Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Dynamic noise filtering for multi-class classification of beehive audio data

Dániel Tamás Várkonyi [a], José Luis Seixas Junior [a], Tomáš Horváth [a,b,*]

[a] ELTE – Eötvös Loránd University, Faculty of Informatics, Pázmány Péter sétány 1/C, 1117 Budapest, Hungary
[b] Pavol Jozef Šafárik University, Faculty of Natural Science, Šrobárova 2, 041 54 Košice, Slovakia

ARTICLE INFO

Keywords:
Audio data analysis
Spectrogram
Noise filtering
Audio feature extraction
Apiculture

ABSTRACT

Honeybees are the most specialized insect pollinators and are critical not only for honey production but, also, for keeping the environmental balance by pollinating the flowers of a wide variety of crops.

Recording and analyzing bee sounds became a fundamental part of recent initiatives in the development of so-called smart hives. The majority of researches on beehive sound analytics are focusing on swarming detection, a relatively simple binary classification task (due to the obvious difference in the sound of a swarming and a non-swarming bee colony) where machine learning models achieve good performance even when trained on small data.

However, in the case of more complex tasks of beehive sound analytics, even modern machine learning approaches perform poorly. First, training such models would need a large dataset but, according to our knowledge, there is no publicly available large-scale beehive audio data. Second, due to the specifics of beehive sounds, efficient noise filtering methods would be required, however, we could not find a noise filtering method that would increase the performance of machine learning models substantially.

In this paper, we propose a dynamic noise filtering method applicable on spectrograms (image representations of audio data) which is superior to the most popular image noise filtering baselines. Further, we introduce a multi-class classification task of bee sounds and a large-scale dataset consisting of 10.000 beehive audio recordings. Finally, we provide the results of a large-scale experiment involving various combinations of audio feature extraction and noise filtering methods together with various deep learning models. We believe that the contributions of this paper will facilitate further research in the area of (beehive) sound analytics.

1. Introduction

Despite the importance of honeybees in the ecosystem and agriculture (Goulson et al., 2015; Patrício-Roberto & Campos, 2014) as well as extensive research on apiculture and honeybee health, honeybee colonies have suffered significant declines in recent years due to various diseases and irresponsible agricultural practices (Neumann & Carreck, 2010).

In the majority of apiaries, identification of the health condition of a bee colony is done manually by opening and inspecting the hive. Opening the hive, however, introduces certain stress to the colony while changing the micro-climate within the hive. Afterward, bees have to expend considerable energy to re-establish the equilibrium within the beehive. Consequently, frequent manual inspection of a hive reduces the amount of honey the given bee colony produces.

Several IoT-based research were addressed to non-invasive investigation of within hive processes and colony health monitoring without its physical intervention. Early works have utilized various technologies to measure some quantitative parameters of the hive, such as temperature (Zacepins et al., 2013), mass (Zacepins et al., 2015) and humidity (Ferrari et al., 2008).

A very important qualitative parameter one can measure within the hive, which is in the focus of this paper, is the sound of the colony. Analyzing the colony's sound might reveal certain anomalous events within the hive, like the presence of an intruder or the preparation of the colony for swarming, which might be hard or even impossible to detect from the above-mentioned quantitative parameters. The first technological approaches used for monitoring bees' condition via audio analysis were conducted in the late 20th century using spectral analysis in the range of 0–3 kHz (Dietlein, 1985).

Recent works on beehive audio analysis are focusing mostly on bee queen presence detection (Soares et al., 2022) and swarming prediction (Bencsik et al., 2011; Cejrowski et al., 2018; Ferrari et al., 2008; Howard et al., 2013; Zgank, 2020), two correlating factors related to the strength of the colony. Translated into the terminology of machine learning (ML), these two problems correspond to a rather basic binary classification task, i.e. predicting if the queen is present or not and if

* Corresponding author at: ELTE – Eötvös Loránd University, Faculty of Informatics, Pázmány Péter sétány 1/C, 1117 Budapest, Hungary.
E-mail addresses: varkonyid@inf.elte.hu (D.T. Várkonyi), jlseixasjr@inf.elte.hu (J.L. Seixas Junior), tomas.horvath@inf.elte.hu (T. Horváth).

https://doi.org/10.1016/j.eswa.2022.118850
Received 13 July 2022; Received in revised form 12 September 2022; Accepted 14 September 2022
Available online 20 September 2022

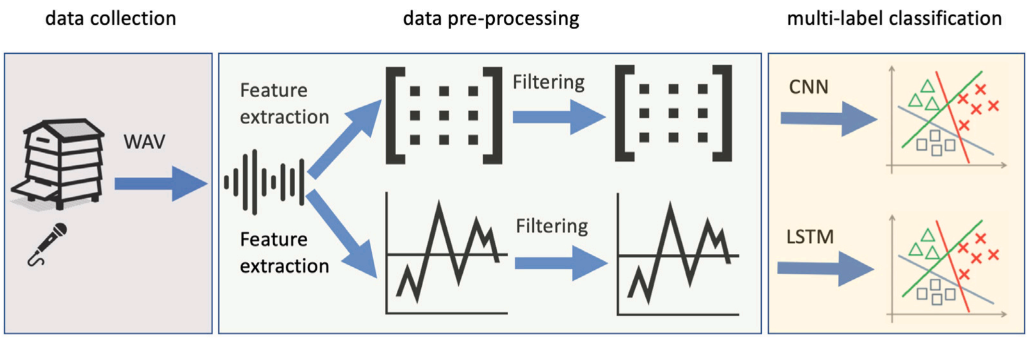

Fig. 1. A general audio data analytics workflow used in our experiments.

the colony is swarming or not. Moreover, the sound of a bee colony in a "queenless" or swarming state is well distinguishable, even for human ears, from its sound when a queen is present or the colony is not swarming (Allen, 1956; Hord & Shook, 2013). In the light of the above facts, it is not surprising that many ML techniques achieve good prediction accuracy on these tasks (Soares et al., 2022; Zgank, 2021).

There are other problems important in beekeeping, such that, for example, assessing the health status of the colony (Robles-Guerrero et al., 2017), detecting the exposure of bees to chemicals (Sharif et al., 2020), identifying the presence of predators (Zahid Sharif et al., 2020) and pests in the hive (Chao & Shouying, 2022) or monitoring the bees' daily activity (Dietlein, 1985; Mezquida & Martínez Llorente, 2009). These problems are approached in the literature by either (manually) analyzing audio features in bee sounds (Shostak & Prodeus, 2019) and/or as a binary classification task (i.e. predicting if the colony is healthy, if bees have been exposed to chemicals or if predators/pests are present in the beehive). Moreover, in each of these use-cases, similarly to swarming or queenless states described before, the stress level of the colony exposed to an "anomaly" likely implies a substantially different sound from its sound when being in a normal state. An important issue, contravening the approaches found in the literature, is that these problems might not necessarily correspond to binary classification tasks. For example, identification of various (more than two) types of diseases, intrusion detection by different pests, or estimation of exposure of bees to diverse palettes of chemicals, naturally, calls for multi-label or even multi-class approaches. An assumption here, posing possible difficulties for application of ML techniques, is that bee sounds corresponding to various classes (labels) might not be so easily distinguishable from each other as they usually are in the before mentioned binary cases where "anomalous" and "normal" states are well-detectable even for humans.

Problem 1. There is a huge gap in the literature considering multi-label or multi-class classification of bee audio data. According to our knowledge, also listing through recent literature surveys by Zacepins et al. (2015), Hadjur et al. (2022) and Abdollahi et al. (2022), the only work related to multi-label bee sound classification, published by Gradišek et al. (2017), is focused on identifying 12 bumblebee species from their buzzing sounds (from a small-scale dataset using traditional ML techniques, such that naïve Bayes decision trees, support vector machines, and random forests).

Usually, the classification accuracy of the audio data analytics workflow, shown in Fig. 1, does not only depend on the discriminative power of the used ML techniques but, also, on the used pre-processing steps which play a crucial role. The first step is to filter out irrelevant information from the audio file, such that non-essential frequency ranges.[1] One of the basic problems of audio signals is that, even after filtering out irrelevant information, the audio information represented as time-series have very high dimensionality. Feature extraction (FE) methods are utilized to extract and represent the most important features within the audio signal in a compact form, such that in form of time-series or spectrograms, by preserving a significant portion of its original information content. After FE, ML techniques can be applied on the data depending on the amount of the data, its dimensionality and the concrete task which is intended to be solved. Usually, the data resulting from FE still contains lots of noise, causing problems in the optimization of a reliable ML model. For such, time-series or image noise filtering (NF) methods can be applied on the data, depending if FE resulted in time-series or spectrograms (images). Given the number of possible FE and NF methods, the decision of how to set up a bee sound data analytics workflow (Fig. 1) would benefit from a comparative survey study on the influence of its various settings on its performance.

Problem 2. According to our knowledge, there is no reference in the literature to a survey article comparing various combinations of audio FE, (image or time-series) NF and ML (image or time-series classification) techniques for bee sound analytics, including thorough hyperparameter (HP) tuning and validation procedures as well as utilizing large datasets. Related comparisons are either focused on certain FE or ML techniques, are using relatively small amount of data, lack a thorough HP tuning or do not utilize proper validation procedures for the resulting ML models.

Such a survey would need large-scale publicly available bee sound data, crucial for training a more complex ML model (Bae et al., 2016; Zgank, 2021), such that a convolutional neural network (CNN), in case of spectrogram data (Nolasco et al., 2019), or a long–short term memory (LSTM) model, in case of time-series data (Ruvinga et al., 2021), on their inputs. Preferably, bee sound data corresponding to multi-label or multi-class problems would be welcome.

Problem 3. According to our knowledge, there is no large-scale publicly available bee sound benchmark data suitable for research on multi-label or multi-class bee sound classification approaches. Available data are either small-scale or recordings were not gathered under similar circumstances, mainly concerning same time frames, locations and recording devices.

Several ML approaches, e.g. various deep learning (DL) models, have been developed recently which work efficiently on specific problems related to audio processing like, for example, human speech recognition. However, our experiments showed that these methods are not performing well in case of a multi-label bee sound classification problem in which the difference between various classes is inconspicuous.

Problem 4. The main reason for the poor performance of state-of-the-art ML models for multi-label bee sound classification (with marginal differences between classes), identified by us, lies in the poor performance of the used NF methods which seem to be not suitable for bee sound data.

Reflecting to the identified problems, the contributions of this paper are the following:

[1] Meaningful information in bee sounds are carried in the 0–4 kHz frequency range (Collison, 2018; Hord & Shook, 2013; Qandour et al., 2014).

- *Collection of large-scale beehive audio data* consisting of 10.000 recordings, recorded from 10 beehives (From 700 to 1050 recordings were used from each hive) in the same apiary and in the same season by the same recording tools;
- *Introduction of a multi-label beehive audio classification task*, in which the 10 class labels correspond to the 10 different beehives. The goal in this use-case is to predict to which of the 10 hives the given sound record belongs to[2];
- *Development of a simple but efficient NF method for beehive audio data*, which substantially increases the performance of ML models while reducing their training time;
- *Extensive experimental study*, comparing the performances of various combinations of state-of-the-art audio FE and related NF methods as well as ML models;

The rest of the paper is organized as follows: In the next section, various FE, NF and ML methods utilized in a general audio data analytics workflow, as well as in our experiments, are described. Following, related works including publicly available data in beehive sound analytics are presented in Section 3. In the following two short Sections 4 and 5, respectively, the proposed dynamic audio noise filtering method and the collected large-scale bee sound data are introduced. Experiments and results are described in Sections 6 and 7, respectively, followed by a discussion (Section 8) and concluding remarks (Section 9).

2. Beehive sound analytics

The beehive sound analytics workflow used in this paper, illustrated in Fig. 1, consists of three main steps, such that FE, NF and ML (namely, multi-label classification), discussed in the subsections below.

In our experiments, despite the suggested frequency range of 0–4 kHz by Hord and Shook (2013) and Qandour et al. (2014), for the sake of investigation, the frequency range of 0–8 KHz has been considered. Thus, before the FE step, frequencies beyond this range are filtered out.[3]

2.1. Audio feature extraction (FE)

Formally, the input of FE is an audio file in the form of a real-valued time-series data $x(t) = \{x_t \mid 0 \le t < N\} = \{x_0, x_1, \dots, x_{N-1}\}$ of length N, where each x_t represents the amplitude of the signal in time t, as illustrated in Fig. 2.

The goal of FE is to transform or represent $x(t)$ via features carrying important information, such that the values of the magnitude spectrum at a certain time or time interval (preferably, the whole time interval of the recording). The main FE methods are briefly introduced below.

2.1.1. Short-Time Fourier Transform (STFT)

Discrete-Time Fourier Transformation (DTFT) (Winograd, 1978) transforms x_n into another complex series $y(k) = \{y_k \mid 1 \le k \le K\} = \{y_1, y_2, \dots, y_K\}$

$$y_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x_n[\cos(\frac{2\pi}{N}kn) - i \cdot \sin(\frac{2\pi}{N}kn)] \tag{1}$$

which represents the audio data $x(n)$ in the frequency domain instead of the time domain, thus, revealing important information. In case of STFT, $x(n)$ is broken up to windows $w(n-m)$, with a window size m, which might overlap. DTFT is applied on each window, recording the magnitude (and phase) of $x(n)$ for each frequency and time. STFT can be expressed as DTFT on $x(n)w(n-m)$, such that

$$y_k = \sum_{n=0}^{N-1} x_n w(n-m) \cdot e^{-i\frac{2\pi}{N}kn} \tag{2}$$

The information of the magnitude of the signal can be represented in a 2D "time–frequency" matrix called spectrogram, visualized using a heat map in Fig. 3.

2.1.2. Chroma

Chroma features (Kattel et al., 2019) represent the tonal content of an audio signal $x(n)$. Extraction includes the following steps: (1) Providing STFT for frequency analysis; (2) Filtering frequency: keeping the range between 0 and 8 KHz; (3) Peak detection: only the local maximum values of the spectrum are considered; (4) Reference frequency computation: estimate the deviation with respect to 440 Hz, which is the frequency of the A4 chord, often used as a standard for tuning musical instruments; (5) Pitch class mapping: with respect to the estimated reference frequency (for determining the pitch class value from frequency values) using a weighting scheme with a cosine function[4]; (6) Normalizing the feature: frame by frame, dividing through the maximum value to eliminate dependency on global loudness. Result of Chroma FE for the audio data from Fig. 2 is shown in Fig. 4.

2.1.3. Mel-frequency Cepstral coefficients (MFCC)

Extracting MFCC (Kattel et al., 2019) starts with computing DTFT of the signal $x(n)$, resulting in $y(k)$, as introduced in Eq. (1). Then, the Mel spectrum of the magnitude spectrum is computed as $s(m) = \sum_{k=0}^{N-1}[|y(k)|^2 h_m(k)]$, where $0 \le m \le M-1$, M is the total number of triangular Mel weighting filters (Fang et al., 2001; Ganchev et al., 2005) and $h_m(k)$ is the weight given to the kth energy spectrum bin contributing to the mth output band, expressed as $h_m(k) = \frac{2(k-f(m-1))}{f(m)-f(m-1)}$ for $f(m-1) \le k < f(m)$, $h_m(k) = \frac{2(f(m+1)-k)}{f(m+1)-f(m)}$ for $f(m) \le k < f(m+1)$ and $h_m(k) = 0$ otherwise. Finally, Discrete Cosine Transformation is applied such as $c(n) = \sum_{m=0}^{M-1} log(s(m))cos(\frac{\pi n(m-0.5)}{M})$, where $0 \le n \le C-1$, $c(n)$ are the Cepstral coefficients and C is the number of MFCCs. Result of MFCC FE for the audio data from Fig. 2 is shown in Fig. 5.

2.1.4. MFCC differential coefficients (MFCC delta)

The first order derivative of MFCC (differential) coefficients (Hossan et al., 2010) are defined as $d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$ where d_t is the delta coefficient of the frame t in terms of MFCCs from c_{t+n} to c_{t-n}. Result of MFCC Delta FE for the audio data from Fig. 2 is shown in Fig. 6.

2.1.5. Spectral centroid (SC)

SC (Giannakopoulos & Pikrakis, 2014), a measure used to characterize the spectrum, is defined as $SC(n) = \frac{\sum_{m=0}^{N-1} m \cdot |X[n,m]|^2}{\sum_{m=0}^{N-1} |X[n,m]|^2}$. SC indicates the "gravity centrum", in other words, the center frequency of the nth frame. Result of SC FE for the audio data from Fig. 2 is shown in Fig. 7.

2.1.6. Zero-crossing rate (ZCR)

ZCR (Giannakopoulos & Pikrakis, 2014), indicating the number of sign changes in the nth frame, is defined as $ZCR(n) = \frac{1}{2N}\sum_{m=1}^{N} |sgn(x[n+m]) - sgn(x[n+m-1])|$. ZCR can be interpreted as a measurement of noisiness. The higher the ZCR in a frame, the noisier the signal. Result of ZCR FE for the audio data from Fig. 2 is shown in Fig. 8.

The SC and ZCR FE result in a time-series type output, and not in a spectrogram as in case of the other FE methods introduced above.

[2] While such a use-case might seem "synthetic", the goal is to provide a certain challenge, i.e. a complex ML task, to the community.

[3] This filtering does not indicate noise filtering for which a novel approach is proposed later in this paper, but, it refers to irrelevant frequency range filtering.

[4] It considers the presence of harmonic frequencies (harmonic summation procedure), taking into account a total of 8 harmonics per frequency. To map the value on one-third of a semitone, the size of the pitch class distribution vectors must be equal to 36.

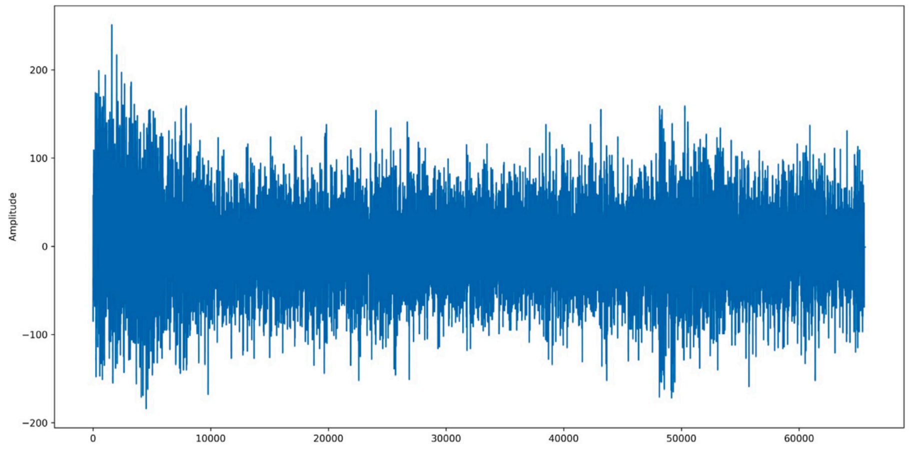

Fig. 2. The recorded audio (.wav) data.

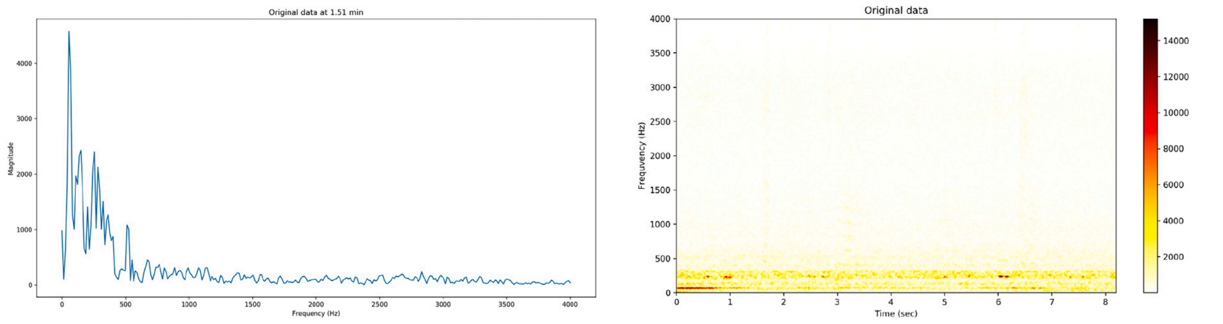

Fig. 3. The spectrogram related to audio data from Fig. 2. The magnitude spectrum at the time 1.51 is on the left while the whole magnitude spectrum is on the right.

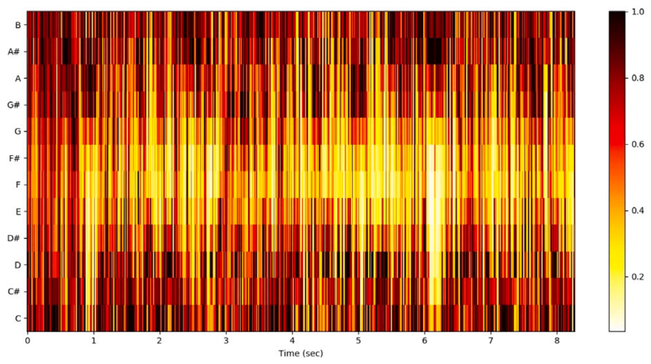

Fig. 4. Chroma feature representation of the audio data from Fig. 2.

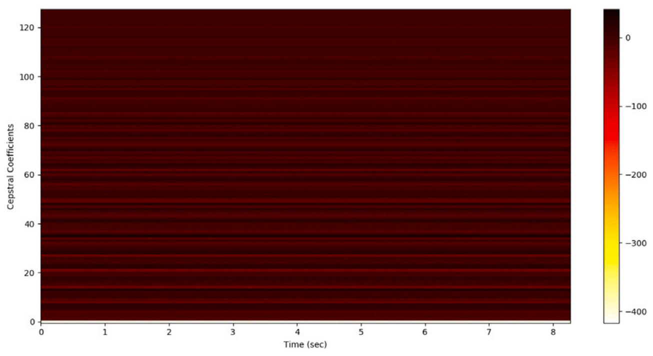

Fig. 5. MFCC feature representation of the audio data from Fig. 2.

2.2. *Multi-label audio data classification using machine learning (ML)*

Depending on which method is being used, FE might result in two different types of output, time-series data or a 2D feature matrix, a spectrogram, which can be seen as an image. According to these two types of outputs, the most suitable ML techniques can be used (as illustrated in Fig. 1).

2.2.1. *Time-series classification*

One of the best performing ML model for (multi-label) classification of long time-series, in which some trends might occur, is the LSTM, an artificial recurrent neural network (RNN) architecture built up by so-called LSTM units (Yuan et al., 2019). The architecture of our LSTM network contains an LSTM layer as input layer with n_u number of LSTM units. Then come n_{dl} number of deep layers, with n_u number of LSTM units. The next layer is the last LSTM layer with n_u number of LSTM units. To avoid the burnout of the training process, a dropout layer with 25% dropout rate is the following layer. The output layer of the network is a dense layer with 10 neurons (corresponding to the 10 labels/classes), using an act_o activation function. The training process is using an op optimizer. These HPs are related to the architecture of the network and are being fine-tuned during the learning process (the fine-tuned values are reported in Section 6). The general architecture of the used LSTM network is shown in Fig. 9.

2.2.2. *Image classification*

Following the latest years' research trends, it is clear that CNN is one of the best performing tools for (multi-label) image classification (Wang et al., 2021). CNN contains a "convolutional–max pooling" layer pair as input layers. The convolutional layer contains n_u number of 3×3 convolutional filters with act_d activation function and all filters are

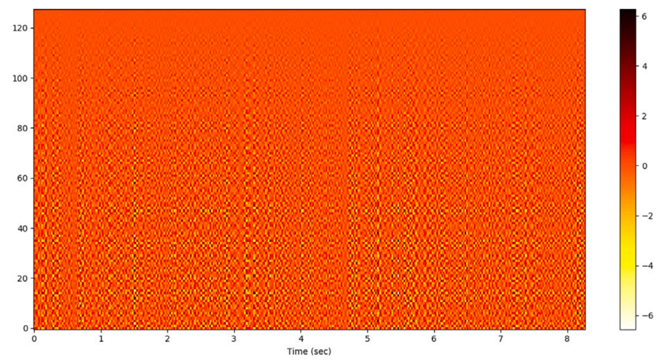

Fig. 6. MFCC delta feature representation of the audio data from Fig. 2.

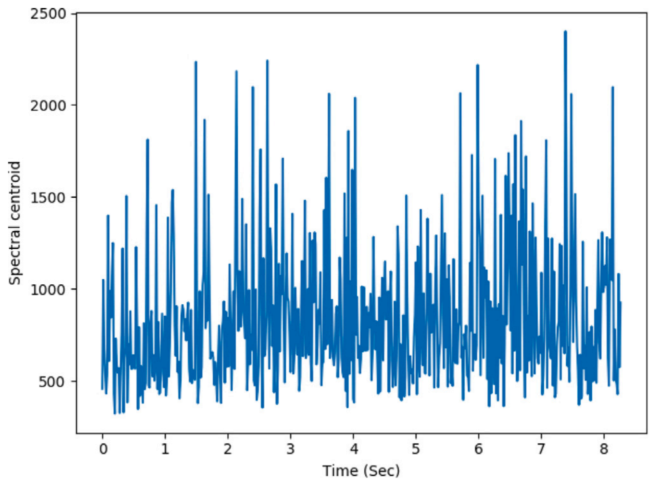

Fig. 7. SC feature representation of the audio data from Fig. 2.

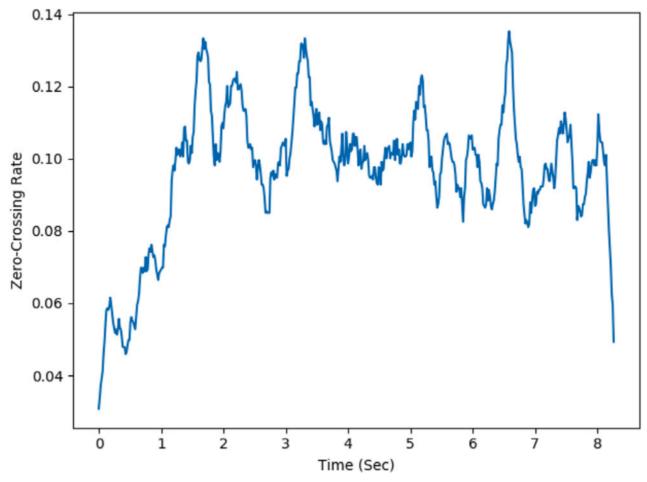

Fig. 8. ZCR feature representation of the audio data from Fig. 2.

followed by a 2 × 2 max pooling unit. After the first layer pair, n_{dl} number of convolutional–max pooling layer pair is present with n_u number of 3 × 3 convolutional filters using act_d activation function and all filters are followed by a 2 × 2 max pooling unit. All units are using stride value $(2, 2)$. To avoid the burnout of the training process, a dropout layer with 25% dropout rate is the following layer. The last deep layer is a dense layer with n_u number of units using act_d activation function. The output layer of the network is a dense layer with 10 neurons (corresponding to the 10 labels/classes), using act_o activation. The training process is using an op optimizer. These HPs are related to the architecture of the network and are being tuned during the learning process (the tuned values are reported in Section 6). Since Chroma FE results in a 2D feature set with the dimensionality of 12 × 132, the structure of the network in this case is simpler than in the case of other FE methods. The general architecture of the created CNN is shown by Fig. 10.

2.3. *Noise Filtering (NF)*

According to the presented methodology (Fig. 1), NF is applied after the FE and before the ML steps. Since FE results in a time-series or in an image (histogram), relevant NF methods are utilized. According to the experiments, FE resulting in time-series, i.e. SC and ZCR, have shown inferior performance compared to FE resulting in spectrograms (more details in Section 6). Thus, the focus of NF in this paper is on spatial filters with a focus on smoothing and/or eliminating noise in spectrograms.

A spatial filter consists of a kernel corresponding to the neighborhood of the analyzed pixel (x, y) and an operation, which can be linear or non-linear.

In Fig. 11, there is a schematic example of how a two-dimensional filter operation takes place, where f is the original signal (image/spectrogram), g is the kernel to be convoluted by the points of the original signal and h is the resulting signal. Therefore, h generates a new value for each (x, y) coordinate as the kernel g operates on each pixel $f(x, y)$.

2.3.1. *Laplacian filter*

Laplacian (Forsyth & Ponce, 2011) is a linear filter based on the discrete second derivative (Wang, 2007) $\frac{\partial^2 f}{x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$ In the two-dimensional case, the filter is obtained with the sum of the partial derivatives in both directions

$$g^L(x, y) = \frac{\partial^2 f}{x^2} + \frac{\partial^2 f}{y^2} \tag{3}$$

Eq. (3) can be generated by the kernels (Gonzalez & Woods, 2008)

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

taking into account, if preferable, to use so-called 4 or 8-neighboring.

As it is an operator that uses derivatives, the effect is to highlight intensity discontinuities, that is, edges, as the face areas have much smoother intensity transitions when they occur at all. This makes the edge lines stand out against the background, creating a sharpening filter.

2.3.2. *Gaussian filter*

The Gaussian filter (Dougherty, 2020) works like a weighted average where the weight values decrease with the distance from the pixel being evaluated. Convolution kernel generation follows the equation

$$g^G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ is the standard deviation and, as usual, (x, y) integers for kernel center coordinates, that is, the pixel being analyzed. Because it is an average, the value is divided by the sum of weights, just as the Gaussian is divided by the area of the circle to which it is distributed.

The standard deviation in the function controls how far the maximum value will influence neighbors, that is, how much of the distance causes a significant decrease.

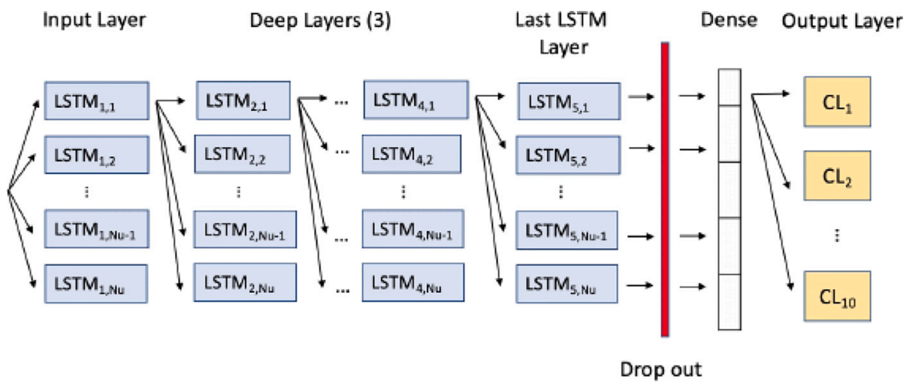

Fig. 9. The general LSTM structure used in the proposed methodology.

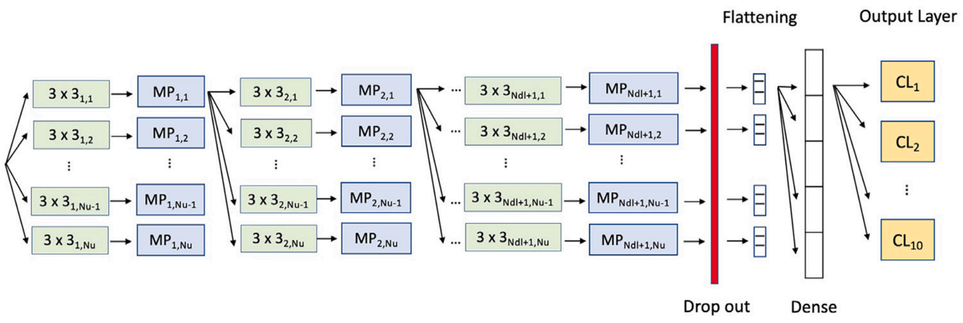

Fig. 10. The general CNN structure used in the proposed methodology.

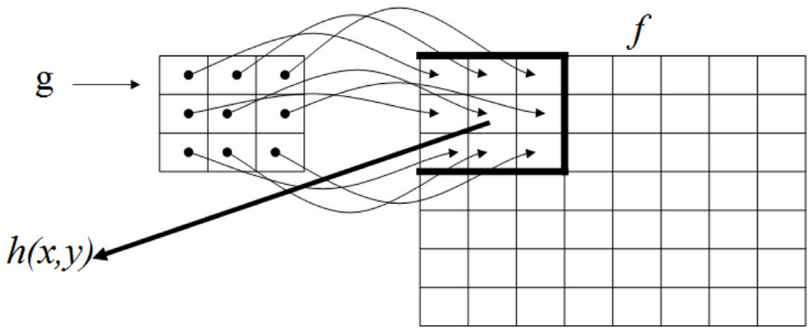

Fig. 11. Basic convolution scheme.

2.3.3. *Laplacian of Gaussian filter*

Adjustments can also be performed with a combination of filters. The strategy at this point is to use a Laplacian operator to enhance fine details and the gradient for prominent edges. The Laplacian of Gaussian filter (Dougherty, 2020) $LoG(x,y)$ of a 2D image $f(x,y)$ for (x,y) pixel is given by:

$$g^{LoG}(x,y) = -\frac{1}{\pi\sigma^4}[1-\frac{x^2+y^2}{2\sigma^2}]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The Laplacian operator is very effective for fine adjustments because it is a second derivative, but it also enhances the noise. The gradient has a lesser action in plain areas where noise is more unwanted, having a greater action in areas of significant transitions. The gradient action is less than the Laplacian for fine adjustments and noise while can still be smoothed out with a mean filter. In this case, the average filter to be used is the Gaussian one, which acts like a weighted mean filter (Gonzalez & Woods, 2008).

2.3.4. *Median filter*

2D median filtering (Pratt & Wiley, 1978; Tukey et al., 1977) is a non-linear order statistic filter, where the resulting pixel $h(x,y)$ is replaced by the median value of the pixel being analyzed and its neighbors

$$h(x,y) = med\begin{pmatrix}(x-1,y-1) & (x,y-1) & (x+1,y-1)\\(x-1,y) & (x,y) & (x+1,y)\\(x-1,y+1) & (x,y+1) & (x+1,y+1)\end{pmatrix}$$

The median of a set of values, by definition, is the value at which half of the values are greater than or equal to it and the other portion is less than or equal to it. For this reason, the filter is widely used for noise removal, as noise tends to be positioned at the ends of the range, that is, being higher or lower than most of the values in the set.

2.3.5. *High pass filter*

The high pass filter (or low band reject) is a filtering done in the frequency domain Jensen (1986), that is, the intensity of the channel is converted to the frequency domain (using the Fast Fourier Transform, for example) in which the pass or reject filter is applied and later the image is converted back to spatial domain. When filtering is performed at high frequencies, the effect caused is edge sharpening.

Another way of performing edge sharpening is to operate on the Gaussian distribution, as plains tend to be areas where the points have not changed much, the Gaussian filter does not cause much distortion on them, but on the edges, as they are distinguished from the remainder, the Gaussian filter causes distortion.

So when the original image is operated with the Gaussian filtered image, the result is an image with sharpened edges. These operations can be performed by convoluting the matrix

$$\begin{bmatrix} -1 & 0 & -1 \\ 0 & 8 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

which includes the Gaussian distribution and the subtraction of the Laplacian operation in the same matrix.

3. Related work

3.1. Publicly available bee sound datasets

The Zenodo (Nolasco & Benetos, 2018) dataset is an open and labeled dataset containing two classes of audio recordings, those belonging to bees and those not belonging to bees. This dataset can be used to build classification models for distinguishing bee noises from other noises.

The Open Source Beehives (OSBH) project (Smith, 2021) shared a freely accessible dataset labeled by the states of the hives. Recording and labeling were performed by several professional and amateur beekeepers. However, they have used different tools for recordings which makes the development of a uniform data processing method difficult.

Mukherjee (2018) shared a publicly available dataset containing bee, cricket and ambient noise audio data. Favre (2020) has created an open access dataset that contains 6 recordings of honeybees on New Year's Eve 2019.

Several researchers have recognized that the small amount of freely available datasets hinders the research and development of the area of bee audio analytics (Al-Turjman, 2019; Badenhorst & de Wet, 2019; Terenzi et al., 2020; Zgank, 2020, 2021), a factor which needs to be changed.

3.2. Related work on bee audio FE

Audio analysis of honeybees has been in focus since the mid of 20th century. Pre-processing, including FE, is a particularly important step in the entire process. Seasonal characteristics using frequency decomposition with a 10-filter standard audio analyzer were observed by Dietlein (1985), identifying three relevant frequencies, such that 300, 410 and 510 Hz. A research to differentiate between European and African honeybees using Fourier analysis was introduced by Kerr et al. (1988), noticing that the European honeybee signature has a fundamental power peak between 210 to 240 Hz while the African honeybee signature has a fundamental power peak in the 260 to 290 Hz range. The process of swarming was studied and the audio signal was decomposed to frequencies in Ferrari et al. (2008), to identify that swarming is indicated by an increase in the power spectral density at about 110 Hz and, approaching to swarming, the sound augments in amplitude and frequency to 300 Hz. Frequency analysis for hive monitoring with the help of Fourier transformation was used by Brundage (2009). Wavelet transform for colony collapse disorder analysis was used in Mezquida and Martínez Llorente (2009). Linear predictive coding (LPC) has been done to classify "queenless" and "queenright" states by Cejrowski et al. (2018). Bromenshenk et al. (2007) used fast Fourier transform and created a complex monitoring system to identify the health state of a hive. Robles-Guerrero et al. (2019) used MFCC to identify sound patterns in queenless bee colonies, while MFCC, Mel spectrogram and Hilbert–Huang transform (HHT) have been utilized to identify swarming and the presence of a queen in a hive by Nolasco et al. (2019). HHT has been used for monitoring the beehive's condition using temperature, humidity, CO2, hive weight, and external weather data in Cecchi et al. (2018), while (Terenzi et al., 2019) applied HHT on both STFT and MFCC for non-invasive monitoring of health condition of a beehive. S-transform and FFT (Stockwell et al., 1996) was used by Howard et al. (2013) to identify queenless states in beehives. Ramsey et al. (2017) identified that a signal similar to a stop signal occurs quite frequently by using simple discriminant function analysis on the scores of principal component analysis (PCA) (Wold et al., 1987). MFCC and LPC features have been used for swarm detection in an IoT-based classification framework (Zgank, 2020, 2021) utilizing a hidden Markov model (HMM). Two comprehensive studies have been presented by Kulyukin et al. (2018) and Amlathe (2018), respectively, using various features such that MFCC, Chroma, Mel Spectrogram, Spectral contrast, STFT and Tonnetz (first described in Euler (1739)), to separate ambient noise, bee buzzing, and cricket chirping.

3.3. Related work on bee sound classification

Several researches have already dealt with the distinction between different types of sounds, e.g. speech, music and environmental sounds (Purwins et al., 2019). Among these studies, several have dealt with the classification of bee sounds and other natural sounds. Comprehensive studies using various feature types were introduced by Kulyukin et al. (2018) and Amlathe (2018) using classification via logistic regression (LR), random forest (RF), k-nearest neighbor (kNN) and support vector machine (SVM) classifiers and, also, CNN (the traditional ML techniques, other than CNN, were executed with the one-vs-rest classification approach to separate ambient noise, bee buzzing, and cricket chirping). Singular value decomposition (SVD) for a multi-label classification problem has been utilized to identify similarities in queenless states in a strong and in a weak hive by Robles-Guerrero et al. (2019). Nolasco et al. (2019) used SVM with RBF kernel and CNN to identify hives with or without a queen. Self-organizing map (SOM) was used in Howard et al. (2013) to separate the sounds of hives with and without a queen. SVM and CNN classifiers have been utilized by Terenzi et al. (2019) for a non-invasive health monitoring system to separate "orphaned bees" and bees with a queen. Gaussian mixture model (GMM) and HMM were used in Zgank (2020) and Zgank (2021), respectively, to separate audio samples from swarming and normal conditions of a bee colony. Bae et al. (2016) introduced a parallel combination of CNN and LSTM for acoustic scene classification, achieving approx. 85% accuracy. LSTM, multi-layer perceptron (MLP) and LR have been used in Ruvinga et al. (2021) to distinguish hives with "queen-absent" and with "queen-present" states.

3.4. Related work on noise filtering in bee sound analytics

The articles related to bee audio analysis which include some NF on the audio data (Dietlein, 1985; Ferrari et al., 2008) have focused solely on filtering out relevant frequencies outside the range of 0 Hz and 2 kHz. Other researches (Cecchi et al., 2018; Nolasco et al., 2019; Terenzi et al., 2020) have utilized two microphones in their approach: one for internal sounds of the beehive and another one for external sounds (like wind noise, passing vehicle noise, etc.) to filter out the sounds not belonging to the beehive.

4. Dynamic audio noise filtering

As our results show (Section 7), when using the introduced FE methods, the selected state-of-the-art ML algorithms performed poorly on the collected data (see Section 5). The reasons for it might be the following:

- STFT, one of the most common and applicable FE methods, is a domain-independent technique with a linear resolution, meaning that the bins are evenly spaced. Although the technique works very well on a large set of problems, it does not work efficiently on bee sound data since the relevant information can be extracted from the low frequencies. Human hearing has much better resolution in the lower frequencies than in the higher ones.

- The above described recognition is applied by MFCC where the resolution is exponential. However, MFCC features are sensitive to background noise of which the bee audio recordings contain a lot. Additionally, the quality of extraction heavily depends on the number of bins.
- MFCC delta is examining the change of MFCC coefficients, however, bringing the disadvantages mentioned for MFCC features.
- Chroma features represent the tonal content of an audio signal in a solid form and perform well in high-level semantic analysis, like chord recognition or harmonic similarity estimation, but results show that these examinations can be performed with only strong limitations in such a noisy environment.
- Other baseline methods, like SC or ZCR, are just simple descriptors of an audio signal and are only suitable for analysis of simple features.

Moreover, also referring to our results (Section 7), using the introduced NF methods did not improve the performances of the selected state-of-the-art ML algorithms as much as we expected. This can be mainly because of the following reasons:

- High pass filtering is also called sharpening in image processing. If the intensity value of a pixel is the same as the intensity of its neighbors the filtering does not change the image. If the intensity is changing between a pixel's intensity and one of its neighbor's intensity, high pass filtering increases the difference between the two intensity values, resulting in a greater and sharper intensity change. This filter is not changing fundamentally the characteristics of an image, just sharpening its intensity changes.
- Median filtering is a filtering method that preserves sharp edges and filters out spike-like noise ("salt and pepper" type noise) quite effectively (Justusson, 1981). If the noise is not spiky, but has expanse, median filtering is not effective.
- Gaussian filtering or Gaussian smoothing is very effective for removing Gaussian noise, a noise that has a density function similar to the normal distribution. The weights give higher significance to pixels near the edge (reduces edge blurring). If the noise is not Gaussian noise, the filtering is just moderately effective.
- Laplacian filtering is using the second derivative of intensity i.e. highlights regions of rapid intensity change and is therefore often used for edge detection. The method is sensitive to noise.
- The Laplacian over Gaussian (LoG) filtering is Laplacian applied to an image that has first been smoothed with a Gaussian smoothing filter in order to reduce its sensitivity to noise. If the noise is not Gaussian then the result of LoG is moderate.

Motivated by the analysis of limitations of state-of-the-art FE and NF methods, we propose a simple NF method for bee audio data. The proposed heuristic is called DANi, as an abbreviation for "*D*ynamic *A*udio *N*oise F*i*ltration", requiring a spectrogram on its input, as illustrated in Fig. 3. DANi consists of the following 4 steps:

(1) Z-score normalization. The first step is to perform a Z-score normalization on the spectrogram-per-time window. All coefficients of different frequency components belonging to the same time window are handled as one array of values. Fig. 12 shows the absolute values of the normalized coefficients of the time window at 1.51 min and the absolute value of the normalized coefficients for the whole signal as well.

(2) First thresholding. The Z-Score normalized data is thresholded. All values below the mean, i.e., negative values after normalization, are considered as noise and are replaced by 0, as shown in Fig. 13.

(3) Smoothing. In the next step, the idea that the main characteristics of the signal are mainly determined by the low-frequency components is taken into account. These components define a signal with similar characteristics but significantly smoother and simpler, while high-frequency components are responsible for signal refinement. The next step is to take a Fourier Transform (DTFT) of the normalized and thresholded data again, keeping only the coefficients belonging to the first k components. The coefficients belonging to the rest of the frequency components are set to 0. A result of the smoothing step is illustrated in Fig. 14. The HP of this step is the number of selected components k.

(4) Second thresholding. To overcome the distortion of the signal coming from the smoothing step, the final step is another thresholding according to the mean of the values-per-time window. Each value smaller than the mean is set to 0, and all values above the mean remain unchanged, as shown in Fig. 15.

5. The naturami dataset

It is essential to have a sufficiently large and labeled dataset for research purposes in the area of honeybee audio analytics. Such a database should include audio recordings of sufficient length and similar quality. The reason is that, unfortunately, if the quality or the length of recordings contained in different databases or even within one database differ significantly, the usage of these datasets for audio analytics research is cumbersome.

Data has been collected in one Hungarian apiary (see Fig. 16) by our partner, the Natura Mérnökiroda Kft.,[5] a Hungarian SME working on IoT technologies for Agriculture. The beehives of the same type (made from extra hardened, expanded polystyrene) were located near to each other. All hives were sensed with a separate and externally installed microphone, illustrated in the bottom right picture in Fig. 16. Microphones were managed by a micro-controller (top right picture in Fig. 16). The microphones were taking an 8.2 s long recording every 20 minutes and each recording has 65625 samples sampled at 8 kHz. The microphones recorded audio data in pulse code modulation (`pcm`) format ITU (1988) which is converted to `wav` file with the help of the Python `wave` library (Python Software Foundation, 2021) using the following settings: number of channels = 1 (mono), sample width = 2 and frame rate = 8000.

From all the recordings we have selected 700–1050 samples from each of the 10 hives with matching recording times, i.e. from 7 to 10 recordings in the same time.[6]

The collected data,[7] 10000 recordings, are large enough to provide sufficient support for further research in the area of bee sound analytics. Opposite to OSBH data, our recordings were gathered in the same apiary, with the same types and orientation of hives, the microphones were placed in similar places within the hives and each batch of (7 to 10) recordings correspond to the same time interval and weather conditions. Also, even looking a bit "synthetical", the 10 labels, corresponding to the 10 hives, provide opportunities for conducting large-scale experiments with multi-label classification as well as various clustering or pattern mining approaches. Besides multi-label classification, after selecting a subset of the data, the Naturami dataset can serve for further research in binary classification models for bee audio data as well.

6. Experiments

Experiments have been done according to the methodology presented in Section 1 and illustrated in Fig. 1.

[5] https://naturami.hu/

[6] The goal was to have one record per hive for the same time, i.e. 10 recordings per time slot. However, due to technical issues, it was not always possible.

[7] Available at https://zenodo.org/record/7052981#.YxnJ3NKxVD8.

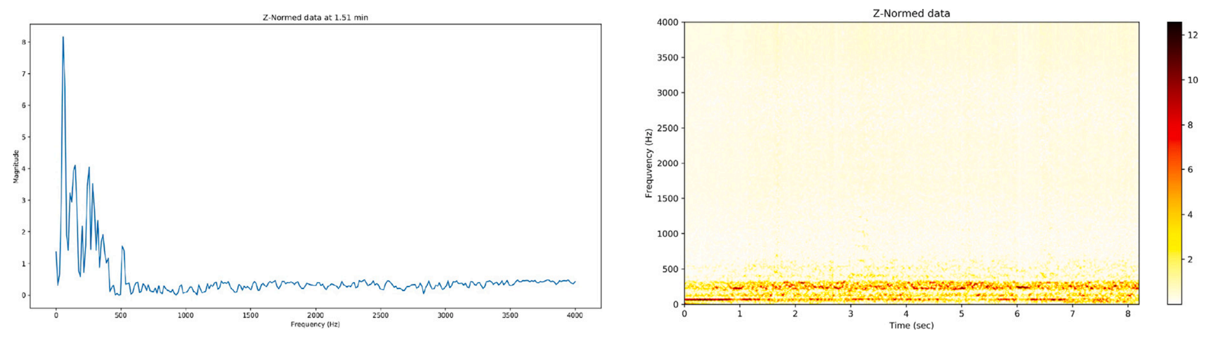

Fig. 12. Z-Score normalized values (of the data from Fig. 3) at the time 1.51 (on the left) and the whole Z-Score normalized data (on the right).

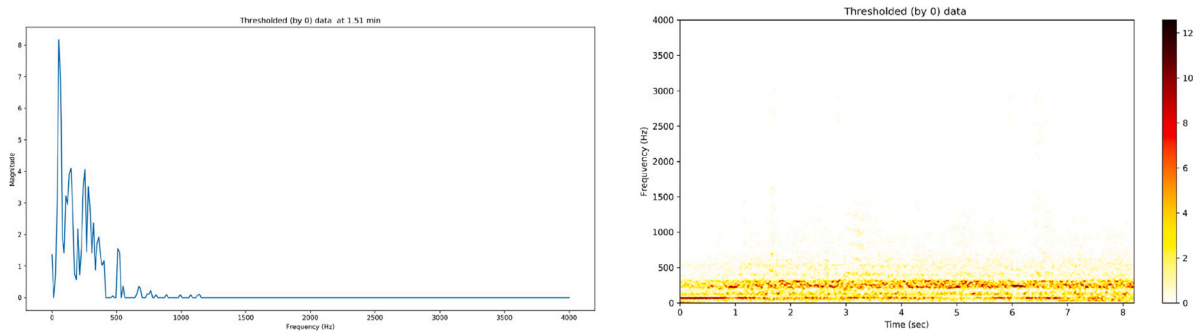

Fig. 13. Thresholded components (of the normalized data from Fig. 12) at the time 1.51 (on the left) and the whole thresholded data (on the right).

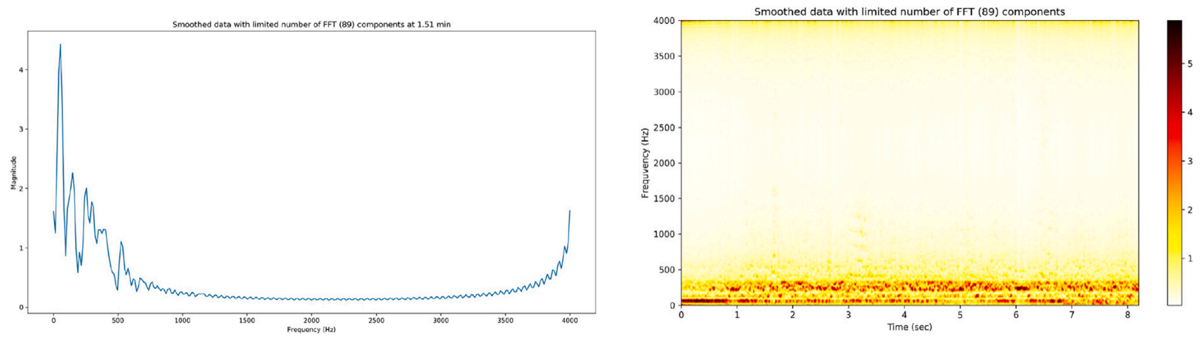

Fig. 14. Smoothed data (of the thresholded data from Fig. 13) at the time 1.51 (on the left) and the whole smoothed data (on the right).

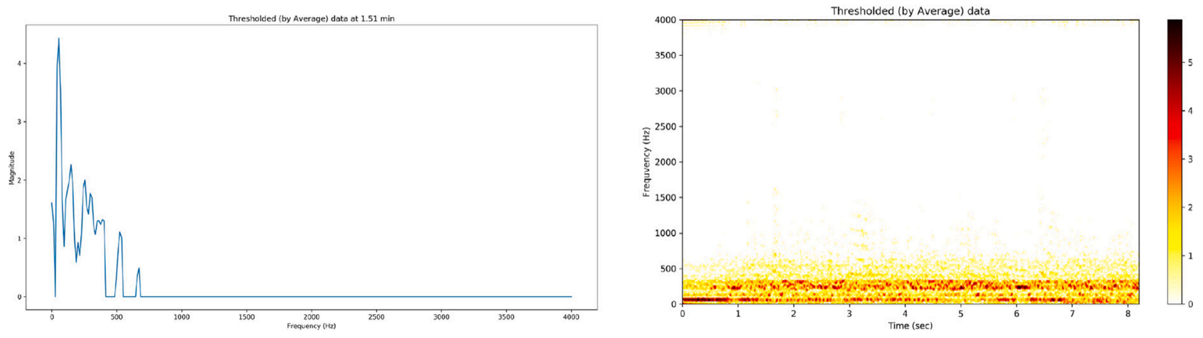

Fig. 15. Thresholded data (of the thresholded data from Fig. 14) at the time 1.51 (on the left) and the whole thresholded data (on the right).

6.1. HP settings

The candidate HPs have been selected to meet the objectives of this study, i.e. to identify the best performing bee audio FE methods and related ML techniques while keeping in mind the simplicity of the used models. The requirement of simplicity is due to the fact that, in many cases, bee audio analytics should run on remote sites (hives located near the fields with blooming crops, e.g. sunflower) with limited access to the internet and/or electricity. Thus, models suitable for offline computations conducted on "small" devices (e.g. a Raspberry PI) are desirable considering the trade-off between performance and complexity.

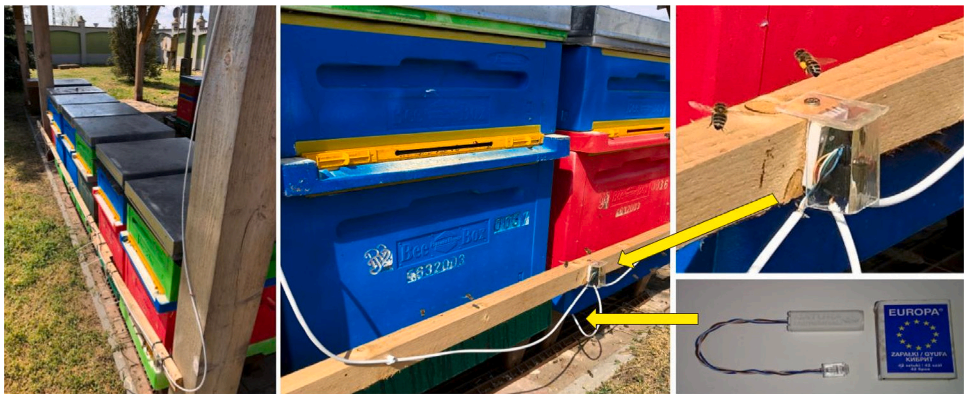

Fig. 16. The data collection site and the audio recording devices.

Due to the large number of HPs appearing in the used analytics workflow (Fig. 1) and the computational demand of the majority of FE, ML and NF methods, conducting an exhaustive search of the HP space would be intractable given the available computational resources. Thus, the HPs of various FE, ML and NF methods have been tuned using the following heuristic: Those HPs which are considered to be the most influential on the performance have been tuned using grid search.[8] The other HPs have been set either according to the information from the literature or empirically, based on initial runs of the related FE, ML or NF methods.

6.1.1. HPs of FE methods

FE was controlled by the following common HP values, determined empirically based on information found in the literature, basic principles of these methods and the main characteristics of bee sounds:

- The selected *windowing type* is Hann windowing;
- *Hop length* was set to 500 samples, which represents 1/16 seconds offset for the windows;
- The *window* length was set to 1/8 seconds which means that there is a 50% overlap between the windows;
- The *number of features* was set to 499 for STFT and DANi, 128 for MFCC and MFCC Delta, 12 for Chroma filtering and 1 for SC and ZCR;
- Since the length of original WAV data is 65625 and hop length was set to 500, the *number of windows* for all feature sets is $\lfloor 65625/500 \rfloor + 1 = 132$.

We decided to choose a very detailed representation of the audio data, thus, the number of features for STFT and DANi filtering is set to 499. Theoretically, the number of features is unlimited, but in the used `Librosa` implementation the number of features for MFCC and MFCC delta is limited to 128. The generally used values are between 10 and 30. Chroma results in 12 different features, while SC and ZCR describe each window by one number (1 feature).

Although there is a space for fine-tuning these HPs, the used settings result in detailed enough features to provide sufficient representations of bee audio data (according to our expectations, the performance of various FE-ML combinations depend more on the type of the used FE method than on its HP setting).

6.1.2. HPs of ML methods

According to our experiments, the HPs of the used ML methods have a considerable influence on the performance of a given FE-ML combination. Thus, we have been focusing on HP tuning of the used DL methods utilizing grid search.

[8] In grid search, a systematic search over all the combinations of pre-specified values for each HP is conducted.

Table 1
Number of samples (in the 2nd row) per labels (hive numbers, in the 1st row).

1	2	3	4	5	6	7	8	9	10
1038	1020	1046	1036	1013	702	1023	1023	1044	1055

- Following the thought on the trade-off between the performance and complexity, the number of deep layers n_{dl} were set to 3, except for the Chroma-CNN combination where n_{dl} was chosen to 1 (the low dimensionality of the input data makes a simpler structure sufficient). Even such low number of deep layers resulted in a structure that reached the 16 GB memory limit set up for simulating capacities available on remote sites;
- The number of units n_u, in case of CNN, has been chosen from the set of values $\{1, 2, 4, 8, 12, 16, 20, 25, 50\}$ while, in case of LSTM, n_u has been chosen from the set $\{1, 2, 4, 8, 16, 32, 64, 128, 256\}$;
- Two optimizers, SGD and $ADAM$, have been used in grid search for both CNN and LSTM;
- The out activation functions act_o have been either *Sigmoid* or *Softmax* for both CNN and LSTM. In case of CNN, the deep activation function act_d was either *ReLu* or *Sigmoid*;
- The remaining two HPs, the number of training epochs and the batch size, not related to the architecture of the network, were set as follows: the number of training epochs was set to 500 for LSTM and 100 for CNN, while the batch size was set to 100 for both LSTM and CNN.

6.1.3. HPs of NF methods

- The High pass filter's *cutoff frequency* was set to 150 Hz (by counting with some tolerance since, based on the literature, the lowest interesting frequency for drone buzzing is 190 Hz) and the *order of the "Butterworth" filter* was set to 1;
- For both Gaussian and Laplacian of Gaussian filters the *sigma* HP was set to 0.5;
- Median filter is controlled by the *radius* (in which the median is calculated) which was set to 2;

These settings seemed to be the most reasonable based on the literature review.

6.2. Data

The dataset contains 10000 audio recordings with 10 different labels the distribution of which is shown in Table 1. The dataset is considered to be balanced, and therefore no balancing technique has been applied.

6.3. Grid search and validation

As mentioned before, a grid search has been performed to tune the HPs of the used ML methods for each FE-NF-ML method combination in the used workflow, while the HPs of the involved FE and NF method have been set up empirically. The following process, illustrated in Fig. 17, has been implemented:

For each HP setting of the used ML model, a separate scenario was executed: The 10000 recordings were divided into k folds such that one fold (marked by gray color in Fig. 17) was used for testing the performance of the ML model trained on the other $k-1$ folds (marked by white color in Fig. 17). Allocation of audio recordings into the k folds has been made in a stratified manner, to avoid an imbalanced distribution of labels in the resulting folds.

In each of the k folds, in every training iteration (using the given train folds) the accuracy of the actual ML model has been evaluated by the *accuracy* metric expressing the ratio of correctly classified recordings to all recordings in the given test fold (i.e. the percentage of recordings correctly labeled by the number of hive to which they belong). The maximal values of the accuracies throughout the iterations have been registered, resulting in k (maximal accuracy) values corresponding to the k folds. To keep track of the maximal accuracies achieved throughout the learning process might resemble a kind of "early stopping", well-known in ML, and is chosen because the purpose of the experiment is to determine the achievable best performing FE-NF-ML combination. At the end of the grid search process, the final scenario (HP setting) chosen for reporting (see the results in Section 7) is the one the maximal accuracies of which reached during the iterations in the k folds were the best in average. In the results, besides the average (AVG), also the minimal (MIN) and maximal (MAX) values as well as the standard deviation (STD) of maximal accuracies achieved during the training process are reported.

Altogether, 72 unique scenarios (HP settings) for each 2D feature type (for CNN), and 36 unique scenarios (HP settings) for each 1D feature type (for LSTM) have been launched for each FE-NF-ML combination, defining unique deep network structures and training processes.

Memory usage of CNN is significantly lower than of LSTM with comparable architecture. Thus, the value of k differs for CNN and LSTM. Computations in the different folds have been executed in parallel with shared memory used to store input dataset between the folds. Classification via CNN was executed with 10 folds, while classification via LSTM was performed only with 5 folds.

7. Results

7.1. SC and ZCR results

The performances of the two FE methods resulting in time-series, SC and ZCR, in combination with the LSTM model and no NF method (as discussed in Section 3.4) are plotted in Fig. 18 on which (according to Fig. 17) the maximal accuracies found during the learning epochs averaged on the 5 folds are depicted. All the statistics, such that the AVG, MAX, MIN and STD of these maximal accuracies are listed in Table 2. The accuracies are plotted against the number of units in the LSTM model to better express the complexity of the related models.

The results related to SC and ZCR show the poor performance of these FE methods for the given multi-class classification problem of beehive audio data, even in case of relatively complex ML models. Moreover, SC became competitive to ZCR only in case of large ML models.

Since the achieved performance using these FE methods, ranging from 10 to 20% accuracy, is not satisfactory, the remaining experiments, introduced in the next Section 7.2, were focused on FE methods resulting in spectrograms.

Table 2
Statistics for SC and ZCR FE in combination with LSTM and no NF (in %).

FE type	Statistics	Number of units						
		4	8	16	32	64	128	256
SC	MIN	10.1	10.3	10.1	10.3	19.3	19.0	20.8
	AVG	11.0	11.0	15.2	18.4	20.3	20.9	21.2
	MAX	13.0	11.7	21.4	21.2	20.8	22.4	21.6
	STD	1.0	0.5	5.0	4.1	0.5	1.2	0.3
ZCR	MIN	20.7	20.9	21.4	21.3	20.1	19.0	20.6
	AVG	21.4	22.3	22.8	22.6	21.8	21.8	22.2
	MAX	22.0	23.5	24.9	24.2	23.5	23.3	23.5
	STD	0.5	0.9	1.1	1.0	1.3	1.6	0.9

7.2. Spectrogram-type FE results

Experiments have been done with 28 different FE-NF-ML method combinations, including 4 (STFT, MFCC, MFCC Delta and Chroma) FE and 6 (Median, High Pass, Gaussian, Laplacian, Laplacian of Gaussian and the proposed DANi) plus 1 ("No NF", as usual in the recent literature) methods combined with the CNN ML model. For each of the 28 combinations the before described HP tuning and result reporting were performed (see Fig. 17), resulting in 12 weeks of computing time, including running the FE and NF run-times as well as the time needed to provide the grid search for tuning the (HPs of) CNN on a computing server with a GPU (Nvidia Tesla v100 16 GB).

The best accuracies, according to the reporting methodology (Fig. 17), are depicted in Fig. 19. More detailed statistics including MIN, MAX and STD, can be found in Tables 4, 5–7 in the Appendix.[9]

It is important to mention that the reported accuracies are the maximal accuracies reachable by "early stopping" when fine-tuning a CNN model with the given number of units in its hidden layers and the optimal HP settings found by grid search over various HP values (see Section 6.1).

In general, utilizing spectrogram-type FE methods (with CNN) resulted in higher accuracy values than using FE methods resulting in time-series (with LSTM). This is, most probably, because of the richer 2D representation of the audio data by spectrograms (resulting from the MFCC, MFCC Delta, STFT or Chroma) than their 1D representation by time-series (resulting from SC or ZCR).

Chroma FE performed the best from the baselines, followed by MFCC and STFT while using MFCC Delta achieved the worst results. However, according to the statistics from Tables 4, 5, 6 and 7, STFT seems to have the highest variance amongst the baselines.

Results show that using the proposed DANi NF method leads to performances that are superior to every baseline by a large margin (about twice as good), even in case of very simple models consisting of only 8 units per layer.

8. Discussion

Several questions might arise regarding the proposed DANi NF method, which are discussed in this section pointing out relevant issues to be taken into consideration during its deployment and further development.

8.1. Efficiency

Compared to the baseline FE methods resulting in a 2D output (spectrogram), each starting with computing the STFT of the input audio data (time-series), DANi has an additional DTFT step. This step, however, is performed on smaller data (normalized and thresholded)

[9] The values in Fig. 19 are the AVG values from Tables 4, 5, 6 and 7 in the Appendix.

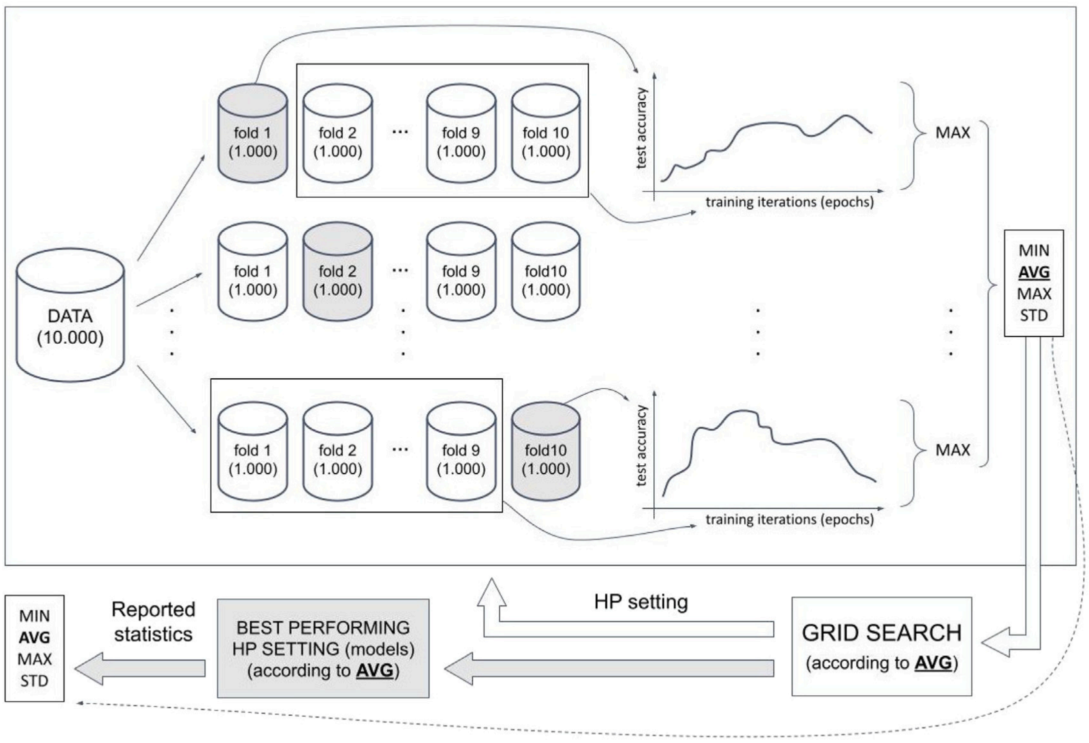

Fig. 17. The used methodology for reporting the results for each FE-NF-ML combination in the used audio data analytics workflow (see Fig. 1.).

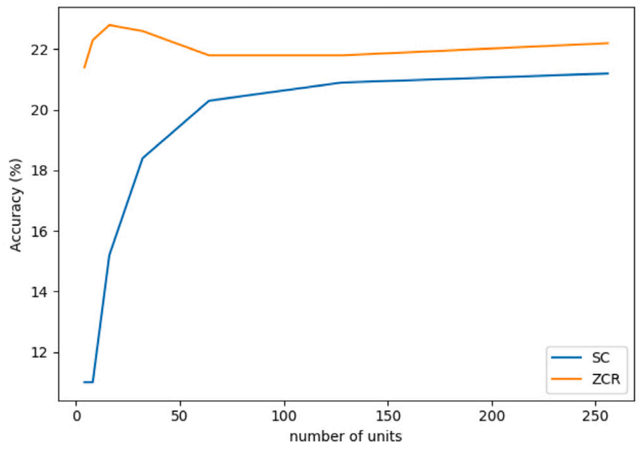

Fig. 18. Results for the SC and ZCR FE in combination with LSTM and no NF.

and, thus, can be computed efficiently. Results showed that the difference between the efficiency of DANi and the other baseline FE methods were similar when using the same HPs (number of frequency components, length of a window, window type and hop length).

8.2. Slicing

A fundamental question when using DANi is by which axis do we perform normalization and thresholding? Normalizing by time slices, i.e. windowing according to the time axis, means that all coefficients belonging to the same time window, but different frequencies, are handled as one unit and express the intensity change through all the frequencies at a specific time window (see the left side of Fig. 20). On the other hand, normalizing by frequency values (see the right side of Fig. 20) means that the signal is divided into separate frequency windows, handled as one unit while looking at them as a function of intensity change through all time steps at a specific frequency window.

After the normalization step, the relationship between the coefficients from different slicing units (time or frequency slices) fundamentally changes while the relationship between the coefficients from the same unit still remains.

If the chosen policy is time slicing then the ratio of coefficients of a certain frequency before normalization and after the normalization may completely change due to the fact that the average of the normalizing (time slice) unit can be different. Normalizing via time slices means that those frequencies in which there are no significant activities due to their small coefficients are considered as irrelevant frequency ranges or, in other words, noise.

If the data is normalized via frequency values (frequency slices), then a value around, below or above the mean means usual, irrelevant or relevant activity, respectively. For example, a bee queen is “piping” around 400–500 kHz frequency (Collison, 2018). If the goal is to find such an activity when the queen is active, looking at a certain frequency window, frequency-wise separation can be a good approach.

To decide on which approach is more suitable, i.e. time or frequency slicing strongly depends on the goal of the analytics and the characteristics of the signal. For example, if there is an activity at all the frequencies and the maximum draws a similar value over time, then the frequency slicing leads to good results, as shown in Fig. 21.

On the other hand, if there is an activity only at certain frequency ranges, and the respective coefficient values belong to a narrow scale, then using frequency slicing can change the characteristics of the signal so that it becomes essentially meaningless, as shown in Fig. 22.

Audio signals of bee hives show activity only in the lower frequency ranges so DANi filtering via time slicing is the good approach for audio FE and information highlighting. On the other hand, frequency slicing can be a good approach if the goal is to find activities in certain ranges, for example searching for queen activities knowing the frequency ranges where these can be found.

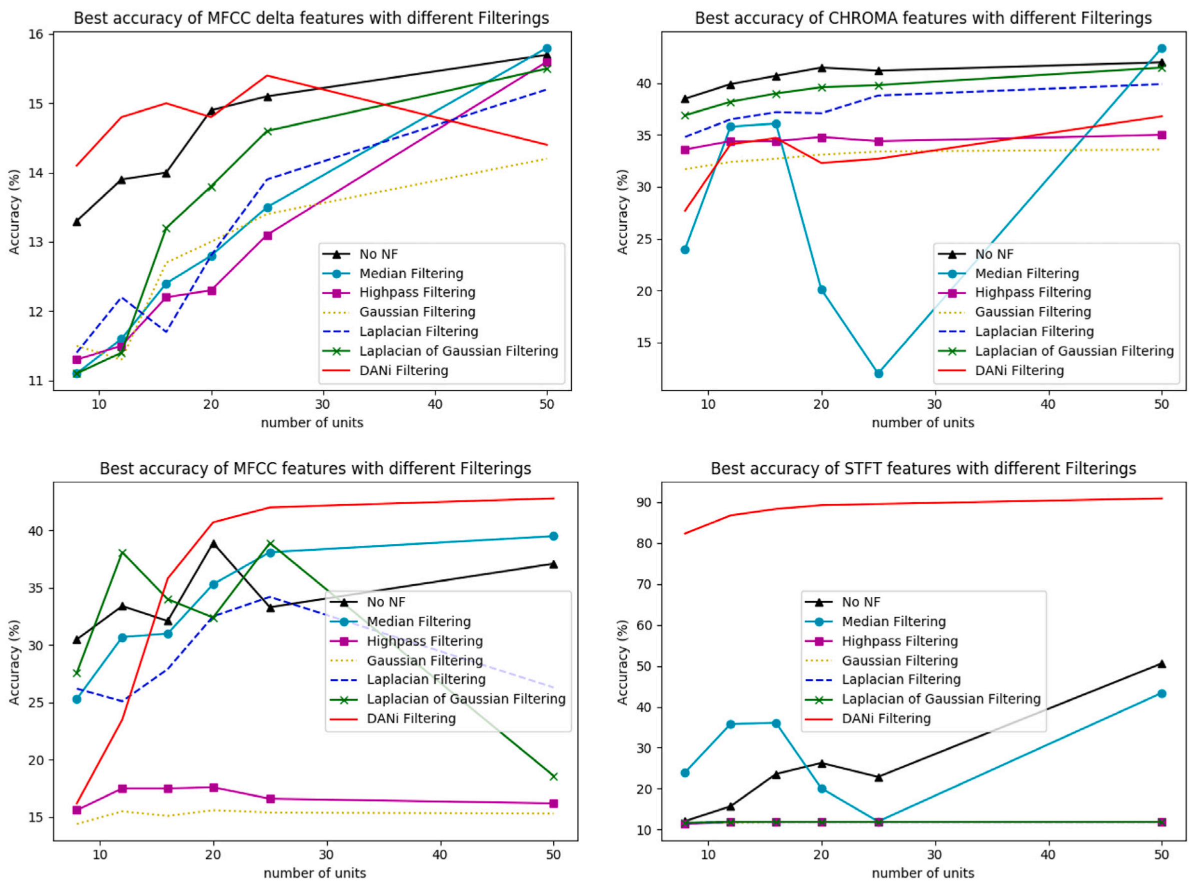

Fig. 19. Results containing the averaged best accuracies (vertical axes) over the learning of the used CNNs with various number of units (horizontal axes) in their inner layers, grouped by the 4 used FE methods (STFT, Chroma, MFCC and MFCC Delta).

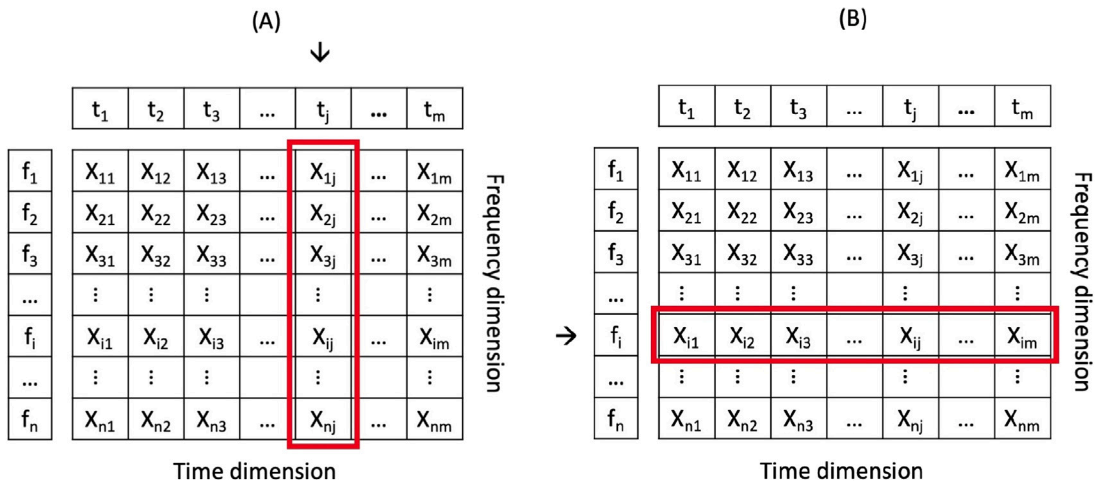

Fig. 20. Normalizing spectrograms by time slices (A) and by frequency values (B).

8.3. Smoothing

The smoothing step of DANi is controlled by one HP, that is the number of Fourier components not set to 0. Smaller values result in a smoother and simpler signal, while higher values result in a spiky signal which is closer to the original one. The smoother and simpler the representation of the signal, the greater the loss of information and greater the distortion. Figs. 23 and 24 show the result of smoothing in cases when 5%, 10%, 25%, 50%, 75% and 100% of the components have been kept.

If only a small proportion of the components are kept (5%–25%), significant distortion occurs and the characteristics of the signal change fundamentally after smoothing. Keeping approximately half of the components (40%–60%) will result in a signal with similar characteristics but a simpler representation. While using more than 60% of the components will not change the characteristics of the signal but make the representation more accurate and spiky. During our experiments, the value of 50% has been chosen, which kept the original characteristics but resulted in a simpler signal.

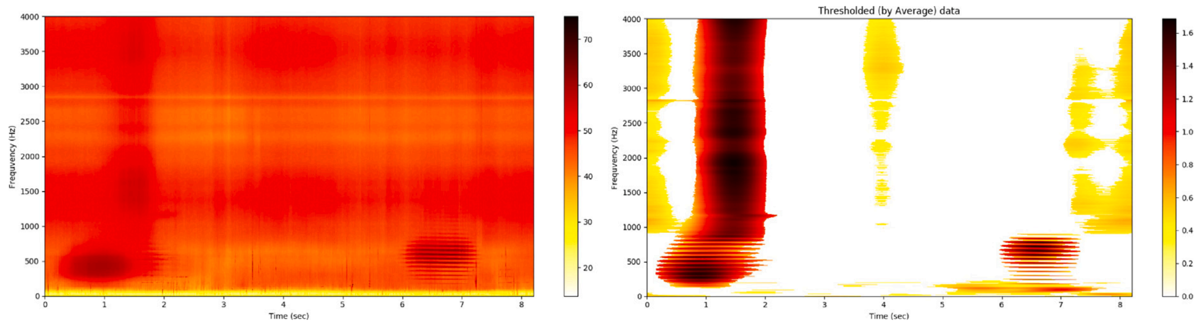

Fig. 21. Example of a sonograph image (left) pre-processed via frequency slicing (right).

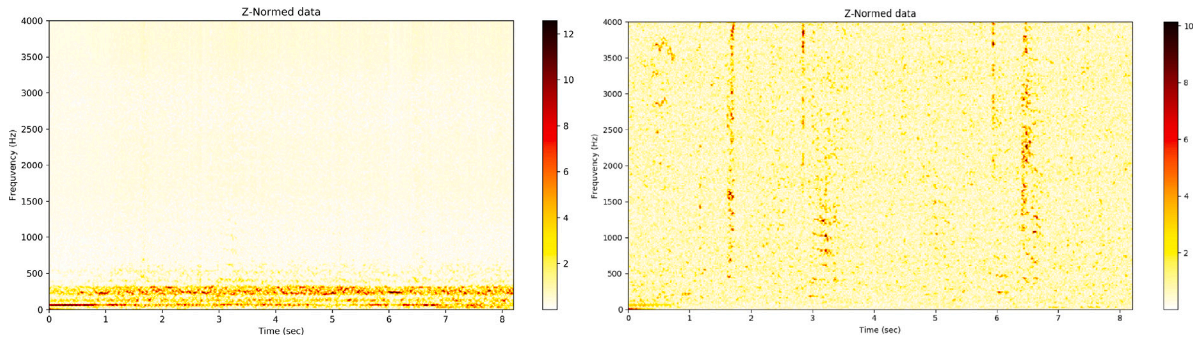

Fig. 22. The results of a bee audio sample using time slicing (left) and frequency slicing (right).

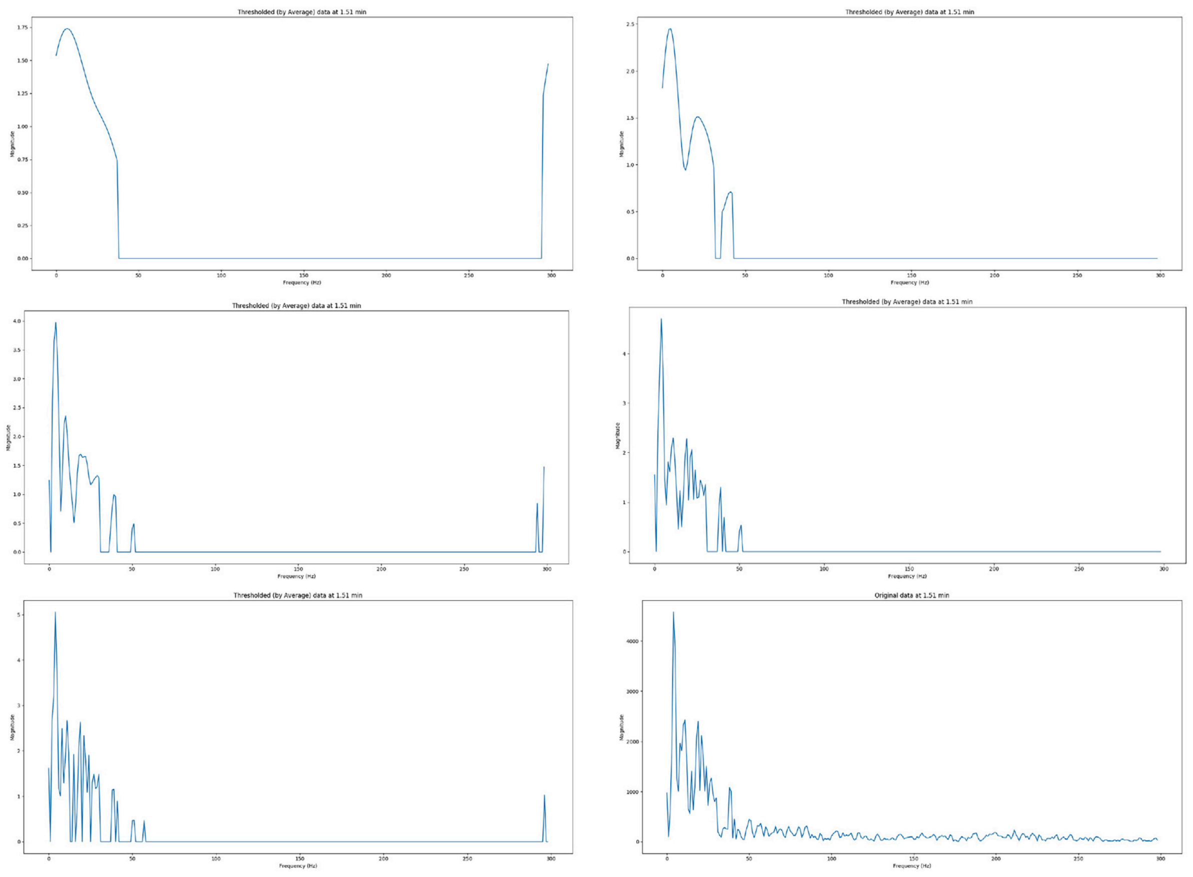

Fig. 23. Result of smoothing for the ratio of selected components set to 5%, 10%, 25%, 50%, 75% and 100%.

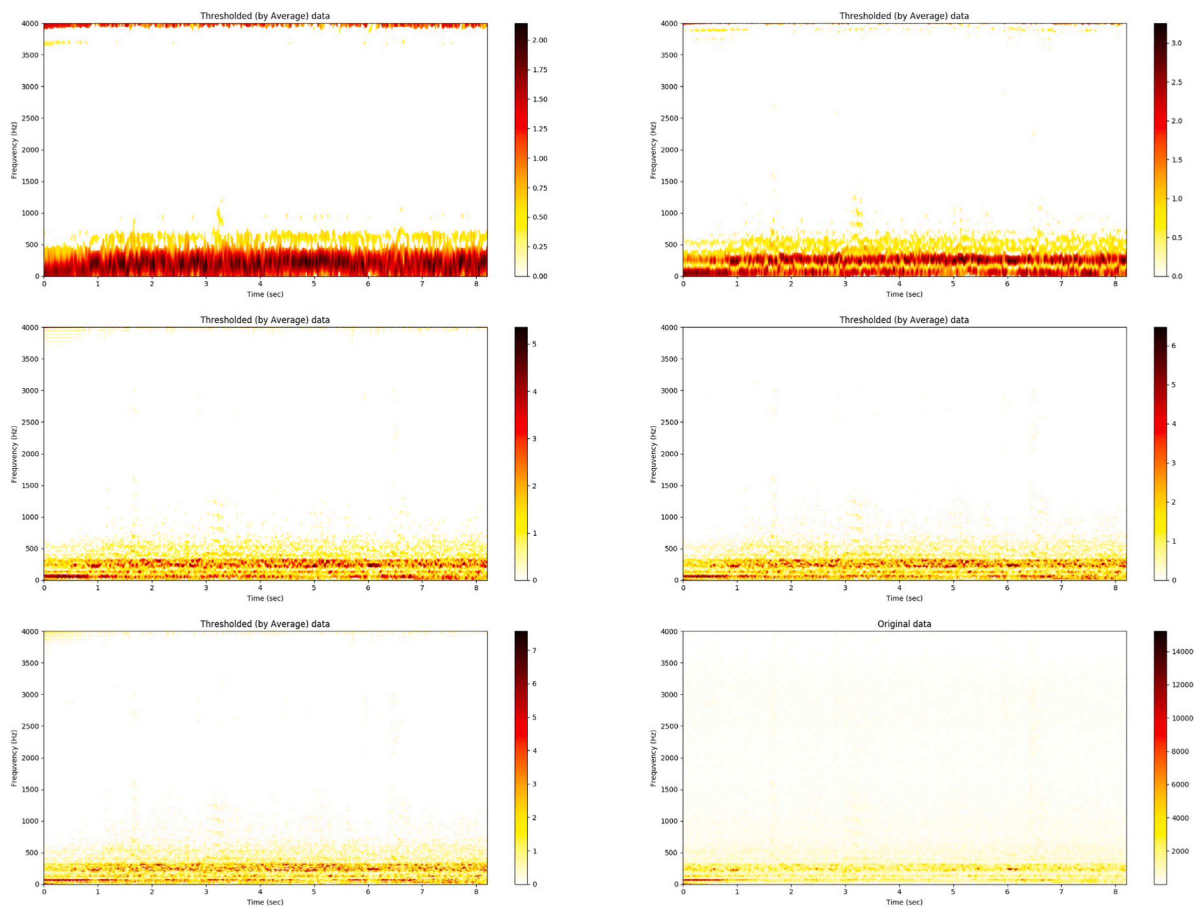

Fig. 24. Result of smoothing for the ratio of selected components set to 5%, 10%, 25%, 50%, 75% and 100%.

Table 3
Accuracy results (in %) and statistics for applying the combination of the Median NF followed by the Laplacian over Gaussian NF with different FE types.

FE type	Statistics	Number of units					
		8	12	16	20	25	50
STFT	MIN	9.7	9.7	10.4	10.4	9.8	10.0
	AVG	11.4	18.5	12.3	11.2	19.7	25.9
	MAX	12.7	56.3	22.3	11.8	94.6	89.8
	STD	0.9	14.5	3.4	0.5	25.0	30.6
MFCC	MIN	30.5	30.8	31.0	31.8	32.2	34.1
	AVG	33.0	34.0	34.7	35.8	37.0	40.1
	MAX	35.8	39.4	39.8	41.1	41.8	44.1
	STD	1.7	2.9	2.1	3.1	3.3	3.7
MFCC delta	MIN	12.9	9.8	10.4	10.6	11.4	9.9
	AVG	14.1	13.8	14.0	14.7	14.6	16.4
	MAX	16.7	16.7	16.3	17.7	16.6	18.2
	STD	1.1	2.1	1.6	2.1	1.4	2.3
Chroma	MIN	36.9	36.3	35.5	37.2	38.4	39.5
	AVG	37.7	38.5	39.3	39.4	40.4	42.0
	MAX	39.1	42.3	41.0	42.6	41.7	43.9
	STD	0.8	1.9	1.4	1.8	0.9	1.2

8.4. Combining NF methods

An important feature of DANi is to make use of elimination and smoothing processes, as the other common NF methods only eliminate or smooth (or sharpen).

In addition, two stages of elimination are used, but with a smoothing process in between. The first elimination step gets rid of noise before the smoothing algorithm spreads the values to its neighbors. The second step readjusts the elimination threshold to a smoothed distribution and gets rid of values that would not be eliminated in the first pass (after smoothing, values considered to be noise will be changed).

Furthermore, DANi uses global methods, that is, it uses information from the image as a whole. The Median filter, even though it is a good method for eliminating noise, only acts on its radius. Thus, information from other areas of the image, that could be useful for noise elimination, will not be used. This can help to explain the results obtained by combining the Median filter with the Laplacian over the Gaussian filter presented in Table 3.

The combination of the Median NF method followed by the Laplacian over Gaussian NF method can reach relatively good maximal accuracies when using the STFT FE method and more complex networks. However, the results of this combination are not as stable as in the case of utilizing DANi with STFT. According to the results, such a combination of two NF methods is inferior to DANi, on average. Since these two NF methods, being the best candidates for "simulating" the above described characteristics of DANi, did not reach the desirable (i.e. high and stable) performance, other combinations of NF methods have not been tested. It can be seen in Table 4 that using the Median NF method alone has reached better performance than its combination with the Laplacian over Gaussian NF method.

9. Conclusions

In this paper, a new method for audio noise filtering, called Dynamic Audio Noise Filtering (DANi), has been proposed that makes bee audio processing more effective. A large-scale experiment was conducted using 10000 bee hive recordings and state-of-the-art feature extraction, noise filtering as well as machine learning methods (including their hyper-parameter tuning).

The collected data are available at https://zenodo.org/record/7052981#.YxnJ3NKxVD8. The source codes, including the used feature extraction and noise filtering methods, the developed DANi method, the machine learning models as well as experimental evaluations are available at https://github.com/varkonyidaniel/behretims.git

Experiments showed that, while performing well for binary classification tasks, stat-of-the-art noise filtering methods performed poorly for multi-label classification of bee sounds. The proposed heuristic allows fast and efficient processing and, since it works well with simple neural networks, is suitable for its deployment also in off-line computation scenarios on small devices.

The classification models resulting from the presented audio analytics workflow (Fig. 1), utilizing the short-time Fourier transform feature extraction and the proposed DANi noise filtering methods, are able to identify a bee colony based on its sound with 90% accuracy. Since these models are small, they are easily interpretable, allowing to extract useful information from them related to the colonies which are addressed in the future work. The investigation of the connection between the resulting models from our experiments (and the "latent" information contained in them) and the so-called "colony fingerprints" introduced by Cejrowski and Szymański (2021) is the subject of future investigation. The ultimate goal for future work in this direction would be the development of "personalized" anomaly detection methods on a colony level, assuming that a colony's sound in case of swarming, missing queen, disease, pest, predator, exposure to chemicals, etc. bears unique characteristics.

Experimental results encourage further investigation of the usability of the proposed noise filtering method in other application domains as well, in which the noise resembles audio characteristics of bee buzzing, e.g. machinery vibration analysis and predictive maintenance (Scheffer, 2004).

CRediT authorship contribution statement

Dániel Tamás Várkonyi: Methodology, Software, Validation, Investigation, Data curation, Visualization, Writing – original draft, Writing – review & editing. **José Luis Seixas Junior:** Methodology, Validation, Investigation, Writing – review & editing. **Tomáš Horváth:** Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

"Application Domain Specific Highly Reliable IT Solutions" project has been implemented with the support provided by the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme. The authors would like to thank to Mr. Sándor Kerekes and Mr. Ruszlán Vladimír from the Natura Mérnökiroda Kft. for providing the audio recordings and the necessary domain knowledge on beekeeping.

Appendix

See Tables 4–7.

Table 4
Accuracy results (in %) and statistics for STFT FE with different NF types.

NF type	Statistics	Number of units					
		8	12	16	20	25	50
No NF	MIN	9.8	10.1	9.8	9.5	9.7	8.1
	AVG	12.1	15.7	23.6	26.3	22.9	50.6
	MAX	27.3	52.6	79.1	83.2	62.2	82.3
	STD	5.1	12.3	25.2	26.6	17.2	28.8
Median	MIN	9.7	9.7	9.5	10.4	11.4	9.7
	AVG	24.0	35.8	36.1	20.1	12.0	43.4
	MAX	88.0	94.6	96.4	95.8	12.7	96.1
	STD	27.1	37.9	38.4	25.2	0.4	32.9
High Pass	MIN	10.3	11.5	11.5	11.5	11.5	11.5
	AVG	11.4	11.9	11.9	11.9	11.9	11.9
	MAX	12.6	12.7	12.7	12.7	12.7	12.7
	STD	0.6	0.4	0.4	0.4	0.4	0.4
Gaussian	MIN	9.7	10.9	11.5	11.5	11.5	11.5
	AVG	11.7	11.7	11.9	11.9	11.9	11.9
	MAX	12.7	12.6	12.7	12.7	12.7	12.7
	STD	0.8	0.4	0.4	0.4	0.4	0.4
Laplacian	MIN	10.6	11.4	11.5	11.5	11.5	11.5
	AVG	11.6	11.9	11.9	11.9	11.9	11.9
	MAX	12.6	12.7	12.7	12.7	12.7	12.7
	STD	0.6	0.4	0.4	0.4	0.4	0.4
Laplacian of Gaussian	MIN	10.4	11.5	11.5	11.5	11.5	11.5
	AVG	11.7	11.9	11.9	11.9	11.9	11.9
	MAX	12.7	12.7	12.7	12.7	12.7	12.7
	STD	0.6	0.4	0.4	0.4	0.4	0.4
DANi	MIN	81.1	84.5	86.9	87.6	88.2	89.5
	AVG	82.3	86.7	88.3	89.2	89.5	90.9
	MAX	83.9	87.8	90.2	90.7	90.2	92.5
	STD	1.0	1.0	1.0	1.0	0.6	0.9

Table 5
Accuracy results (in %) and statistics for CHROMA FE with different NF types.

NF type	Statistics	Number of units					
		8	12	16	20	25	50
No NF	MIN	36.2	35.7	38.6	39.8	39.5	40.2
	AVG	38.5	39.9	40.7	41.5	41.2	42.0
	MAX	40.4	41.9	42.8	43.7	42.9	43.7
	STD	1.3	1.8	1.3	1.3	1.2	0.8
Median	MIN	9.7	9.7	9.5	10.4	11.4	9.7
	AVG	24.0	35.8	36.1	20.1	12.0	43.4
	MAX	88.0	94.6	96.4	95.8	12.7	96.1
	STD	27.1	37.9	38.4	25.2	0.4	32.9
High Pass	MIN	32.1	30.3	32.3	31.6	32.9	33.6
	AVG	33.6	34.4	34.4	34.8	34.4	35.0
	MAX	36.9	36.1	36.4	37.8	35.4	37.1
	STD	1.5	1.7	1.3	1.8	0.9	1.1
Gaussian	MIN	29.5	30.6	30.7	31.4	31.6	31.5
	AVG	31.7	32.4	32.7	33.1	33.4	33.6
	MAX	34.2	34.4	37.0	38.4	37.5	38.0
	STD	1.4	1.0	1.8	2.0	1.8	2.0
Laplacian	MIN	32.8	34.0	35.5	36.0	36.9	37.9
	AVG	34.8	36.5	37.2	37.1	38.8	39.9
	MAX	37.1	40.2	39.2	39.4	40.1	41.5
	STD	1.4	1.9	1.2	1.0	1.1	1.1
Laplacian of Gaussian	MIN	33.8	35.7	36.4	37.8	37.0	38.2
	AVG	36.9	38.2	39.0	39.6	39.8	41.5
	MAX	40.0	40.9	42.4	41.5	42.6	44.2
	STD	2.3	1.9	1.8	1.4	1.6	1.8
DANi	MIN	0.0	30.5	32.8	0.0	0.0	34.9
	AVG	27.7	34.1	34.7	32.3	32.7	36.8
	MAX	33.1	35.5	37.1	37.8	37.3	39.0
	STD	9.3	1.5	1.1	10.9	10.9	1.2

Table 6
Accuracy results (in %) and statistics for MFCC FE with different NF types.

NF type	Statistics	Number of units					
		8	12	16	20	25	50
No NF	MIN	10.4	11.1	26.6	34.4	31.5	34.7
	AVG	30.5	33.4	32.1	38.9	33.3	37.1
	MAX	38.0	39.9	38.9	41.5	38.3	39.7
	STD	9.8	8.2	3.0	2.2	1.9	1.5
Median	MIN	9.7	9.7	9.7	32.2	35.7	34.5
	AVG	25.3	30.7	31.0	35.3	38.1	39.5
	MAX	34.6	41.0	39.0	37.7	40.7	42.7
	STD	7.0	10.3	8.9	1.8	1.5	2.3
High Pass	MIN	9.7	14.9	15.8	15.6	9.9	14.6
	AVG	15.6	17.5	17.5	17.6	16.6	16.2
	MAX	17.8	19.4	18.6	19.4	18.8	17.5
	STD	2.7	1.2	0.9	1.1	2.6	1.0
Gaussian	MIN	10.0	14.3	14.0	14.9	12.3	14.3
	AVG	14.4	15.5	15.1	15.6	15.4	15.3
	MAX	16.5	17.7	16.9	17.0	17.2	17.0
	STD	1.7	1.0	0.8	0.7	1.5	0.8
Laplacian	MIN	10.3	9.5	11.1	11.5	11.6	9.7
	AVG	26.2	25.1	27.9	32.5	34.2	26.3
	MAX	34.6	33.9	41.1	41.7	42.5	43.5
	STD	9.9	7.2	11.1	7.7	8.2	12.7
Laplacian of Gaussian	MIN	10.0	33.3	12.6	10.6	35.6	11.5
	AVG	27.6	38.1	34.0	32.4	38.9	18.6
	MAX	38.6	42.7	40.4	41.9	41.7	35.4
	STD	11.5	3.0	7.5	11.1	2.3	10.3
DANi	MIN	7.9	7.9	28.0	37.3	37.7	40.2
	AVG	16.2	23.5	35.8	40.7	42.0	42.8
	MAX	25.0	34.5	43.2	43.6	44.1	45.1
	STD	7.0	9.6	3.8	2.1	2.0	1.5

Table 7
Accuracy results (in %) and statistics for MFCC delta FE with different NF types.

NF type	Statistics	Number of units					
		8	12	16	20	25	50
No NF	MIN	11.6	11.9	12.8	13.1	13.8	14.3
	AVG	13.3	13.9	14.0	14.9	15.1	15.7
	MAX	14.5	14.7	15.3	16.2	16.0	17.7
	STD	1.0	0.9	0.7	1.0	0.7	0.9
Median	MIN	10.0	9.8	11.3	10.9	12.1	14.6
	AVG	11.1	11.6	12.4	12.8	13.5	15.8
	MAX	12.7	18.7	14.4	14.8	14.2	17.4
	STD	0.8	2.4	0.9	1.2	0.8	0.8
High Pass	MIN	10.3	10.3	9.7	10.6	12.2	14.4
	AVG	11.3	11.5	12.2	12.3	13.1	15.6
	MAX	12.9	12.6	14.7	14.6	14.5	16.5
	STD	0.8	0.8	1.4	1.2	0.7	0.7
Gaussian	MIN	10.5	9.6	11.3	11.8	12.4	13.1
	AVG	11.5	11.3	12.7	13.0	13.4	14.2
	MAX	13.2	12.7	15.6	14.0	15.2	15.3
	STD	0.9	0.9	1.3	0.7	0.7	0.7
Laplacian	MIN	9.9	9.7	10.7	10.6	12.3	13.7
	AVG	11.4	12.2	11.7	12.8	13.9	15.2
	MAX	13.3	14.3	12.7	14.0	16.1	16.4
	STD	1.0	1.2	0.6	1.0	1.1	0.8
Laplacian of Gaussian	MIN	9.5	10.5	12.1	10.9	12.2	14.4
	AVG	11.1	11.4	13.2	13.8	14.6	15.5
	MAX	12.1	13.0	14.8	15.2	16.5	17.4
	STD	0.7	0.7	0.8	1.2	1.2	1.0
DANi	MIN	12.4	10.6	12.6	10.3	12.5	12.2
	AVG	14.1	14.8	15.0	14.8	15.4	14.4
	MAX	16.1	16.5	17.6	17.3	17.4	16.2
	STD	1.2	1.8	1.5	1.8	1.5	1.3

References

Abdollahi, M., Giovenazzo, P., & Falk, T. H. (2022). Automated beehive acoustics monitoring: A comprehensive review of the literature and recommendations for future work. *Applied Sciences*, *12*(8).

Al-Turjman, F. (2019). *Artificial intelligence in IoT* (p. 231). Springer, Cham.

Allen, M. D. (1956). The behaviour of honeybees preparing to swarm. *The British Journal of Animal Behaviour*, *4*(1), 14–22.

Amlathe, P. (2018). *Standard machine learning techniques in audio beehive monitoring: Classification of audio samples with logistic regression, K-nearest neighbor, random forest and support vector machine* (Ph.D. thesis), Utah State University.

Badenhorst, J., & de Wet, F. (2019). The usefulness of imperfect speech data for ASR development in low-resource languages. *Information*, *10*(9).

Bae, S. H., Choi, I., & Kim, N. S. (2016). Acoustic scene classification using parallel combination of LSTM and CNN. In *Detection and classification of acoustic scenes and events 2016 workshop* (pp. 11–15).

Bencsik, M., Bencsik, J., Baxter, M., Lucian, A., Romieu, J., & Millet, M. (2011). Identification of the honey bee swarming process by analysing the time course of hive vibrations. *Computers and Electronics in Agriculture*, *76*(1), 44–50.

Bromenshenk, J. J., Henderson, C. B., Seccomb, R. A., Rice, S. D., & Etter, R. T. (2007). Honey bee acoustic recording and analysis system for monitoring hive health. https://www.freepatentsonline.com/y2007/0224914.html, Accessed: 2021-08-05.

Brundage, T. J. (2009). Acoustic sensor for beehive monitoring. https://patents.google.com/patent/US8152590B2/en, Accessed: 2021-10-21.

Cecchi, S., Terenzi, A., Orcioni, S., Riolo, P., Ruschioni, S., & Isidoro, N. (2018). A preliminary study of sounds emitted by honey bees in a beehive. *Journal of the Audio Engineering Society Convention*, *144*(9981).

Cejrowski, T., & Szymański, J. (2021). Buzz-based honeybee colony fingerprint. *Computers and Electronics in Agriculture*, *191*, Article 106489.

Cejrowski, T., Szymański, J., Mora, H., & Gil, D. (2018). Detection of the bee queen presence using sound analysis. In N. T. Nguyen, D. H. Hoang, T.-P. Hong, H. Pham, & B. Trawiński (Eds.), *Asian conference on intelligent information and database systems* (pp. 297–306).

Chao, L., & Shouying, L. (2022). A pest intrusion detection in Chinese beehive culture using deep learning. *Scientific Programming*, *2022*, Article 4642995.

Collison, C. (2018). A closer look: piping, toothing, quacking. *Bee Culture*, 1–8.

Dietlein, D. G. (1985). A method for remote monitoring of activity of honeybee colonies by sound analysis. *Journal of Apicultural Research*, *24*(3), 176–183.

Dougherty, E. R. (2020). *Digital image processing methods*. CRC Press.

Euler, L. (1739). *Tentamen novae theoriae musicae ex certissimis harmoniae principiis dilucide expositae*. Ex typographia Academiae scientiarum.

Fang, Z., Guoliang, Z., & Zhanjiang, S. (2001). Comparison of different implementations of MFCC. *Journal of Computer Science and Technology*, *16*, 582–589.

Favre, D. (2020). Sound of honeybees during new year's eve 2019, dryad, dataset. http://dx.doi.org/10.5061/dryad.5mkkwh748, Accessed: 2021-10-21.

Ferrari, S., Silva, M., Marcella, G., & Daniel, B. (2008). Monitoring of swarming sounds in bee hives for early detection of the swarming period. *Computers and Electronics in Agriculture*, *64*(1), 72–77.

Forsyth, D., & Ponce, J. (2011). *Computer vision: A modern approach* (second ed.). (p. 792). Prentice Hall.

Ganchev, T., Fakotakis, N., & George, K. (2005). Comparative evaluation of various MFCC implementations on the speaker verification task. *Vol. 1*, In *10th international conference on speech and computer* (pp. 191–194).

Giannakopoulos, T., & Pikrakis, A. (2014). *Introduction to audio analysis: A MATLAB® approach*. Academic Press.

Gonzalez, R. C., & Woods, R. E. (2008). *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall.

Goulson, D., Nicholls, E., Botías, C., & Rotheray, E. L. (2015). Bee declines driven by combined stress from parasites, pesticides, and lack of flowers. *Science*, *347*(6229), Article 1255957.

Gradišek, A., Slapničar, G., Šorn, J., Luštrek, M., Gams, M., & Grad, J. (2017). Predicting species identity of bumblebees through analysis of flight buzzing sounds. *Bioacoustics*, *26*(1), 63–76.

Hadjur, H., Ammar, D., & Lefèvre, L. (2022). Toward an intelligent and efficient beehive: A survey of precision beekeeping systems and services. *Computers and Electronics in Agriculture*, *192*, Article 106604.

Hord, L., & Shook, E. (2013). *Determining honey bee behaviors from audio analysis*. National Science Foundation Research Experience for Teachers. Appalachian State University.

Hossan, M. A., Memon, S., & Gregory, M. A. (2010). A novel approach for MFCC feature extraction. In *4th international conference on signal processing and communication systems* (pp. 1–5). IEEE.

Howard, D., Duran, O., Hunter, G., & Stebel, K. (2013). Signal processing the acoustics of honeybees (APIS MELLIFERA) to identify the "queenless" state in hives. *Proceedings of the Institute of Acoustics*, *35*(1), 290–297.

ITU (1988). *Pulse code modulation (PCM) of voice frequencies*. ITU-T Recommendation G.711 of the International Telecommunication Union.

Jensen, J. R. (1986). Introductory digital image processing: a remote sensing perspective. *Technical report*, University of South Carolina, Columbus.

Justusson, B. I. (1981). Median filtering: Statistical properties. In *Two-dimensional digital signal prcessing II: transforms and median filters* (pp. 161–196). Berlin, Heidelberg: Springer Berlin Heidelberg.

Kattel, M., Nepal, A., Shah, A. K., & Deepe, S. (2019). Chroma feature extraction. *Encyclopedia of GIS*, 1–9.

Kerr, H. T., Buchanan, M. E., & Valentine Kenneth, H. (1988). Method and device for identifying different species of honeybees. https://patents.google.com/patent/US4876721A/en, Accessed: 2021-08-05.

Kulyukin, V., Mukherjee, S., & Amlathe, P. (2018). Toward audio beehive monitoring: Deep learning vs. Standard machine learning in classifying beehive audio samples. *Applied Sciences*, *8*(9).

Mezquida, A. D., & Martínez Llorente, J. (2009). Platform for bee-hives monitoring based on sound analysis. a perpetual warehouse for swarm's daily activity. *Spanish Journal of Agricultural Research*, *7*(4), 824–828.

Mukherjee, S. (2018). BeePi_Audio_Classification, Dataset. https://usu.app.box.com/v/BeePiAudioData, Accessed: 2021-10-21.

Neumann, P., & Carreck, N. (2010). Honey bee colony losses. *Journal of Apicultural Research*, *49*(1), 1–6.

Nolasco, I., & Benetos, E. (2018). To bee or not to bee: An annotated dataset for beehive sound recognition, Dataset, Zenodo. https://zenodo.org/record/1321278#.YQvVhRQzZpQ, Accessed: 2021-10-21.

Nolasco, I., Terenzi, A., Cecchi, S., Orcioni, S., Bear, H. L., & Benetos, E. (2019). Audio-based identification of beehive states. In *2019-2019 IEEE international conference on acoustics, speech and signal processing* (pp. 8256–8260).

Patrício-Roberto, G. B., & Campos, M. J. (2014). Aspects of landscape and pollinators – what is important to bee conservation? *Diversity*, *6*(1), 158–175.

Pratt, W. K., & Wiley, J. (1978). Digital image processing. In *A wiley-interscience publication*. Citeseer.

Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S.-Y., & Sainath, T. (2019). Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, *13*(2), 206–219.

Python Software Foundation (2021). Python 3.10.0. wave module documentation. https://docs.python.org/3/library/wave.html, Accessed: 2021-10-22.

Qandour, A., Ahmad, I., Habibi, D., & Leppard, M. (2014). Remote beehive monitoring using acoustic signals. *Acoustics Australia*, *42*(3).

Ramsey, M., Bencsik, M., & Newton, M. I. (2017). Long-term trends in the honeybee 'whooping signal' revealed by automated detection. *PLoS One*, *12*(2), Article e0171162.

Robles-Guerrero, A., Saucedo-Anaya, T., González-Ramérez, E., & Galván-Tejada, C. E. (2017). Frequency analysis of honey bee buzz for automatic recognition of health status: A preliminary study. *Research in Computing Science*, *142*, 89–98.

Robles-Guerrero, A., Saucedo-Anaya, T., González-Ramírez, E., & De la Rosa-Vargas, J. I. (2019). Analysis of a multiclass classification problem by lasso logistic regression and singular value decomposition to identify sound patterns in queenless bee colonies. *Computers and Electronics in Agriculture*, *159*, 69–74.

Ruvinga, S., Hunter, G. J., Duran, O., & Nebel, J.-C. (2021). Use of LSTM networks to identify "queenlessness" in honeybee hives from audio signals. In *17th international conference on intelligent environments* (pp. 1–4).

Scheffer, C. (2004). *Practical machinery vibration analysis and predictive maintenance*. San Diego, CA: Elsevier.

Sharif, M. Z., Wario, F., Di, N., Xue, R., & Liu, F. (2020). Soundscape indices: New features for classifying beehive audio samples. *Sociobiology*, *67*(4), 566–571.

Shostak, S., & Prodeus, A. M. (2019). Classification of the bee colony condition using spectral features. In *2019 IEEE international scientific-practical conference problems of infocommunications, science and technology (PIC S&T)* (pp. 737–740).

Smith, T. C. (2021). Osbeehives. https://www.osbeehives.com/, Accessed: 2021-10-21.

Soares, B. S., Luz, J. S., de Macêdo, V. F., e Silva, R. R. V., de Araújo, F. H. D., & Magalhães, D. M. V. (2022). MFCC-based descriptor for bee queen presence detection. *Expert Systems with Applications*, *201*, Article 117104.

Stockwell, R. G., Mansinha, L., & Lowe, R. (1996). Localization of the complex spectrum: the S transform. *IEEE Transactions on Signal Processing*, *44*(4), 998–1001.

Terenzi, A., Cecchi, S., Orcioni, S., & Piazza, F. (2019). Features extraction applied to the analysis of the sounds emitted by honey bees in a beehive. In *11th international symposium on image and signal processing and analysis* (pp. 3–8).

Terenzi, A., Cecchi, S., & Spinsante, S. (2020). On the importance of the sound emitted by honey bee hives. *Veterinary Sciences*, *7*(4), 168–174.

Tukey, J. W. (1977). *Vol. 2*, *Exploratory data analysis*. Reading, MA.

Wang, X. (2007). Laplacian operator-based edge detectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(5), 886–890.

Wang, P., Fan, E., & Wang, P. (2021). Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, *141*, 61–67.

Winograd, S. (1978). On computing the discrete Fourier transform. *Mathematics of Computation*, *32*(141), 175–199.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1–3), 37–52.

Yuan, X., Li, L., & Wang, Y. (2019). Nonlinear dynamic soft sensor modeling with supervised long short-term memory network. *IEEE Transactions on Industrial Informatics*, *16*(5), 3168–3176.

Zacepins, A., Brusbardis, V., Meitalovs, J., & Stalidzans, E. (2015). Challenges in the development of precision beekeeping. *Biosystems Engineering*, *130*, 60–71.

Zacepins, A., & Karasha, T. (2013). Application of temperature measurements for the bee colony monitoring: a review. In *12th international scientific conference engineering for rural development* (pp. 126–131).

Zahid Sharif, M., Jiang, X., & Puswal, S. (2020). Pests, parasitoids, and predators: Can they degrade the sociality of a honeybee colony, and be assessed via acoustically monitored systems? *Journal of Entomology and Zoology Studies*, *8*, 1248–1260.

Zgank, A. (2020). Bee swarm activity acoustic classification for an IoT-based farm service. *Sensors*, *20*(1), 21–34.

Zgank, A. (2021). IoT-based bee swarm activity acoustic classification using deep neural networks. *Sensors*, *21*(3), 676–689.