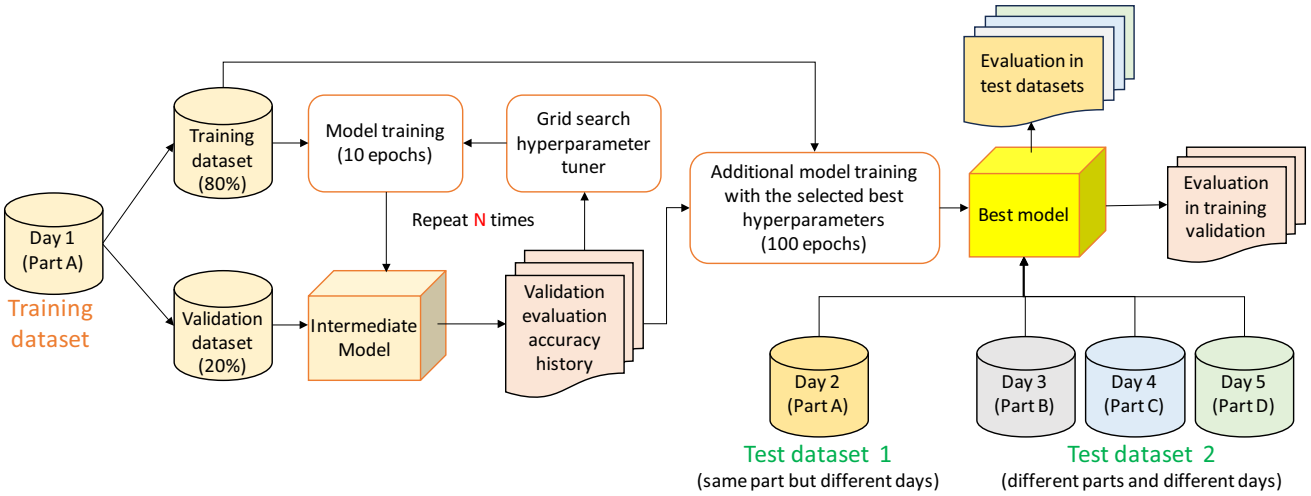

Fig. 9 CNN model training and testing with hyperparameter tuning

data from MTConnect agent is shown in Fig. 10. Each data point is represented by a signed 16-bit integer. The data streamed to the MTConnect agent can be retrieved based on a sequence, allowing the application to request the last 1 second (1-s) of sound data. The steps to retrieve sound signal for the CNN model implementation are as follows in detail.

1. Request using 'current' method to MTConnect agent of sound stream according to MTConnect Rest API. Note that implementation in this research was performed in the Raspberry Pi and the port number is 5000. Therefore, the HTTP request is as below.
   - http://localhost:5000/current
2. Take 'LastSequence' as N from Header of the XML response.
3. Request again for last 1-s sound data using 'sample' method. Note that because sampling rate is 48,000 Hz and the chunk length is 2^{11}, the nearest integer to 1 s is 23.
   - http://localhost:5000/sample?from=N-23&count=23
4. Parse the XML document in ascending order of sequence with converting the space-delimited chunk and then appending to an array ($\boldsymbol{x}$). This ensures that $\boldsymbol{x}$ is always the last 1-s sound signal array.

When applying the CNN model to forecast the operational state of the pipe bending machine, the inference from a model output yields an immediate result for a given input frame of 1-s length. Operational states, labels related to productivity prediction such as 'Loading', and 'Cut' as shown in Table 4, persist for at least 2.5 seconds. An incorrect inference amid the duration of a continuous operational state amplifies prediction errors. For instance, a single false inference during the 'Loading' state can lead to dividing it into two separate 'Loading' counts. To refine the interpretation of CNN model's inference, a buffer algorithm utilizing a queueing method was introduced. Figure 11 demonstrates the interpretation of sequential CNN model inferences with a buffer size of 5. Here, t represents the timestamp at which the CNN model inference occurs, the subscript k is an integer designated the order of the inferences, τ is the prediction interval between inferences. If the prediction interval is shorter than the input frame length (1 second), there is overlap between inferences and it improves the resolution. Since the buffer employs a queue to maintain its size, the first-in element is ejected when the buffer reaches its capacity to enqueue the latest element. The final prediction is ascertained by the majority elements within the buffer. In this scenario, even if an inference 'Executing' is incorrect at

<DisplacementTimeSeries dataItemId="sensor1" timestamp="2023-10-19T12:45:35.394052Z" name="sensor1" sequence="11357" sampleCount="2048" sampleRate="48000">534 -262 -1033 -1119 -631 -446 262 128 299 104 634 1512 1235 1078 502 261 644 809 537 118 -48 -58 223 830 1047 852 287 -322 -697 -830 -715 -647 -588 -325 133 345 516 439 147 -251 -278 596 630 749 680 193 -510 -886 -908 -726 -283 -180 -413 -493 -1453 -1228 680 1192 707 412 245 357 841 1214 1049 -140 -159 -466 -547 -143 427 14 -946 1247 -2514 -2758 196 632 758 68 2139 -215 -258 -879 239 -393 ... </DisplacementTimeSeries>

Fig. 10 Example of sound stream from MTConnect agent

t_{k+1}, the predictive result from the buffer remains 'Loading'. Algorithm 1 describes a buffer algorithm employing a queuing method for continuous sound monitoring and incorporating a custom function to identify the majority element from the buffer array. Here, the input x represents the last one-second length of the sound signals at the moment of request, $\hat{y}$ denotes an inference obtained from a CNN model, and the output $\hat{y}_{buffer}$ denotes the final prediction derived from a queue *buffer* with a maximum length of N. In scenarios without a buffer, where N equals 1, $\hat{y}$ invariably equals the output. The duration required for one loop is signified by the prediction interval τ. It is anticipated that utilizing computationally intensive models will prolong the loop time, thereby compromising the efficacy of the monitoring performance.

Algorithm 1: Continuous sound monitoring

```
while True do
    Input: x ← RetrieveSoundSignal()
    Log-Mel ← ComputeLogMelSpectrogam(x)
    ŷ ← CNN_Model.predict(Log-Mel)
    if buffer size < N then
        Enqueue (ŷ)
    else
        Dequeue()
        Enqueue(ŷ)
    end if
    Output: ŷ_buffer ← FindMajority(buffer)
end while

Function: FindMajority(buffer)
for each unique element e in buffer do
    count[e] ← CountOccurrences(e, buffer)
end for
return element e such that count[e] is maximal
```

4 Results and Discussion

The CNN models were evaluated comprehensively, especially in the context of dealing with datasets that exhibit significant class imbalance among the labels as shown in Fig. 7. Given that the disproportionality among the classes may lead to a biased evaluation when relying solely on accuracy as a performance metric, the additional evaluation metrics was incorporated, namely the macro-averaged precision, recall, and F1-score with accuracy as the performance metrics. Accuracy offers a general view of how often the model is correct across all the classes. However, its limitation, especially in the context of imbalanced data, emanates from its inability to provide specific insights into how well the model performs for each class. That is, a model might still achieve high accuracy by merely predicting the majority class correctly while performing poorly in the minor classes.

Given imbalanced datasets, the macro-averaged metrics become crucial. These metrics calculate the performance for each class independently and then take the average, ensuring all classes are treated equally. Considering C as the total number of classes, TP, TN, FP, and FN denote true positive, true negative, false positive, and false negative counts, respectively. Precision, recall, and F1-score are defined for each class i as follows:

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{3}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{4}$$

$$F1_i = \frac{2 \times Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{5}$$

The macro-averaged precision, recall, and F1-score can be calculated using:

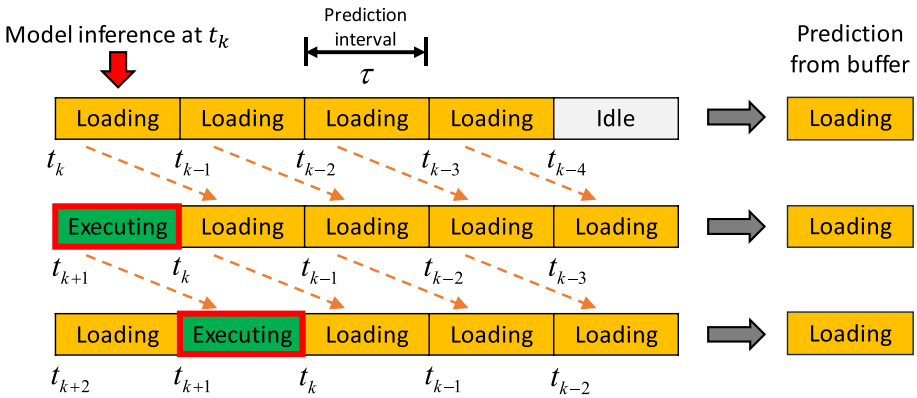

Fig. 11 Interpretation based on selection of the most frequently inferred element from consecutive CNN model predictions using a buffer

$$Precision_{\text{macro}} = \frac{1}{C}\sum_{i=1}^{C} Precision_{\text{i}} \tag{6}$$

$$Recall_{\text{macro}} = \frac{1}{C}\sum_{i=1}^{C} Recall_{\text{i}} \tag{7}$$

$$F1_{\text{macro}} = \frac{1}{C}\sum_{i=1}^{C} F1_{\text{i}} \tag{8}$$

The merit in utilizing macro-averaged metrics lies in their ability to provide an unbiased metric of the model's performance by equally weighing each class, irrespective of their sample size. This impartiality is important because the dataset experiences class imbalance as it avoids the metric being skewed towards the majority class, thereby presenting a more transparent view of how the model performs across all the classes. In the following analysis and throughout the manuscript, macro-averaged performance metrics were employed unless otherwise specified.

4.1 Sensor Selection

Table 6 provides a summary of the optimal hyperparameters for each sensor, determined through grid search. Figure 12 shows the selected models' F1-score comparisons for training and test datasets. Sensor 2 showed the best prediction performance in all training and test datasets. Overall, the CNN models exhibited superior prediction performances on Test dataset 1 compared to Test dataset 2. One plausible explanation for this divergence in performance might be attributed to the nature of the datasets themselves. Test dataset 1 encompasses data from the same part production as the training dataset collected on a different day, thereby potentially sharing similar underlying distribution and patterns. Conversely, Test dataset 2 was derived from a different part production, which might introduce new variances and patterns not present or learned from the training data, causing accurate predictions to be more challenging.

Figure 13 illustrates the F1-scores to evaluate prediction performances for each label using combined test datasets. Among the classes, prediction performances of 'Off' were the best while prediction performances of 'Loading' were the lowest in all sensors. Sensor 2 exhibited the highest performance across most classes except for 'Cut' where Sensor 4 outperformed the others. This could be attributed to the pipe cutting method being shear cutting, driven by the hydraulic pump, which allows Sensor 4 to effectively capture the associated cutting sound from the pump. Nevertheless, Sensor 4 delivered poor prediction performances for 'Idle', 'Loading', and 'Execution' because it struggles to discern distinct sounds during different machine operations. Furthermore, since the hydraulic pump remains active while the machine is

Table 6 Best hyperparameters chosen from grid search

Layer	Parameter	Sensor 1	Sensor 2	Sensor 3	Sensor 4
2D Conv 1	Filter size	16	16	24	24
	Kernel size	3×3	3×3	2×2	3×3
2D Conv 2	Filter size	32	32	32	32
	Kernel size	3×3	3×3	3×3	3×3
Hidden 1	Neuron size	64	64	64	48
Hidden 2	Neuron size	32	64	64	48

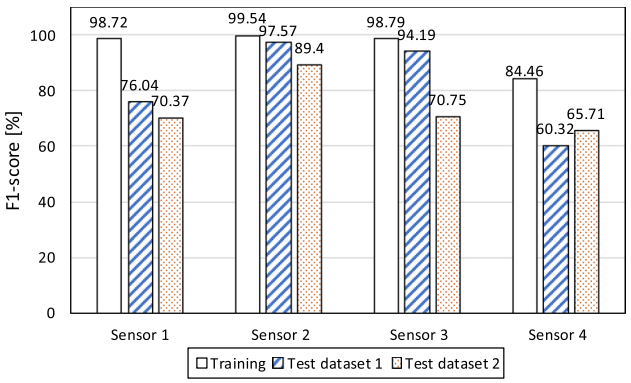

Fig. 12 F1-score comparison of sensor according to training and dataset

turned on, Sensor 4 demonstrated the best prediction performance specifically in predicting 'Off'. If timely operation and precise pipe cutting are pivotal monitoring targets, employing Sensor 4 could be a judicious choice.

Overall, a performance ranking of the sensors was observed: Sensor 2, the internal sound sensor on the machine bed, exhibited the highest performance, followed by Sensor 3, Sensor 1, and Sensor 4. From the result, Sensor 2 was chosen for additional analysis and model implementations, taking advantage of its proven superior predictive capacities across all datasets. This selection is expected to utilize its robust predictive capability and thereby enhance model performance in actual applications.

4.2 Comparison with Other CNN Architectures

Comprehensively analyzing deep learning models, particularly CNN architectures, demands meticulous and multi-pronged approach to accurately capture their efficacy and applicability across various deployment environments. In this section, a comparative analysis was conducted among diverse CNN architectures, examining not merely their computational demand but also their overall size, intrinsic complexity, and predictive performance. FLOPs (floating point operations per second) serve to depict the computational burden of the models, offering insights into the computational resources required during inferential processes [61]. While a lower FLOP count

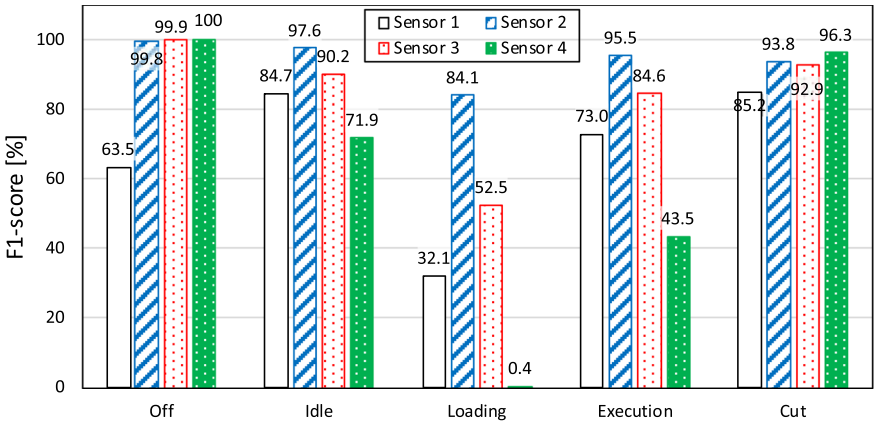

Fig. 13 F1-score of each label on all test datasets according to sensor

typically suggests reduced computational demands, it also provides an estimate of a model's capacity to undertake real-time inference. Further, the physical size of the models, articulated in megabytes (MB), was assessed to determine their storage and memory footprints. This metric is pivotal in ascertaining the viability of model deployment in environments where storage capabilities are constrained such as in edge computers.

Table 7 presents a comparison of various trained CNN models using Sensor 2, evaluating aspects such as training accuracy, complexity, and computational demand. Optimal outcomes characterized by the highest accuracy and the minimal computational load are emphasized in bold. All the examined CNN models demonstrated training accuracies of nearly 99%. While VGG19 secured the top spot in training accuracy, it is notably the largest model with a size exceeding 500 MB. Conversely, the proposed CNN architecture yielded relatively commendable training accuracy with a lightweight model size which is approximately 100 times smaller than that of VGG19.

Prediction performances of the trained CNN architectures using Sensor 2 for the combined test datasets are summarized in Table 8, presenting a detailed insight into the models' capabilities. The bolden value is the best one in the column. While VGG16 exhibited the highest prediction performances across all metrics, the proposed CNN model closely followed, demonstrating only marginal differences in performance outcomes. ResNET50 showed the worst prediction performances across all the metrics. The proposed CNN model manifested an equilibrium among the metrics, not only achieving an impressive accuracy of 97.08% but also securing robust precision, recall, and F1-score, at 94.58%, 93.8%, and 94.19%, respectively. Considering these metrics and computational efficiency, the proposed lightweight CNN model demonstrates comparable predictive performance combined with efficient resource use.

4.3 Real-Time Implementation on Edge Computer

The CNN models were implemented to the same Raspberry Pi with Sensor 2 on the shop floor to evaluate the prediction accuracy and speed of the proposed CNN model in real-time integrated with the buffer algorithm. The inference time $\tau_{\text{inference}}$ of the CNN models was assessed. TensorFlow Lite was employed to execute the CNN models, and the inference time was calculated based on the duration required to obtain the CNN model output, as indicated in line 4 of Algorithm 1. Throughout each iteration of Algorithm 1, the inference time was recorded and stored. To test the computation time in the same environment, each model was loaded for 1 hour long respectively on the same day. Figure 14 illustrates the average inference time results over a 1 hour period of ResNET, YAMNet, and the proposed CNN model. The error bar in Fig. 14 represents the standard deviation. The Raspberry Pi was unable to load VGG16 and VGG19 due to an out-of-memory error encountered during program execution. Even if VGG16 showed the highest accuracy in both training and testing phase, it could not be used for the model deployment on the edge computer. The proposed CNN model demonstrated the quickest speed, at 0.0039 seconds, while ResNET50 and YAMNet recorded approximately 0.173 and 0.017 seconds, respectively. Evidently, the inference time ratio among the models is proportional to several factors such as the number of parameters, model size, or FLOPs shown in Table 7.

Table 7 Comparative analysis of CNN architectures trained using Sensor 2

CNN architecture	Training accuracy (%)	Parameter (million)	FLOP count (million)	Model size (MB)
VGG19	**99.58**	43.12	3946.1	505.5
VGG16	99.06	37.81	3137.3	443.2
ResNET50	98.85	23.59	960.9	227.0
YAMNet	99.51	1.57	60.4	18.5
Proposed CNN	99.55	**0.51**	**4.1**	**5.9**

Table 8 Performance metrics of CNN architectures on test datasets using Sensor 2

CNN architecture	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
VGG19	96.14	91.07	93.09	92.07
VGG16	**97.62**	**95.39**	**94.28**	**94.83**
ResNET50	93.89	86.86	92.64	89.19
YAMNet	97.32	91.24	94.21	92.7
Proposed CNN	97.08	94.58	93.8	94.19

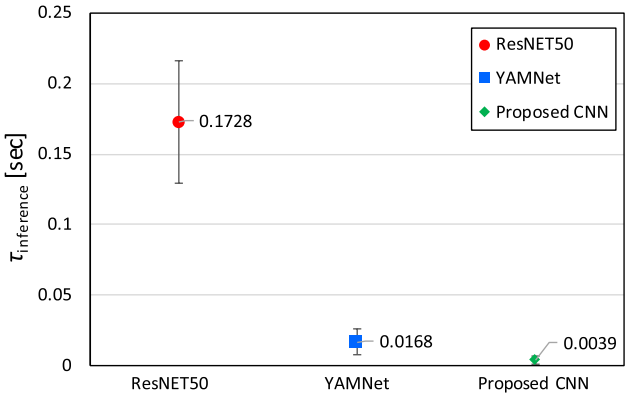

Fig. 14 Inference time according to CNN model on Raspberry Pi

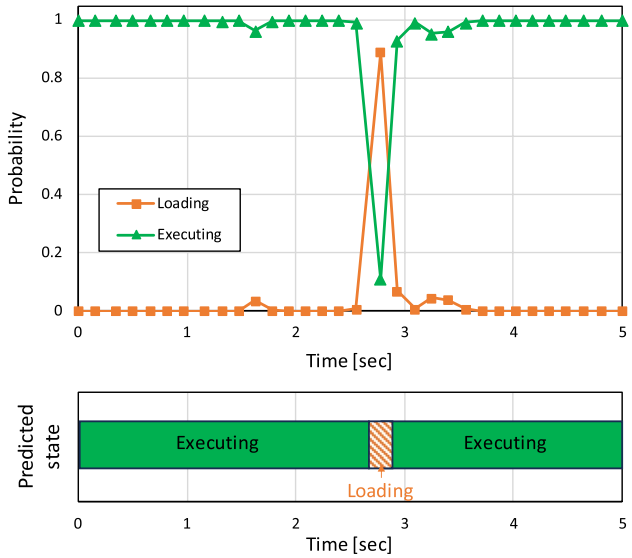

Fig. 15 Inference probability (top) and predicted state (bottom)

While assessing the inference time for the production of another part, distinct from the model training, a substantial number of prediction errors in operation counting were identified. The count for each operation was defined as incremented by one upon a change in the operational state. Figure 15 demonstrates the model output probability (top) and the predicted state (bottom) without the buffer algorithm during 'Executing' operation in all the time range. The raw output values (logits) from the CNN model, denoted as z, are transformed into probabilities through the softmax function, yielding a probability vector $\hat{y}$ represented as Eq. (9).

$$\hat{y} = \text{Softmax}(z) \tag{9}$$

with each element indicating the predicted probability of a class. The softmax function normalizes the raw network outputs to provide values representing probabilities for each class, ensuring the sum of the probabilities in the model's final class prediction equals one. Eventually, the model's inference $\hat{y}$ is obtained by selecting the class with the highest probability. The probabilities depicted in Fig. 15 signify the probabilities associated with labels 'Loading' and 'Executing'. At approximately 2.8 s in Fig. 15, the proposed CNN model incorrectly inferred the 'Executing' state as 'Loading'. While this does not significantly impact the accumulated operation time, it can be crucial when counting the operational state. The count of 'Loading' operations directly correlates with the number of parts produced. Furthermore, the cutting blade for the pipe, being an expensive replacement, has its replacement time determined by the number of cuts. Therefore, it is crucial to count these operations accurately.

The buffer algorithm (Algorithm 1) was implemented using buffer size of 5 alongside the proposed CNN model on the Raspberry Pi. The decision to implement 5 buffers was made to ensure robust detection of operational states while reducing prediction delays. This was chosen to minimize the change of confusion that might arise with an even number of buffers, where conflicting inferences could lead to ambiguous predictions. The choice of 5 buffers effectively balances the necessity for immediate response capabilities with the precision required to accurately capture and respond to brief operational changes, such as those seen in 'Cut' or 'Loading' events. These events, often lasting just a few seconds, demand a buffer length that can quickly process and reflect changes without significant delays. Thus, the selection of five buffers represents a thoughtful compromise, ensuring our monitoring system remains both responsive and accurate in its real-time analysis of operational state changes. During this test, webcam videos were also recorded to retrieve true operational states for the real-world performance of the implementation and comparison. The 'Off' state did not exist in this test. In each iteration of the loop, the timestamp, $\hat{y}$, and

$\hat{y}_{buffer}$ were stored for a duration of 1 hour. Here, $\hat{y}$ represents the model prediction result without applying the algorithm, while $\hat{y}_{buffer}$ denotes the result obtained with the algorithm applied. Table 9 summarizes the results of prediction performances of counting ability and time accuracy in comparisons between the proposed CNN model and the model with the buffer algorithm. The bolden values indicate ones with less error from the true value. Error values are presented in parentheses as percentages. The employment of the buffer algorithm effectively reduces the predictive count discrepancy in all labels. Notably, 'Loading' exhibits a 0% error, indicating accurate prediction with the buffer algorithm. Predictions of the accumulated time on each label are relatively accurate compared to the count predictions, with both models presenting relatively minor error percentages across all the labels. The model employing the buffer algorithm demonstrated a substantial enhancement in predicting count of operational state occurrences.

Finally, the proposed CNN model paired with the buffer algorithm and Sensor 2 was applied for web-based remote monitoring of the legacy pipe bending machine. Figure 16 presents an outline of the monitoring system. MySQL was employed for the database, while the Grafana interface was utilized for the web-based dashboard. In each iteration of the proposed algorithm, the operational state prediction, $\hat{y}_{buffer}$, is transmitted to the database. Data is efficiently stored by writing only the changing operational state result along with the timestamp to the database. Consequently, productivity is calculated by counting operational states, and operational time-related metrics such as runtime and downtime are monitored by measuring the timestamps between changes in the operational state. Figure 17 is the capture of the web-based dashboard includes a discrete time panel for state history change, cut and loading counts, and executing cycle.

Following the presentation of the real-time implementation on an edge computer, it is crucial to acknowledge a limitation regarding long-term reliability. Continual learning for long-term reliability is an important topic that this research did not focus on, but it deserves more attention in the future. The necessity for models to adapt over time, particularly in response to factors such as equipment wear and the introduction of new operational conditions, highlights the importance of continuous learning strategies. Looking forward, the discussion on enabling continual learning within this monitoring system is relevant. This could involve developing mechanisms for incremental model updates, where the system regularly integrates new data, learning from evolving operational patterns without the need for complete retraining [62]. Additionally, exploring techniques such as transfer learning, where a pretrained model is fine-tuned with new data, could prove valuable for efficiently adapting to changes in the operational environment [63]. Future research in these areas will be

Table 9 Prediction performance comparisons in real implementation

Metric	Label	True value	Proposed CNN model only	Proposed CNN model with buffer algorithm
Count	Idle	41	91 (121.9%)	**50 (22.0%)**
	Loading	17	29 (70.6%)	**17 (0%)**
	Executing	112	215 (92.0%)	**120 (3.57%)**
	Cut	66	112 (69.7%)	**67 (1.52%)**
Accumulated time [sec]	Idle	834.5	**848.4 (1.73%)**	850.9 (2.04%)
	Loading	88.3	**86.1 (− 3.43%)**	83.32 (− 6.59%)
	Executing	2495.2	**2489.4 (− 0.16%)**	2490.9 (− 0.209%)
	Cut	182.7	176.6 (− 3.26%)	**175.5 (− 2.46%)**

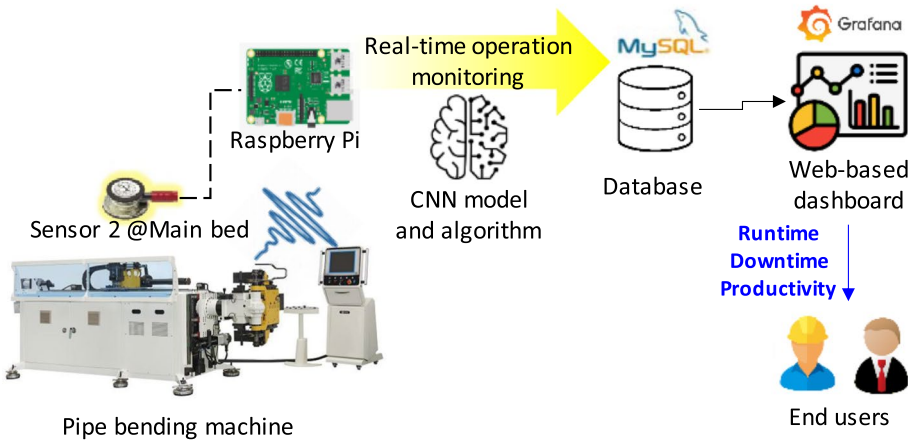

Fig. 16 Schematic of real-time remote monitoring for pipe bending machine

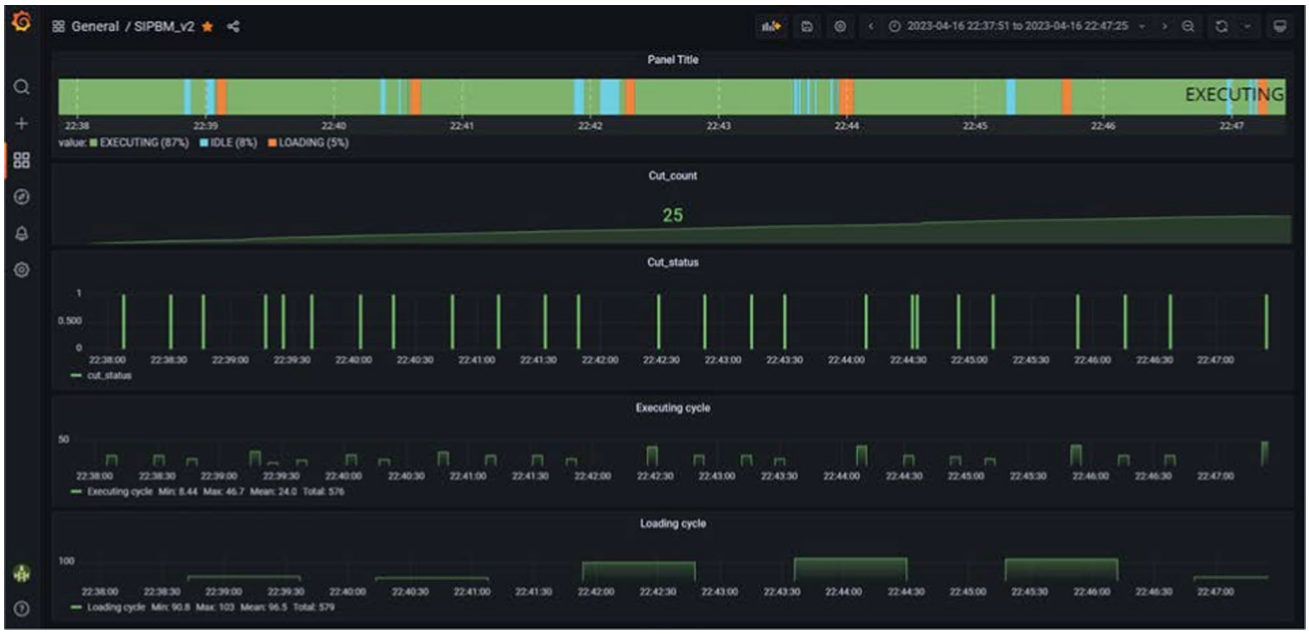

Fig. 17 Capture of web-based dashboard

crucial for advancing the adaptability and sustainability for machine learning models in industrial monitoring, ensuring they remain robust and effective over extended periods.

5 Conclusion

This study introduced a real-time sound monitoring technique employing a lightweight CNN model to monitor operation and productivity of a legacy pipe bending machine on a real shop floor. Initially, four sensors were deployed to determine the optimal sensor type and placement, collecting sound data alongside webcam videos to generate labels for training the CNN model. Various datasets gathered from different production phases and durations were labeled, then utilized to train and test the lightweight CNN model. The model leverages Log-Mel spectrogram for feature extraction and employs a grid search method to optimize hyperparameters across two convolutional and two hidden layers. Amongst training and testing datasets, a CNN model, utilizing an internal sound sensor located on the main bed, exhibited superior prediction performances with accuracies of 99.55% and 97.07%, respectively.

The proposed CNN model was compared with state-of-the-art DL architectures, such as VGG16, VGG19, ResNET50, and YAMNet, to discern both prediction performance and efficacy in edge computing. While VGG16 exhibited the highest prediction performance with a 97.62% accuracy on testing datasets, VGG16 and VGG19 were not implemented on the edge computer due to their substantial size, inhibiting their applicability on Raspberry Pi. Moreover, the inference time of the proposed CNN model on Raspberry Pi was measured, revealing an average inference time of 3.9 ms, the shortest amongst the compared architectures. However, during real-world application, the model encountered a significant number of errors in predicting operational state occurrences. To mitigate this, a buffer algorithm was introduced to enhance count performance. A queuing method was proposed for continuous sound monitoring, effectively mitigated predictive count discrepancies in operational states. This is particularly vital for accurately counting loading and cutting operations, which directly correlate with the number of parts produced, thereby ensuring precise monitoring and operational efficiency in the manufacturing process. In conjunction with the buffer algorithm, the proposed lightweight CNN model was successfully deployed on the edge computer for remote monitoring of the machine in real-time.

Future work will delve deeper into the buffer algorithm and related techniques to enhance the robustness and prediction performances of CNN models in sound monitoring. A key focus will also be on exploring continual learning methods to ensure the system adapts and evolves with changing operational conditions, aiming to maintain the long-term effectiveness of our monitoring solutions. Additionally, the scope of sound monitoring and recognition for legacy machines will be expanded to encompass condition-based monitoring (CbM).

Appendix

See Fig. 18.

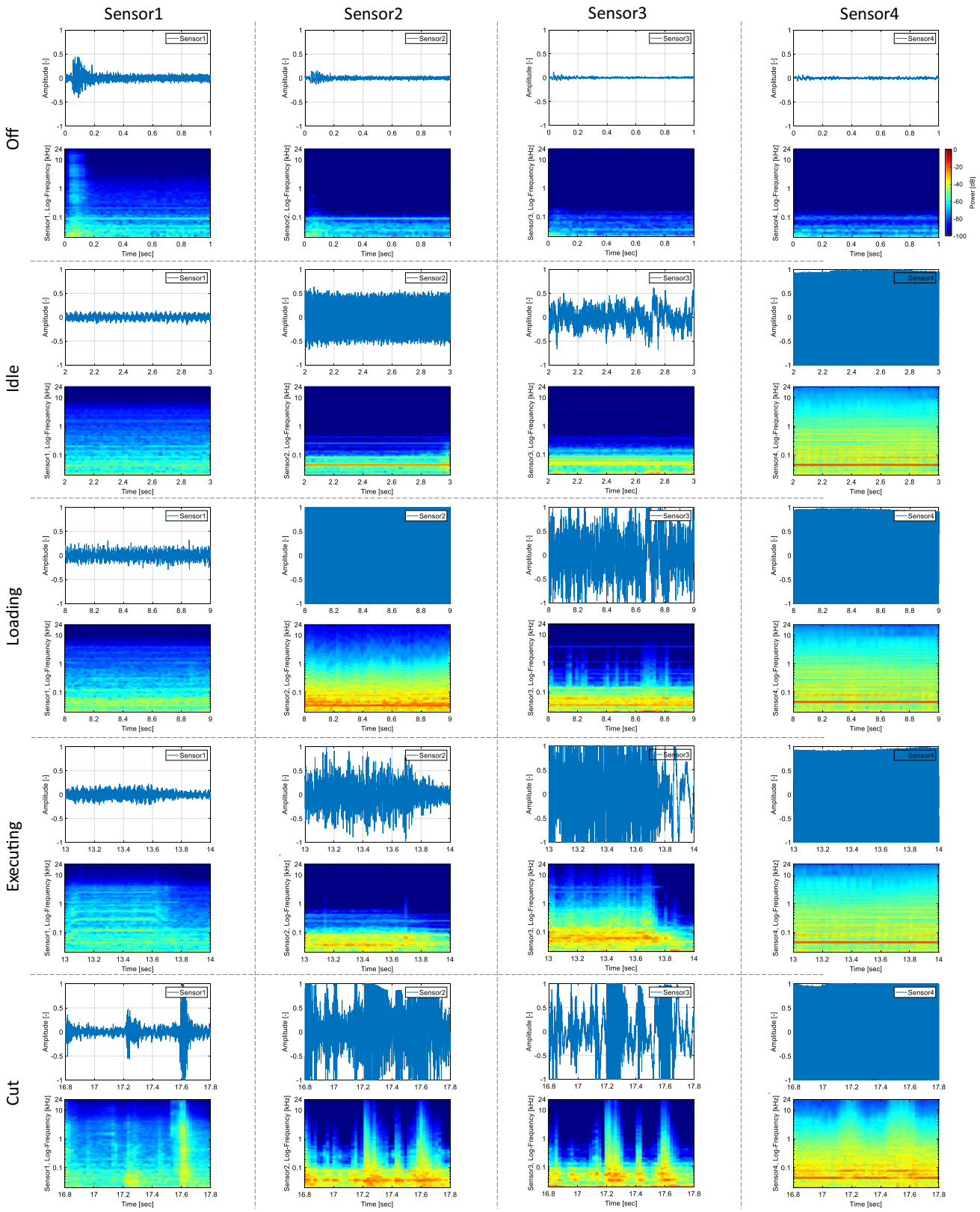

Fig. 18 Sound signal in time domain and sound feature of each operational state (row) and sensor (column) in grid: Each cell contains time domain plot (top) and Log-Mel spectrogram (bottom) and the colormap throughout all Log-Mel spectrograms is the same with it in Off case of Sensor 4. The time axis is the same as in Fig. 4

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12541-024-01018-3.

Acknowledgements This research was supported by Wabash Heartland Innovation Network (WHIN), Indiana Next Generation Manufacturing Competitiveness Center (IN-MaC), and Standard Industrial Inc. The authors also acknowledge support from National Science Foundation under Grant No. 2134667 "FMRG: Manufacturing USA: Cyber: Privacy-Preserving Tiny Machine Learning Edge Analytics to Enable AI-Commons for Secure Manufacturing," and AnalytiXIN through CICP.

Funding Open Access funding enabled and organized by KAIST.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Ren, S., Zhang, Y., Sakao, T., Liu, Y., & Cai, R. (2022). An advanced operation mode with product-service system using lifecycle big data and deep learning. *International Journal of Precision Engineering and Manufacturing-Green Technology, 9*(1), 287–303.
2. Jeong, Y. (2023). Digitalization in production logistics: How AI, digital twins, and simulation are driving the shift from model-based to data-driven approaches. *International Journal of Precision Engineering and Manufacturing-Smart Technology, 1*(2), 187–200. https://doi.org/10.57062/ijpem-st.2023.0052
3. Alqoud, A., Schaefer, D., & Milisavljevic-Syed, J. (2022). Industry 4.0: A review of digital retrofitting solutions for legacy manufacturing systems. In *Advances in manufacturing technology XXXV*. https://doi.org/10.3233/atde220557
4. Alias, C., Salewski, U., Ortiz Ruiz, V. E., Alarcón Olalla, F. E., Neirão Reymão, J. d. E., & Noche, B. (2017). Adapting warehouse management systems to the requirements of the evolving era of industry 4.0. In *International manufacturing science and engineering conference.*
5. Contreras Pérez, J. D., Cano Buitrón, R. E., & García Melo, J. I. (2018). Methodology for the retrofitting of manufacturing resources for migration of SME towards industry 4.0. In *Applied informatics* (pp. 337–351). https://doi.org/10.1007/978-3-030-01535-0_25
6. Deshpande, A., & Pieper, R. (2011). Legacy machine monitoring using power signal analysis. In *International manufacturing science and engineering conference*.
7. Maeda, M., Sakurai, Y., Tamaki, T., & Nonaka, Y. (2017). Method for automatically recognizing various operation statuses of legacy machines. *Procedia CIRP, 63*, 418–423. https://doi.org/10.1016/j.procir.2017.03.150
8. Trembley, D. K., Haghnegahdar, L., & Wang, Y. (2018). A survey of advanced manufacturing with legacy machinery: The Internet of Other Things. In *Proceedings of the 2018 IISE Annual Conference*.
9. Matt, D. T., Modrák, V., & Zsifkovits, H. (2020). Industry 4.0 for SMEs: Challenges, opportunities and requirements. https://doi.org/10.1007/978-3-030-25425-4
10. Kolla, S. S. V. K., Lourenço, D. M., Kumar, A. A., & Plapper, P. (2022). Retrofitting of legacy machines in the context of Industrial Internet of Things (IIoT). *Procedia Computer Science, 200*, 62–70.
11. Guerreiro, B. V., Lins, R. G., Sun, J., & Schmitt, R. (2018). Definition of smart retrofitting: First steps for a company to deploy aspects of industry 4.0. In *Advances in manufacturing* (pp. 161–170). https://doi.org/10.1007/978-3-319-68619-6_16
12. Moeuf, A., Tamayo, S., Lamouri, S., Pellerin, R., & Lelievre, A. (2016). Strengths and weaknesses of small and medium sized enterprises regarding the implementation of lean manufacturing. *IFAC-PapersOnLine, 49*(12), 71–76.
13. Picaut, J., Can, A., Fortin, N., Ardouin, J., & Lagrange, M. (2020). Low-cost sensors for urban noise monitoring networks: A literature review. *Sensors (Basel)*. https://doi.org/10.3390/s20082256
14. Mittal, S., Khan, M. A., Purohit, J. K., Menon, K., Romero, D., & Wuest, T. (2019). A smart manufacturing adoption framework for SMEs. *International Journal of Production Research, 58*(5), 1555–1573. https://doi.org/10.1080/00207543.2019.1661540
15. Wang, L. (2013). Machine availability monitoring and machining process planning towards Cloud manufacturing. *CIRP Journal of Manufacturing Science and Technology, 6*(4), 263–273. https://doi.org/10.1016/j.cirpj.2013.07.001
16. Tedeschi, S., Emmanouilidis, C., Farnsworth, M., Mehnen, J., & Roy, R. (2017). New threats for old manufacturing problems: secure IoT-enabled monitoring of legacy production machinery. In *Advances in production management systems. The path to intelligent, collaborative and sustainable manufacturing* (pp. 391–398). https://doi.org/10.1007/978-3-319-66923-6_46
17. Selvaraj, V., Xu, Z., & Min, S. (2023). Intelligent operation monitoring of an ultra-precision CNC machine tool using energy data. *International Journal of Precision Engineering and Manufacturing-Green Technology, 10*(1), 59–69.
18. Selvaraj, V., & Min, S. (2023). Real-time fault identification system for a retrofitted ultra-precision CNC machine from equipment's power consumption data: A case study of an implementation. *International Journal of Precision Engineering and Manufacturing-Green Technology, 10*(4), 925–941.
19. Alahmari, A., & Duncan, B. (2020). Cybersecurity risk management in small and medium-sized enterprises: A systematic review of recent evidence. In *2020 international conference on cyber situational awareness, data analytics and assessment (CyberSA)*.
20. van Haastrecht, M., Sarhan, I., Shojaifar, A., Baumgartner, L., Mallouli, W., & Spruit, M. (2021). A threat-based cybersecurity risk assessment approach addressing SME needs. In *The 16th international conference on availability, reliability and security*.
21. Deshpande, A. M., Telikicherla, A. K., Jakkali, V., Wickelhaus, D. A., Kumar, M., & Anand, S. (2020). Computer vision toolkit for non-invasive monitoring of factory floor artifacts. *Procedia Manufacturing, 48*, 1020–1028.
22. Kim, H., Jung, W. K., Choi, I. G., & Ahn, S. H. (2019). A low-cost vision-based monitoring of computer numerical control (CNC) machine tools for small and medium-sized enterprises (SMEs). *Sensors (Basel)*. https://doi.org/10.3390/s19204506
23. Ren, Z., Fang, F., Yan, N., & Wu, Y. (2022). State of the art in defect detection based on machine vision. *International Journal*

of Precision Engineering and Manufacturing-Green Technology, 9(2), 661–691.
24. Lou, P., Li, J., Zeng, Y., Chen, B., & Zhang, X. (2022). Real-time monitoring for manual operations with machine vision in smart manufacturing. *Journal of Manufacturing Systems, 65*, 709–719. https://doi.org/10.1016/j.jmsy.2022.10.015
25. Maeno, K., Nagahara, H., Shimada, A., & Taniguchi, R.-I. (2013). Light field distortion feature for transparent object recognition. In *2013 IEEE conference on computer vision and pattern recognition*.
26. Mukherjee, R., Bessa, M., Melo-Pinto, P., & Chalmers, A. (2021). Object detection under challenging lighting conditions using high dynamic range imagery. *IEEE Access, 9*, 77771–77783. https://doi.org/10.1109/access.2021.3082293
27. Caso, E., Fernandez-del-Rincon, A., Garcia, P., Iglesias, M., & Viadero, F. (2020). Monitoring of misalignment in low speed geared shafts with acoustic emission sensors. *Applied Acoustics*. https://doi.org/10.1016/j.apacoust.2019.107092
28. Wu, H., Wang, Y., & Yu, Z. (2015). In situ monitoring of FDM machine condition via acoustic emission. *The International Journal of Advanced Manufacturing Technology*. https://doi.org/10.1007/s00170-015-7809-4
29. Kuntoğlu, M., Salur, E., Gupta, M. K., Sarıkaya, M., & Pimenov, D. Y. (2021). A state-of-the-art review on sensors and signal processing systems in mechanical machining processes. *The International Journal of Advanced Manufacturing Technology, 116*(9–10), 2711–2735. https://doi.org/10.1007/s00170-021-07425-4
30. Ooi, B.-Y., Beh, W.-L., Lee, W.-K., & Shirmohammadi, S. (2020). A parameter-free vibration analysis solution for legacy manufacturing machines' operation tracking. *IEEE Internet of Things Journal, 7*(11), 11092–11102. https://doi.org/10.1109/jiot.2020.2994395
31. Grimmelsman, K. A., & Zolghadri, N. (2020). Experimental evaluation of low-cost accelerometers for dynamic characterization of bridges. In *Dynamics of civil structures, Volume 2* (pp. 145–152). https://doi.org/10.1007/978-3-030-12115-0_19
32. Grimmelsman, K. (2022). Investigation of low-cost accelerometer performance for vibration analysis of bridges. In *Dynamics of civil structures, Volume 2* (pp. 129–137). https://doi.org/10.1007/978-3-030-77143-0_13
33. Park, D., Kim, S., An, Y., & Jung, J. Y. (2018). LiReD: A lightweight real-time fault detection system for edge computing using LSTM recurrent neural networks. *Sensors (Basel)*. https://doi.org/10.3390/s18072110
34. Xu, T., Xu, X., Xu, D., Zou, Z., & Zhao, H. (2021). Low-cost and efficient thermal calibration scheme for MEMS triaxial accelerometer. *IEEE Transactions on Instrumentation and Measurement, 70*, 1–9. https://doi.org/10.1109/tim.2021.3096290
35. Martinez, J., Asiain, D., & Beltran, J. R. (2022). Self-calibration technique with lightweight algorithm for thermal drift compensation in MEMS accelerometers. *Micromachines (Basel)*. https://doi.org/10.3390/mi13040584
36. Ooi, B.-Y., Lim, J. J.-W., Lee, W.-K., & Shirmohammadi, S. (2020). Non-intrusive operation status tracking for legacy machines via sound recognition. In *2020 IEEE international instrumentation and measurement technology conference (I2MTC)*.
37. Grumiaux, P. A., Kitic, S., Girin, L., & Guerin, A. (2022). A survey of sound source localization with deep learning methods. *Journal of the Acoustical Society of America, 152*(1), 107. https://doi.org/10.1121/10.0011809
38. Yun, H., Kim, H., Kim, E., & Jun, M. B. (2020). Development of internal sound sensor using stethoscope and its applications for machine monitoring. *Procedia Manufacturing, 48*, 1072–1078.
39. Kim, E., Yun, H., Jun, M. B.-G., Kim, K., & Cha, S. W. (2021). Multiple sound sensors and fusion in modern CNN-based machine state prediction. In *International manufacturing science and engineering conference*.
40. Yun, H., Kim, H., Jeong, Y. H., & Jun, M. B. G. (2021). Autoencoder-based anomaly detection of industrial robot arm using stethoscope based internal sound sensor. *Journal of Intelligent Manufacturing, 34*(3), 1427–1444. https://doi.org/10.1007/s10845-021-01862-4
41. Kim, E., Yun, H., Araujo, O. C., & Jun, M. B. G. (2023). Sound recognition based on convolutional neural network for real-time cutting state monitoring of tube cutting machine. *International Journal of Precision Engineering and Manufacturing-Smart Technology, 1*(1), 1–18. https://doi.org/10.57062/ijpem-st.2022.0038
42. Zhang, T., Ding, B., Zhao, X., Liu, G., & Pang, Z. (2021). LearningADD: Machine learning based acoustic defect detection in factory automation. *Journal of Manufacturing Systems, 60*, 48–58. https://doi.org/10.1016/j.jmsy.2021.04.005
43. Li, H., Ota, K., & Dong, M. (2018). Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE Network, 32*(1), 96–101. https://doi.org/10.1109/mnet.2018.1700202
44. Lee, J., Chua, P. C., Chen, L., Ng, P. H. N., Kim, Y., Wu, Q., Jeon, S., Jung, J., Chang, S., & Moon, S. K. (2023). Key enabling technologies for smart factory in automotive industry: Status and applications. *International Journal of Precision Engineering and Manufacturing-Smart Technology, 1*(1), 93–105.
45. Jun, M. B. G., Yun, H., & Kim, E. (2021). Human expertise inspired smart sensing and manufacturing. In *2021 International conference on electronics, communications and information technology (ICECIT)*.
46. Kim, J., Lee, H., Jeong, S., & Ahn, S.-H. (2021). Sound-based remote real-time multi-device operational monitoring system using a Convolutional Neural Network (CNN). *Journal of Manufacturing Systems, 58*, 431–441. https://doi.org/10.1016/j.jmsy.2020.12.020
47. Dayal, A., Yeduri, S. R., Koduru, B. H., Jaiswal, R. K., Soumya, J., Srinivas, M. B., Pandey, O. J., & Cenkeramaddi, L. R. (2022). Lightweight deep convolutional neural network for background sound classification in speech signals. *Journal of the Acoustical Society of America, 151*(4), 2773. https://doi.org/10.1121/10.0010257
48. Rabiner, L. R., & Schafer, R. W. (2007). Introduction to digital speech processing. *Foundations and Trends® in Signal Processing, 1*(1–2), 1–194.
49. Solanki, A., & Pandey, S. (2019). Music instrument recognition using deep convolutional neural networks. *International Journal of Information Technology, 14*(3), 1659–1668. https://doi.org/10.1007/s41870-019-00285-y
50. Stern, R. M., Acero, A., Liu, F.-H., & Ohshima, Y. (1996). Signal processing for robust speech recognition. In *Automatic speech and speaker recognition: Advanced topics* (pp. 357–384). Springer.
51. Li, J., Dai, W., Metze, F., Qu, S., & Das, S. (2017). A comparison of deep learning methods for environmental sound detection. In *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*.
52. Injadat, M., Salo, F., Nassif, A. B., Essex, A., & Shami, A. (2018). Bayesian optimization with machine learning algorithms towards anomaly detection. In *2018 IEEE global communications conference (GLOBECOM)*.
53. Toma, R. N., Prosvirin, A. E., & Kim, J. M. (2020). Bearing fault diagnosis of induction motors using a genetic algorithm and machine learning classifiers. *Sensors (Basel)*. https://doi.org/10.3390/s20071884
54. Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural Computation, 12*(8), 1889–1900.
55. Belete, D. M., & Huchaiah, M. D. (2021). Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results. *International Journal of Computers and Applications, 44*(9), 875–886. https://doi.org/10.1080/1206212x.2021.1974663
56. Bergstra, J., & Bengio, Y. (2012). Random search for hyperparameter optimization. *Journal of Machine Learning Research, 13*(2), 281–305.

57. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint* arXiv:1409.1556.
58. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint* arXiv:1704.04861.
59. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
60. MTConnect Institute. (2018). MTConnect Standard. In *ANSI/MTC1. 4-2018*. 2018: MTConnect Institute.
61. Kumar, A., Sharma, A., Bharti, V., Singh, A. K., Singh, S. K., & Saxena, S. (2021). MobiHisNet: A lightweight CNN in mobile edge computing for histopathological image classification. *IEEE Internet of Things Journal, 8*(24), 17778–17789. https://doi.org/10.1109/jiot.2021.3119520
62. Zhang, C., Song, N., Lin, G., Zheng, Y., Pan, P., & Xu, Y. (2021). Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.*
63. Liao, Y., Ragai, I., Huang, Z., & Kerner, S. (2021). Manufacturing process monitoring using time-frequency representation and transfer learning of deep neural networks. *Journal of Manufacturing Processes, 68*, 231–248.

Eunseob Kim is a Ph.D. student in the School of Mechanical Engineering at Purdue University, IN, USA. He received his BS degree in Mechanical Engineering from Gyeongsang National University, Korea in 2013, and his M.S. degree in Mechanical and Aerospace Engineering from Seoul National University, Korea in 2016. His research interests include smart monitoring, sound recognition, and artificial intelligence application for manufacturing.

Daeseong Mun graduated with M.S. degree in Mechanical Engineering at Purdue University, West Lafayette, IN, USA. He received his B.S. degree in Mechanical System Engineering from Korea Military Academy. He currently works for Repulbic of Korea Army. His research interests include Internet of Things (IoT) application for manufacturing, predictive maintenance, and machine learning for machine monitoring.

Martin Byung-Guk Jun is an Associate Professor of the School of Mechanical Engineering at Purdue University, West Lafayette, IN, USA. Prior to joining Purdue University, he was an Associate Professor at the University of Victoria, Canada. He received the B.Sc. and M.A.Sc. degrees in Mechanical Engineering from the University of British Columbia, Vancouver, Canada in 1998 and 2000, respectively. He then received his Ph.D. degree in 2005 from the University of Illinois at Urbana-Champaign in the Department of Mechanical Science and Engineering. His main research focus is on advanced multi-scale and smart manufacturing processes and technologies for various applications. His sound-based smart machine monitoring technology led to a start-up company on smart sensing. He has authored over 140 peer-reviewed journal publications. He is an ASME fellow and Associate Editor of Journal of Manufacturing Processes. He is also the recipient of the 2011 SME Outstanding Young Manufacturing Engineer Award, 2012 Canadian Society of Mechanical Engineers I.W. Smith Award for Outstanding Achievements, and 2015 Korean Society of Manufacturing Technology Engineers Damwoo Award.

Huitaek Yun is an assistant professor in the department of mechanical engineering, Korea Advanced Institute of Science and Technology (KAIST). In 2021, he received a Ph.D. degree in the school of mechanical engineering, Purdue university. He is interested in smart manufacturing to combine manufacturing processes and systems with information technologies: machine connectivity in cyber-physical systems (CPS), robotic manufacturing via human-robot collaboration, mixed reality-based human machine interface, data analysis from Internet of Things (IoT) sensors, and development of cyber systems like digital twin and artificial intelligence (AI) for self-decision making.