

Abstract—On pig farms, many piglets die because they are crushed when sows roll from side to side or lie down. On average, 1.2 piglets are crushed by sows every day. To resolve the piglet mortality issue, this article proposes PigTalk, an artificial intelligence (AI)-based Internet of Things (IoT) platform for detecting and mitigating piglet crushing. Through real-time analysis of the voice data collected in a farrowing house, PigTalk detects if any piglet screaming occurs, and automatically activates sow-alert actuators for emergency handling of the crushing event. We propose an audio clip transform approach to pre-process the raw voice data, and utilizes min-max scaling in machine learning (ML) to detect piglet screams. In our first contribution, the above data preprocessing method together with subtle parameter setups of the machine learning model improve the piglet scream detection accuracy up to 99.4%, which is better than the previous solutions (up to 92.8%). In our second contribution, we show how to design two cyber IoT devices, i.e., DataBank for data pre-processing and MLDevice for real-time AI to automatically trigger actuators such as floor vibration and water drop to force a sow to stand up. We conduct analytic analysis and simulation to investigate how the detection delay affects the critical time period to save crushed piglets. Our study indicates that PigTalk can save piglets within 0.05 s with 99.93% of the successful rate. Such results are validated in a commercial farrowing house. PigTalk is a new approach that automatically mitigates piglet crushing, which could not be achieved in the past.

Index Terms—Convolutional neural network (CNN), crushing, Internet of Things (IoT), machine learning (ML), piglet, scream detection.

Manuscript received February 21, 2020; revised June 9, 2020 and June 29, 2020; accepted July 16, 2020. Date of publication August 20, 2020; date of current version March 5, 2021. This work was supported in part by the Center for Open Intelligent Connectivity from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education in Taiwan, and in part by the Ministry of Science and Technology 108-2221-E-009-047, 107-2221-E-197-006-MY3, and 108-2321-B-197-003 Ministry of Economic Affairs 107-EC-17-A-02-S5-007. Paper no. TII-20-0908. (Corresponding author: Yi-Bing Lin.)

Whai-En Chen and Li-Xian Chen are with the Department of Computer Science and Information Engineering, Yuan 26047, Taiwan (e-mail: wechen@nru.edu.tw; b0543017@ems.niu.edu.tw).

Yi-Bing Lin is with the Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: liny@cs.nctu.edu.tw).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.3012496

I. INTRODUCTION

In the pig farming business, piglet mortality is a serious issue that needs to be carefully addressed. In particular, 7.5% of piglets are accidentally killed by sows in the first three days [1]. On average, 1.2 piglets are crushed by sows every day [2]. Specifically, piglet crushing may occur when a sow is lying down or turning its body over while lying [3]. Such accidents occur to weaker piglets more often than normal ones because weaker piglets are more likely to stay near their mother sows for sucking [4]. Also, piglets stay closer to their mothers to keep warm when the farrowing house is cold [5]. Therefore, to prevent piglet crushing, we should keep the piglets away from hunger and cold. Alternatively, farrowing cages are used to restrict the posture changes of sows, and provide more safe space to piglets [6].

When a piglet is crushed, the hog farmer must quickly take action before it is too late. A skillful farmer can detect piglet crushing by scream vocalization [7], and force a sow to stand up or separate crushed piglet from the sow. However, monitoring farrowing cages is a 24-h job for a hog farmer, and the cost of labor is too high.

Through the Internet of Things (IoT) technology [8], [24], we can collect pig vocalization data from microphone sensors and automatically take actions through actuators when a piglet crushing event occurs. Based on an IoT device management platform called IoTalk [8], this article proposes PigTalk to resolve the piglet crushing problem.

Several methods [7], [9] were proposed to automatically analyze piglet’s vocalizations. The study in [9] investigated the relationship between a piglet’s distress vocalizations and its age, weight and health status. In [7], directional microphone and webcam are used to record vocalizations and video for labeling in scream detection. The prediction accuracy was reported to be 32.1%, which is too low to be useful. These methods can be improved through artificial intelligence (AI) driven mechanisms [23]. Three AI related notations used in this article are Mel frequency Cepstral coefficient (MFCC), min-max scaling, and convolutional neural network (CNN).

1) The MFCCs of a signal are a small number of features (typically 20 features) that concisely describe the overall shape of a spectral envelope. MFCC is often used to describe timbre and speech recognition in information retrieval of vocalizations.

2) In min-max Scaling, the samples are scaled within a fixed range so that the standard deviation is smaller to suppress the effect of outliers.
3) A CNN is a class of deep neural networks, which is most commonly used to analyzing visual images. CNN can automatically extract the features of piglet vocalizations, and is robust and fault tolerant to environment noises. Details of CNN will be elaborated in Section III-B. Several machine learning approaches were proposed for vocalization analysis. The study in [10] utilizes MFCC to extract the frequency features of vocalizations and performs CNN to automatically classify heart sound for pathology detection. The prediction accuracy reaches 81.5%. Similar study was conducted to detect infant cry [11] and the prediction accuracy can be up to 92.8%. In [12], vocalizations are transformed into spectrograms, and eight indicators are extracted from the spectrograms as the features for the K-means clustering method to classify different piglet’s vocalizations. The accuracy for identifying the piglet crushing events was reported to be 88%. None of the above studies discussed how to automatically trigger actuators in real-time to take actions for the piglet crushing events and stopped actuators for false alarms. Based on our study conducted in a farrowing house (see Fig. 1), this article improves the previous studies in the following aspects.

1) Similar to the studies in [10] and [11], PigTalk uses CNN to interpret vocalizations from farrowing cages in real time to detect piglet screaming. Similar to [12], spectrogram transformation is used in PigTalk. On the other hand, we propose a novel audio clip transform approach to preprocess the raw voice data, which utilizes min-max scaling to detect piglet screaming. The above novel data preprocessing method together with subtle parameter setups for machine learning model tuning improve the piglet scream detection accuracy up to 99.4%. The best performance of previous approaches for infant cry is 92.8%.

2) PigTalk is more advanced than the previous solutions in [7] and [10]–[12] because PigTalk can automatically trigger sow-alert actuators to stop piglet crushing by integrating AI with IoT in real time, which cannot be achieved by the previous IoT solutions. In other words, the existing solutions still need to push a sow to stand up manually. Our achievement is due to the design of two cyber IoT devices DataBank and ML_device [see Fig. 2 (n) and (o) to be elaborated in Section II]. The DataBank device is responsible for generating the enhanced spectrograms of vocalizations collected from farrowing cages. The ML_device device implements the CNN machine learning model. By tightly integrating IoT and AI, PigTalk can save crushed piglets within 0.05 seconds with 99.93% of the successful rate. No previous solutions have this kind of mechanism and performance.

The rest of this article is organized as follows. Section II describes the PigTalk platform. Section III elaborates on preprocessing of vocalizations and piglet screaming through machine learning. We show that the accuracy of our prediction is better than those reported in the previous studies. Section IV elaborates on optimal AI model parameter setting. Section V conducts measurements, analytic analysis and simulation experiments to investigate the time complexity for detecting piglet screaming and triggering sow-alert actuators. Finally, Section VI concludes this article.

II. PigTalk Platform Based on IoT

Based on the IoT application platform IoTtalk [8], we have deployed PigTalk in a farrowing house with several farrowing cages in Yi-Lan, Taiwan. In the current implementation, the top of each cage has a directional microphone [see Fig. 1(a)] to receive sounds from that cage. The microphone is installed at about 150 cm above the pigs to ensure that weak screams can still be detected by PigTalk. A rotating Internet (IP) camera [see Fig. 1(b)] is installed on the wall to monitor several cages. The heating lights [see Fig. 1(c)] are controlled by the temperature sensors to keep every cage warm. Heating lights can also be used to alert a sow to stand up. Other sow-alert actuators include floor vibration [see Fig. 1(d)], water drop [see Fig. 1(e)] and more.

A. PigTalk Functional Block Diagram

Fig. 2 shows the PigTalk functional block diagram. Besides the microphone and the temperature sensor [see Fig. 2(a)], we also use an audio database [see Fig. 2(d)] to store the historical scream data and their labels to train AI models. The label
associated with an audio sample indicates whether the sample is a piglet scream or not. The IP Camera [see Fig. 2(b)] connects to the streaming server [see Fig. 2(e)] through an Ethernet cable. A hog farmer uses a smartphone [see Fig. 2(f)] to access the video from the streaming server. The hog farmer also uses the smartphone to inform PigTalk whether an alert is true or false. Two types of actuators are controlled by PigTalk [see Fig. 2(c)]. In this example, the environment actuators are the heating lights. Sow-alert actuators include vibration floors, air blasts, sprinklers, and electrodes. The PigTalk server consists of the engine [see Fig. 2(g)] and the graphical user interface [see graphical user interface (GUI); Fig. 2(g)]. The GUI is used to connect IoT devices (to be elaborated in Fig. 3).

B. Device Models and Device Feathers

Six software modules called device models [Fig. 2(i) and (l)–(p)] are developed to connect IoT devices [see Fig. 2(a)–(d), (f), (q), and (r)] to the PigTalk engine. Every device model consists of two components: the sensor/actuator application [SA; e.g., Fig. 2(j)] and the device application [see DA; e.g., Fig. 2(k)]. Consider the sensors device model [e.g., Fig. 2(i)] as an example. The sensor SA is responsible for the interaction between the physical sensor devices [e.g., Fig. 2(a), (d), and (f)] and the corresponding device models. Every DA consists of several device features (DFs) to interact with the PigTalk engine. An input DF [e.g., Temperature-I of the Sensor DA; Fig. 2(k)] forwards the data obtained from the physical IoT device [e.g., the temperature sensor] to the PigTalk engine. On the other hand, an output DF [e.g., Vibration-O of the Actuator DA; Fig. 2(m)] receives the instructions from the PigTalk engine to activate the actuator [e.g., vibration floor; Fig 2(c)]. For the discussion purpose, all input DFs are appended with “-I” and all output DFs are appended with “-O.” Two device models DataBank [see Fig. 2(n)] and ML_device [see Fig. 2(o)] are created in PigTalk for raw data preprocessing and AI prediction.

1) DataBank is connected to a database [see Fig. 2(q)] to store the real-time audio data obtained from the microphones. The database is accessed by DataBank as well as Audio2 [see Fig. 2(d)]; the connection is not shown in the figure).

2) ML_device is connected to an AI machine [see Fig. 2(r)]. In the current implementation, Nvidia GeForce RTX 2080 is used for training and validation, and an Intel Core i7-7800X CPU is used for pre-processing. The details of DataBank and ML_device will be given in Section III.

C. PigTalk GUI

Fig. 3 provides the details of the PigTalk GUI [see Fig. 2(h)] and illustrates how it is used to connect the sensors/controls to actuators.

In this web-based GUI, an icon representing a group of input DFs is placed in the left-hand side of the window [e.g., the sensor...
DFs in Fig. 3(a)], and an icon representing a group of output DFs is placed in the right-hand side of the window [e.g., the actuator DFs in Fig. 3(g)]. Every device model icon consists of several small icons representing the DFs. If a device model has both input-DF and output-DF parts, then it is represented by two device model icons in the GUI window [e.g., DataBank in Fig. 3(b) and (e)]. A developer can drag a line to connect an input DF to an output DF. Every line consists of two segments and one “join” circle. By clicking the circle, a window is popped up. The developer can write a Python function in this window to manipulate the data delivered from the input DFs to the output DFs (to be elaborated later). In Fig. 3, Join 1 delivers the voice data from the directional microphones to Audio-O1 of DataBank through Audio-I1 of Sensors for real-time piglet crushing detection. Join 2 delivers the voice data from the historical database to Audio-O2 through Audio-I2 for AI model training. Join 3 provides the labels for the historical data. Both the microphone and the historical scream data are connected to DataBank through the Sensors device model where the Real-Time Streaming Protocol (RTSP) is used to transmit the streaming audio.

**D. DataBank, AI and Automatic Control**

The DataBank SA includes a database [see Fig. 2(q)] to store the real-time data received from Audio-I1 (i.e., the microphone) associated with the corresponding labels received from Label-I. The labeled voice data are cut into fixed-length audio clips. The SA then performs fast Fourier transforms to transform the audio clips into the spectrograms. The spectrograms are sent from Spectrogram-I of DataBank [see Fig. 3(b)] to Spectrogram-O of ML_device [see Fig. 3(f)]. When a hog farmer receives the alert, he/she watches the video sent from the camera [see Fig. 4(b)] through path (b)->(e)->(f) in Fig. 2.

If the piglet scream is not caused by crushing, then the hog farmer remotely stops actuators [see Fig. 4(f)]. If the piglet crush does occur, he/she should run to the cage to handle this piglet crushing event, and stop actuators when the dangerous situation is relieved. The hog farmer may select the actuators to be turned ON or OFF [see Fig. 4(a)]. In PigTalk, the environment actuators such as heating light can be manually turned ON/OFF [see Fig. 4(d)] or automatically turned on/off when the temperature [see Fig. 4(e)] changes.

Automatic temperature control is achieved by creating a Python function as follows. By clicking the join 4 circle in Fig. 3, a function management window pops up (see Fig. 5) and the developer programs how the temperature sensor is used to trigger the heating lights. Specifically, if the temperature is no higher than 26 °C, then the heating lights are turned on (value 1). On the other hand, if the temperature is no lower than 28 °C, then the heating lights are turned off (value 0). Setting two thresholds (26 °C and 28 °C) avoids oscillation of triggering the heating lights.
The CNN model consists of five major layers including the Convolution Layer, the rectified linear units (ReLU) layer, the pooling layer, the fully connected layer and the loss layer. The CNN model can extract more features with a larger number of kernels. Note that a large number of kernels or a small kernel size result in more convolution operations and thus increase the processing time.

2) The ReLU layer [see Fig. 6(c)] replaces the negative value with 0 to reduce the calculation time and improves the accuracy of prediction.

3) The pooling layer [see Fig. 6(d)] performs maximum or average of the features obtained from the previous step. For example, in min-max scaling, a window of size $3 \times 3$ is used to move across the features and extracts the maximum values in the windows. Following [17] and [18], this layer adopts the max-pooling to achieve better performance.

4) The fully connected layer [see Fig. 6(e)] is similar to the multilayer perceptron, which classifies the output from the previous layer. The dropout rate or the probability to drop the nodes in the neural network, is defined in this layer to prevent the neural network from overfitting the training subset. The Loss Layer [see Fig. 6(f)] estimates the difference between the predicted result and the target result. This layer uses $\text{Softmax}$ as a classic loss function for multiple exclusive classes. The GUI window for configuring the parameters of DataBank and ML_device can be found in [19], and the details are omitted.

To classify the audio data, we apply a majority voting mechanism with the member $m$. Fig. 7(c) and (d) illustrates an example.
where \( m = 5 \). In this example, the CNN model produces a prediction 0 (normal) or 1 (abnormal) for every spectrogram. The majority vote for the consecutive \( m \) predictions is used as the final prediction results. For example, based on the majority voting mechanism, the consecutive predictions [0, 0, 1, 0, 0] and [0, 1, 0, 0, 1] in Fig. 7(d) result in “normal” for \( t = 2 \) and 2.5, and the consecutive predictions [1, 0, 0, 1, 1] result in “abnormal” for \( t = 3 \).

IV. MODEL TUNING

To obtain better prediction accuracy, we investigate the optimal values of the kernel number and the kernel size in the Convolution Layer [see Fig. 6(b)], the pooling window size in the pooling layer [see Fig. 6(d)] and the dropout rate in the loss layer [see Fig. 6(e)]. To see how good the training is, the categorical cross-entropy loss function \([15]\) is often used in the classification problems where only one result can be correct. The training process is iterated for several times until the loss is sufficiently small. The early stopping function \([16]\) is used to monitor the loss from the training and the testing processes. Specifically, the training process stops when the loss of the test is less than a threshold (i.e. min-delta). In our experiments, the min-delta threshold is set to 0.02.

A. Kernel Number and Kernel Size

The effects of the kernel number and the kernel size (at the convolution layer; see Fig. 6(b)) on the prediction accuracy are shown in Fig. 8. The figure indicates a general trend that the prediction accuracy increases as the kernel number or the kernel size increase. When the kernel number is larger than 7, increasing the number does not significantly improve the prediction accuracy. For min-max scaling, the optimal number is 7. For MFCC, the value is 3. Similarly, when the kernel size is larger than \( 2 \times 2 \), increasing the size does not significantly improve the prediction accuracy. Fig. 8(b) shows that Min-Max Scaling is more stable than MFCC in terms of the kernel size.

B. Polling Window and Dropout Rate

The effects of the pooling window size [at the pooling layer; Fig. 6(d)] and the dropout rate [at the loss layer; Fig. 6(e)] are shown in Fig. 9. This figure indicates a general trend that the prediction accuracy decreases as the optimal window size increases. Specifically, an appropriate polling window size is either \( 2 \times 2 \) or \( 3 \times 3 \). The figure also indicates that the optimal accuracy is observed when the dropout rate is \( p = 0.35 \).

dropout rate is insensitive to Min-Max Scaling for all \( p \) values, and is insensitive to MFCC when \( p > 0.2 \).

Figs. 8 and 9 show that if MFCC is used in preprocessing [see Fig. 6(a)], the prediction accuracy is significantly affected by the kernel size, the kernel number and the dropout rate, and is insensitive to the window size. When min-max scaling is used, the accuracies are significantly affected by the kernel number, and are insensitive to other parameters. The above experiments indicate that it is easier to tune the parameters with Min-Max Scaling than that with MFCC.

C. MFCC Versus Min-Max

Figs. 10(a) and 11(a) visualize the normal vocalization data after they are preprocessed by min-max scaling and MFCC, respectively. Similarly, Figs. 10(b) and 11(b) visualize the abnormal vocalization data after they are pre-processed. Fig. 10 indicates that after preprocessing with min-max scaling, abnormal vocalizations have significant ripples above 2500 Hz, while normal vocalizations do not. On the other hand, through feature extraction and smooth filtering steps of MFCC, the information retained is less than that of min-max scaling, and it is more difficult to identify the differences between the visual pictures of normal and abnormal vocalizations.
Fig. 12. Effects of the voting mechanism on the prediction accuracy.

The optimal prediction accuracy values and the corresponding parameter values of MFCC and min-max scaling are listed in Table I for \( m = 1 \) (without voting).

1) The accuracies given in Table I are obtained with balanced dataset. The table indicates that the prediction accuracies output by MFCC and min-max scaling are 98.9\% and 99.4\%. Therefore, min-max scaling provides higher accuracy with a larger spectrogram size of balanced dataset as compared with MFCC.

2) If we use the original imbalanced data set including 848-second abnormal vocalizations and 43 200-s normal clips. The prediction accuracies output by the MFCC and the min-max scaling methods using the same parameters given in Table I are 16.799\% and 94.478\%, respectively. These results show that the min-max scaling method still provides high accuracy when the normal and abnormal vocalizations are imbalanced. However, the MFCC method results in very low accuracy with the imbalanced dataset.

3) By applying majority vote with the imbalanced dataset, Fig. 12 shows that the accuracy of min-max Scaling increases when \( m \) increases, but the accuracy of MFCC decreases as the number \( m \) increases. When the number \( m \) of the voting members is 3, the prediction accuracies of both MFCC and min-max scaling methods reach 100\% with the balanced dataset.

### V. DELAY PERFORMANCE OF PIGTALK

To handle piglet crushing with the IoT technology, it is essential that the screaming is quickly detected by the microphone and the AI prediction, and sow-alert actuators are quickly activated to force a sow to stand up. In other words, the delay of the message path from a PigTalk sensor to sow-alert actuators must be shorter than the elapsed time between when the piglet is crushed and when it dies. This elapsed time \( \tau \) is called the piglet crushing time. This time is reported to be one minute [13]. Let \( t_0 \) be the delay that the crashed voice is sent from the microphone to the PigTalk server. Suppose that there are \( n \) sow-alert actuators in PigTalk. For \( 1 \leq l \leq n \), let \( t_l \) be the delay that the PigTalk server sends the instruction to the \( l \)-th actuators. In the current PigTalk implementation, \( n = 4 \), and \( l = 1 \) represents light-heating, \( l = 2 \) represents floor-vibration, \( l = 3 \) represents water-drop, and \( l = 4 \) represents air-blast. By convention, let \( l = 0 \) represent the subscript of \( t_0 \).

#### A. DELAY MEASUREMENTS

For \( 0 \leq l \leq n \), we have measured the delays \( t_l \) between PigTalk and the sensors/actuators using asymmetric digital subscriber loop (ADSL) (wired) and 4G LTE (wireless) communications. Our study indicates that these delays for sending data from the sensors to the PigTalk server (i.e., \( t_0 \)) and sending instructions from the PigTalk server to actuators (i.e., \( t_l \) for \( 1 \leq l \leq n \)) are roughly the same, and therefore can be approximated by \( n + 1 \) i.i.d. random variables. Specifically, for \( 0 \leq l \leq n \), Fig. 13 illustrates the histograms of \( t_l \) [20], where the expected value \( E[t_{l,ADSL}] \) and the variance \( V[t_{l,ADSL}] \) for the ADSL communication are

\[
E[t_{l,ADSL}] = 42.638 \, \text{ms}, \quad V[t_{l,ADSL}] = 0.018 \, E[t_{l,ADSL}]^2.
\]

For the 4G communication, Fig. 13 indicates that

\[
E[t_{l,4G}] = 85.958 \, \text{ms}, \quad V[t_{l,4G}] = 0.057 \, E[t_{l,4G}]^2.
\]

Fig. 13 indicates that the IoT message transmission using ADSL is much faster than that for 4G. Also, ADSL is more stable than 4G (in terms of the variance).

In [20], \( t_l \) are approximated by the \( i \)-stage Erlang distribution with the shape parameter \( i \) and the scale parameter \( \lambda \). The density function \( f(t_l, \lambda, i) \) and its Laplace transform \( f^*(s, \lambda, i) \) are

\[
f(t_l, \lambda, i) = \frac{\lambda^i t_l^i e^{-\lambda t_l}}{(i-1)!} \quad \text{and} \quad f^*(s, \lambda, i) = \frac{\lambda^i}{(s + \lambda)^i}
\]

where the shape parameter is \( i = 56 \) and the scale parameter is \( \lambda = 1.313 \) for ADSL. For 4G, \( i = 17 \) and \( \lambda = 0.198 \).
The Erlang distribution is considered because the mixture of this distribution is widely used in modeling the transmission delay in telecommunications networks, and our approximation is validated by the Kolmogorov-Smirnov test for goodness of fit [20]. From (3), the cumulative function of $t_l$ is

$$F(t_l, \lambda, i) = 1 - \left(\frac{1}{\lambda}\right)^i \sum_{j=1}^{i} f(t_l, \lambda, j).$$

Consider the order statistics of $n$ i.i.d. random variables $t_l$ ($1 \leq l \leq n$). We rearrange $t_l$ such that $t_{(1)} < t_{(2)} < \cdots < t_{(i)} < \cdots < t_{(n)}$. Then for the $l$th largest random variable, its density function $f(t_{(l)}, \lambda, i)$ is

$$f(t_{(l)}, \lambda, i) = \frac{n!}{(l-1)! (n-l)!} f(t_l, \lambda, i) \times \frac{1}{\lambda} \left[ F(t_{(l)}, \lambda, i) \right]^{l-1} \left[ 1 - F(t_{(l)}, \lambda, i) \right]^{n-l}.$$

When $l = n$, the density function $f_{(n)}(t)$ of the largest random variable is

$$f_{(n)}(t_{(n)}, \lambda, i) = \frac{n!}{(n-1)!} \int_{t_{(n)}}^{\infty} f(t, \lambda, i) \, dt = n f(t_{(n)}, \lambda, i) \left[ F(t_{(n)}, \lambda, i) \right]^{n-1}. \quad (5)$$

Substitute (4) into (5) to yield

$$f_{(n)}(t_{(n)}, \lambda, j) = n f(t_{(n)}, \lambda, j) \left[ 1 - \frac{1}{\lambda} \sum_{j=1}^{i} f(t_{(n)}, \lambda, j) \right]^{n-1}. \quad (6)$$

When the PigTalk engine receives the voice data, it instructs DataBank and ML_device to predict if an abnormal situation occurs. The delay between when the PigTalk engine sends the prediction request and when it receives the result is denoted as $t_c$. Fig. 14 shows the histogram of $t_c$, where the expected value $E[t_c]$ and the variance $V[t_c]$ for min-max scaling are

$$E[t_c] = 7.090 \text{ ms}, V[t_c] = 0.024 E[t_c]^2. \quad (7)$$

Fig. 14 indicates that the AI prediction times is fast ($E[t_c]$ is small) and stable ($V[t_c]$ is small). The $t_c$ histogram can be approximated by the Gamma function with the shape parameter $\alpha$ and the scale parameter $\beta$, where

$$f_c(t_c, \beta, \alpha) = \frac{\beta^\alpha t_c^{\alpha-1} e^{-\beta t_c}}{\Gamma(\alpha-1)} \quad \text{and} \quad f_c^*(s, \beta, \alpha) = \frac{\beta^\alpha}{(s+\beta)^\alpha} \quad (8)$$

where from (7), $\alpha = 41.667$ and $\beta = 5.877$.

Let $T_0 = t_0 + t_c$ then the density function $T_0$ is

$$f_0(T_0, \beta, \alpha, \lambda, i) = \int_{t_c=0}^{T_0} f(T_0 - t_c, \lambda, i) \left[ \frac{\beta^\alpha t_c^{\alpha-1} e^{-\beta t_c}}{\Gamma(\alpha-1)} \right] \, dt_c \quad (9)$$

and its Laplace transform can be derived from (3), (8), (9), and the convolution law of Laplace transform

$$f_0^*(s, \beta, \alpha, \lambda, i) = \frac{\beta^\alpha \lambda^i}{(s+\beta)^\alpha (s+\lambda)^i}. \quad (10)$$

### B. Analytic Modeling for Piglet Survival Rate

This section uses the modeling tools similar to those in [22] to conduct analytic analysis. From [7], we assume that a sow stands up when all $n$ actuators are activated. Thus, the piglet is saved after it is crushed by the sow if actuators are activated to force the sow to stand up within $\tau$. After the PigTalk server have sent out the instructions, all actuators are activated in the period $t_{(n)} = \max_{1 \leq l \leq n} t_l$. Therefore, crushed piglet is saved with the probability $P(\tau > T_0 + t_{(n)}).$ In [7] the mean $E[\tau] = \frac{1}{\lambda} = 1$ min, however, no one has derived the distribution for $\tau$. To conduct the mean value analysis [21], we assume that the piglet crushing time $\tau$ has an Exponential distribution with the density function $f_c(\tau) = \gamma e^{-\gamma \tau}$. From (5) and (9), we have

$$P(\tau > T_0 + t_{(n)}) = \left[ f_0^* (s, \beta, \alpha, \lambda, i) \Big|_{s = \gamma} \right] \times \left[ f_{(n)}^* (s, \lambda, i) \Big|_{s = \gamma} \right]. \quad (11)$$

From (10), we have

$$P(\tau > T_0 + t_{(n)}) = \left[ \frac{\beta^\alpha \lambda^i}{(\gamma + \beta)^\alpha (\gamma + \lambda)^i} \right] \times \left[ f_{(n)}^* (s, \lambda, i) \Big|_{s = \gamma} \right]. \quad (11)$$

Hensman and Masko [13] reported that by using the floor vibration and the air blast mechanisms, 80% of sows react up. Therefore, it is appropriate to select $n = 2$. For $n = 2$, (6) is rewritten as

$$f(2) (t_{(n)}, \lambda, j) = 2 f(t_{(n)}, \lambda, i) F(t_{(n)}, \lambda, i) = 2 f(t_{(n)}, \lambda, i) - \sum_{j=1}^{i} \left[ \frac{f(t_{(n)}, 2\lambda, i+j-1)}{2^{i+j-2}} \right]. \quad (12)$$
From (3), the Laplace transform of (12) is

\[ f_{(2)}^{(i)}(s, \lambda, i) = \frac{2\lambda^i}{(s + \lambda)^i} - \sum_{j=1}^{i} \left( \frac{i + j - 2}{i - 1} \right) \left[ \frac{2\lambda^{i+j-1}}{(s + 2\lambda)^{i+j-1}} \right]. \tag{13} \]

From (3) and (13), (11) rewritten as

\[
\begin{aligned}
\Pr[\tau > T_0 + t_{(2)}] &= \left[ \frac{\beta^\alpha \lambda^i}{(\gamma + \beta)^\alpha (\gamma + \lambda)^i} \right] \\
&\times \left\{ \frac{2\lambda^i}{(\gamma + \lambda)^i} - \sum_{j=1}^{i} \left( \frac{i + j - 2}{i - 1} \right) \left[ \frac{2\lambda^{i+j-1}}{(s + 2\lambda)^{i+j-1}} \right] \right\}.
\end{aligned}
\tag{14}
\]

From (14) and since

\[
\sum_{j=1}^{i} \left( \frac{i + j - 2}{i - 1} \right) \left( \frac{1}{2^{i+j-2}} \right) = 1.
\]

We have the intuitive results that

\[
\lim_{\gamma \to \infty} \Pr[\tau > T_0 + t_{(2)}] = 0 \quad \text{and} \quad \lim_{\gamma \to 0} \Pr[\tau > T_0 + t_{(2)}] = 1.
\]

If \( i = 1 \), then (14) is simplified as

\[
\Pr[\tau > T_0 + t_{(2)}] = \left( \frac{\beta}{\gamma + \beta} \right)^\alpha \left( \frac{2\lambda^2}{\gamma + \lambda} \right) \left\{ \frac{1}{\gamma + \lambda} - \frac{1}{\gamma + 2\lambda} \right\} = \frac{2\beta^\alpha \lambda^3}{(\gamma + \beta)^\alpha (\gamma + \lambda)^3 (\gamma + 2\lambda)}.
\]

If three sow-alert actuators are selected, then \( n = 3 \), and (6) is rewritten as

\[
f_{(3)}(t_{(3)}, \lambda, i) = 3 \left[ A(t_{(3)}, \lambda, i) + B(t_{(3)}, \lambda, i) \right],
\tag{15}
\]

where

\[
A(t_{(3)}, \lambda, i) = f(t_{(3)}, \lambda, i)
\]

\[
- \left( \frac{2}{\lambda} \right) f(t_{(3)}, \lambda, i) \left[ \sum_{j=1}^{i} f(t_{(3)}, \lambda, j) \right]
\]

and

\[
B(t_{(3)}, \lambda, i) = \left( \frac{1}{\lambda} \right)^2 \times \left[ \sum_{k=1}^{i} \sum_{j=1}^{i} f(t_{(3)}, \lambda, i) f(t_{(3)}, \lambda, j) f(t_{(3)}, \lambda, k) \right].
\tag{17}
\]

Substitute (4) into (16) to yield

\[
A(t_{(3)}, \lambda, i) = f_{(2)}(t_{(3)}, \lambda, i) - f(t_{(3)}, \lambda, i).
\tag{18}
\]

From (18) and (13), the Laplace transform of \( A(t_{(3)}, \lambda, i) \) is

\[
A^*(s, \lambda, i) = \frac{\lambda^i}{(s + \lambda)^i} - \sum_{j=1}^{i} \left( \frac{i + j - 2}{i - 1} \right) \left[ \frac{2\lambda^{i+j-1}}{(s + 2\lambda)^{i+j-1}} \right]. \tag{19}
\]

From (17), \( B(t_{(3)}, \lambda, i) \) is expressed as

\[
B(t_{(3)}, \lambda, i) = \sum_{k=1}^{i} \sum_{j=1}^{i} \left( \frac{i + j - 2}{i - 1} \right) \left( \frac{i + j + k - 3}{k - 1} \right)
\times \left[ f(t_{(3)}, 3\lambda, i + j + k - 2) \right]. \tag{20}
\]

From (3) and (20), the Laplace transform of \( B(t, \lambda, i) \) is

\[
B^*(s, \lambda, i) = \sum_{j=1}^{i} \left( \frac{i + j - 2}{i - 1} \right)
\times \left[ \sum_{k=1}^{i} \left( \frac{i + j + k - 3}{k - 1} \right) \left( \frac{\lambda}{3\lambda} \right)^{i+j+k-2} \right]. \tag{21}
\]

From (15), (19), and (21), the Laplace transform of \( f_{(3)}(t_{(3)}, \lambda, i) \) is

\[
f_{(3)}^* (s, \lambda, i) = 3 \left[ A^*(s, \lambda, i) + B^*(s, \lambda, i) \right]
\]

\[
= 3 \left\{ \left[ \frac{\lambda^i}{(s + \lambda)^i} - \sum_{j=1}^{i} \left( \frac{i + j - 2}{i - 1} \right) \left( \frac{\lambda}{s + 2\lambda} \right)^{i+j-1} \right]
\times \left[ \left( \frac{i + j + k - 3}{k - 1} \right) \left( \frac{\lambda}{s + 3\lambda} \right)^{i+j+k-2} \right] \right\}.
\tag{22}
\]

Substitute (22) into (11) to yield

\[
\Pr[\tau > T_0 + t_{(n)}] = \left[ f^* (s, \lambda, i) \right]_{s=\gamma} = \left[ f_{(3)}^* (s, \lambda, i) \right]_{s=\gamma} = \left[ \frac{3\lambda^i}{(\gamma + \lambda)^i} - \sum_{j=1}^{i} \left( \frac{i + j - 2}{i - 1} \right) \left( \frac{\lambda}{\gamma + 2\lambda} \right)^{i+j-1} \right]
\times \left[ \left( \frac{i + j + k - 3}{k - 1} \right) \left( \frac{\lambda}{\gamma + 3\lambda} \right)^{i+j+k-2} \right].
\tag{23}
\]

From (23) and since

\[
\sum_{j=1}^{i} \left( \frac{i + j - 2}{i - 1} \right) \sum_{k=1}^{i} \left( \frac{i + j + k - 3}{k - 1} \right) \left( \frac{1}{3^{i+j+k-3}} \right) = 1.
\]

We have the intuitive results that

\[
\lim_{\gamma \to \infty} \Pr[\tau > T_0 + t_{(3)}] = 0 \quad \text{and} \quad \lim_{\gamma \to 0} \Pr[\tau > T_0 + t_{(3)}] = 1.
\]

If \( i = 1 \), then (23) is simplified as

\[
\Pr[\tau > T_0 + t_{(n)}] = \left( \frac{\beta}{\gamma + \beta} \right)^\alpha \left( \frac{3\lambda^2}{\gamma + \lambda} \right)
\times \left\{ \left( \frac{1}{\gamma + \lambda} - \frac{\gamma + 4\lambda}{(\gamma + 2\lambda)(\gamma + 3\lambda)} \right) \frac{6\beta^\alpha \lambda^4}{(\gamma + \beta)^\alpha (\gamma + \lambda)^3 (\gamma + 2\lambda)(\gamma + 3\lambda)} \right\}.
\]
increases. For error between simulation and analytic analysis increases as for Piglet Survival Rate

We have conducted simulation to compute \( \Pr[\tau > T_0 + t_{(n)}] \). The simulation model follows the same approach used in [22], and the details are omitted. The simulations are validated against (14) and (23) with various parameter values for \( n \) and \( \gamma \). The error between simulation and analytic analysis increases as \( n \) increases. For \( \lambda = 1 \), when \( n \) increases from 1 to 3, the error increases from 0.016\% to 0.447\%. In all cases considered in our study, the error is within 0.5\%. In performance modeling, such low error means that the simulation agrees with analytic analysis.

After validation, we use the measured data (1) and (2) in the simulation to obtained \( \Pr[\tau > T_0 + t_{(n)}] \) for a farrowing house in Yi-Lan, Taiwan. We observe the following trends.

1) It is intuitive that wired (ADSL) transmission is better than wireless (4G) transmission. The nontrivial part is that we have derived \( \Pr[\tau > T_0 + t_{(n)}] \) for different types of transmissions, which is 99.93\% for ADSL and is 99.84\% for 4G (assuming \( \muE[\tau] = 60 \) s).

2) It is also intuitive that PigTalk performs well if the piglet crunching time \( \tau \) is short. The nontrivial part is that we have derived \( \Pr[\tau > T_0 + t_{(n)}] \) for various \( \muE[\tau] \) values, which is 99.82\% for \( \muE[\tau] = 30 \) s, 99.93\% for \( \muE[\tau] = 60 \) s, and is 99.94\% for \( \muE[\tau] = 90 \) seconds (assuming ADSL transmission).

In the realistic case where ADSL is used, \( \Pr[\tau > T_0 + t_{(n)}] \) is 99.93\%, which indicates that PigTalk can quickly activate sow-alert actuators, and can successfully avoid piglet crushing. We have continually collected the real-time datasets, and observed the PigTalk operations in the farrowing house since two years ago. Piglets were saved two or three times per month through piglet scream alerts. There was no piglet casualty during the observation period, which indicates that PigTalk effectively detects and avoids crushing. The snapshots in Fig. 15 show how PigTalk works in a commercial farrowing house. In Fig. 15(1), a sow stands up normally. It sits down in Fig. 15(2), and piglet screams are detected by PigTalk. PigTalk activates floor vibration and enhanced light heat (in the sow back). The sow lies down totally in Fig. 15(3). The floor continues to vibrate, and light heat has been increased in the sow back. In Fig. 15(4), the sow stands up again.

**VI. Conclusion**

Based on IoT and machine learning technologies, we developed the PigTalk platform for detecting and mitigating piglet crushing. We elaborated on how data preprocessing and machine learning mechanisms are developed in PigTalk as IoT devices for piglet scream detection. Through real-time analysis of the data received from the microphones, PigTalk detects if any piglet screaming occurs, and sends out an alert for emergency handling of the crushing event.

We showed that the spectrograms generated from min-max scaling yields better prediction performance than MFCC. Our CNN implementation yields the prediction with 99.4\% of accuracy when the kernel number is 7, the kernel size is 5, the pooling window size is 3, and the dropping rate is 0.05.

We also show how to trigger actuators such as floor vibration and water drop to force a sow to stand up using IoT. With the scream detection and the real-time sow-alert activation mechanism, PigTalk automatically reacts to the piglet crushing accidents, and can save crushed piglets with 99.93\%. PigTalk also provides a video control board that allows a hog farmer to remotely monitor farrowing cages through any smartphone with a simple web-based GUI, and can stop the actuators immediately if a false alarm occurs. Our approach nicely integrates IoT and AI in real time for mitigating piglet crushing, which detects and starts saving a piglet in 0.05 s (ADSL version). As more new data samples are collected and sent to the PigTalk server, the new samples are helpful to revise the prediction model. Such intelligent control for farrowing house has not been found in the literature.

PigTalk won the 2019 Special Award of MediaTek Smart Hometown Contest, Silver Award of 2019 Mobileheroes Communications Contest (Ministry of Economic Affairs), and Bronze Medal of 2019 Smart Manufacturing Big Data Contest (Ministry of Education). In the future, we will extend our work for data fusion of the microphone and the camera to automatically detect the crushed parts of piglets. Also, we will design new actuators to humanize the mitigation mechanism.

**References**


