Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Forest Sound Event Detection with Convolutional Recurrent Neural Network-Long Short-Term Memory

Muhammad Naqiuddin Zaini[1], Marina Yusoff[1, 2*], Muhammad Amirul Sadikin[3]

[1] College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Shah Alam, 40450, Selangor, Malaysia
[2] Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Universiti Teknologi MARA, Shah Alam, 40450, Selangor, Malaysia
[3] Shab Electronics, Pulai, 81110 Johor Bahru, Johor, Malaysia

ARTICLE INFO

Article history:
Received 13 May 2023
Received in revised form 15 August 2023
Accepted 22 August 2023
Available online 13 September 2023

Keywords:
Convolution Recurrent Neural Network; feature extraction; forest; Long Short-Term Memory; sound event detection

ABSTRACT

Sound event detection tackles an audio environment's complex sound analysis and recognition problem. The process involves localizing and classifying sounds mainly to estimate the start point and end points of the separate sounds and describe each sound. Sound event detection capability relies on the type of sound. Although detecting sequences of distinct temporal sounds is straightforward, the situation becomes complex when the sound is multiple overlapping of much single audio. This situation usually occurs in the forest environment. Therefore, this aim of the paper is to propose a Convolution Recurrent Neural Network-Long Short-Term Memory algorithm to detect an audio signature of intruders in the forest environment. The audio is extracted in the Mel-frequency cepstrum coefficient and fed into the algorithm as an input. Six sound categories are chainsaw, machete, car, hatchet, ambiance, and bike. They were tested using several epochs, batch size, and filter of the layer in the model. The proposed model can achieve an accuracy of 98.52 percent in detecting the audio signature with a suitable parameter selection. In the future, additional types of audio signatures of intruders and further aspects of evaluation can be added to make the algorithm better at detecting intruders in the forest environment.

1. Introduction

Sound Event Detection (SED) is a growing research area. It tackles the complex problem of sound analysis and recognition within a general audio environment [1]. In recent years, there has been a surge of interest in developing SED systems for environmental sound analysis. SED systems aim to automatically identify and classify various sound events from acoustic recordings, such as bird calls, animal vocalizations, and human activities. Forests are especially important for sound event detection because they house various sound sources and provide critical habitats for many species. On the other hand, the noisy and complex acoustic environment of forests poses significant challenges for SED systems.

* *Corresponding author.*
E-mail address: marina998@uitm.edu.my

https://doi.org/10.37934/araset.32.2.242254

Several ideas on security surveillance [2], wildlife monitoring [3], and audio indexing and classification [4]. Locating and classifying sounds audio, determining distinct sound event occurrences, and creating a description for each circumstance are processes in SED [1]. SED can detect many sounds from audio samples from the general classification problems that allocate sounds to each class. However, the difficulty of SED changes according to the task assigned. In a more complex situation, sound event detection in several overlapping sounds in single audio typically occurs daily. An audio of a street could contain multiple sound sources, such as footsteps, people talking, and car passing. All those sounds can be identified as a mixture of events or polyphonic SED [5]. In a real-life situation, audios contain variety of multiple sounds. There have been many attempts established for SED classification in past years, including CNN [6,7], Support Vector Machine [8], Random Forest (RF) [9], Masked Conditional Neural Network [6], Convolution Recurrent Neural Network (CRNN) [1,3,10]. CRNN has garnered more attention in recent SED studies.

Research has used real data from the forest for intruder detection, vehicle movement detection with MFCC and K-Nearest Neighbour to assist in reducing illegal entry in the forest, tree cutting, chainsaw, hatchet, ambiance, and vehicle sound [9,11]. RF and Linear Predictive Coding (LPC) work in the surveillance intrusion system performed up to 86 percent accuracy in detecting intrusion in the forest [12]. This work is based on vehicle movement detection with MFCC and K-Nearest Neighbour to reduce illegal entry into the forest. However, the accuracy of sound detection should be further improved. Existing SED systems for forest environments rely on traditional machine learning algorithms that require hand-crafted features and are limited in capturing the acoustic environment's complexity and dynamic nature. To address these limitations, we propose a novel approach for detecting forest sound events based on convolutional recurrent neural network-long short-term memory (CRNN-LSTM). The CRNN-LSTM deep learning model learns local and temporal features from acoustic data by combining the strengths of convolutional neural networks (CNNs) and recurrent neural networks (RNNs). This study aims to improve the performance of SED systems in forest environments and develop a more accurate and efficient method for sound event detection.

As a result, this paper aims to investigate CRNN-LSTM using real data from the tropical forest environment. SED addresses the complex problem of sound analysis and recognition in an audio environment. The process entails localizing and classifying sounds, primarily to estimate the start and end points of individual sounds and to describe each sound. SED capability is determined by the type of sound. Consequently, the research work makes three contributions:

i. Creating a novel CRNN-LSTM model for detecting forest sound events: We present a deep learning model that can extract local and temporal features from acoustic data collected in forests. Our model can learn the intricate relationships between sound events and their temporal variations.
ii. The proposed CRNN-LSTM model is evaluated on a new dataset: We collected a unique dataset of forest sounds and evaluated our model's performance on it to validate its effectiveness. On this dataset, our evaluation results show that our proposed model outperforms state-of-the-art SED systems.
iii. Analysis of learned features and insights: We thoroughly examine the available features and insights obtained from our proposed CRNN-LSTM model. Our research sheds light on the distinguishing characteristics that contribute to accurately detecting various sound events in forest environments.

The remainder of this paper is structured as follows. Section 2 focuses on the research background information. The third section goes over the materials and methods. Sections 4 and 5 contain the results and discussion. Section 6 contains the conclusion.

2. Research Background

CNN is one of the many deep-learning algorithms that can take a 2D input and learn and differentiate the input from others [13]. It is due to its ability to extract spatial features such as edges, distribution of color in the image, or data that contains spatial properties, which makes it a great option in data classification [14]. RNN is another variation of a neural network that is a bit different from others. A neural network will take a fixed input vector size that limits its use when dealing with an input series without a pre-determined size Convolutional recurrent neural networks for music classification [15]. However, RNN is designed to take a series of entries without a pre-determined size. Although a neural network can be configured to call more than once, the input series means that one part of the input influences the other. Otherwise, it will just be many inputs, simply. RNNs can remember what past outputs have learned and affects current entry according to the past.

CRNN is a combination of CNN and RNN. CRNN starts with CNN, followed by RNN. In CRNN, the convolution layer acts as feature extraction while the recurrent layer integrates the output from the convolution layer, thus providing context information [5]. The last layer of CRNN, in which all layers are fully connected, produces the probability for each class available. CRNN also can eliminate the limitation that occurs in both CNN and RNN. For instance, CRNN tends to use less memory than CNN since the RNN layer in CRNN uses the global structure for summarization rather than the local structure in CNN [6]. CRNN algorithm can also tag multiple classes without dropping the algorithm's accuracy. Therefore, CRNN can be a viable option in SED as it can utilize the advantages of both CNN and RNN while eliminating some of the limitations that both CNN and RNN have. CRNN uses the CNN local feature extraction capability, and the RNN temporal summarization would lead to an efficient and effective model than standard CNN or RNN on its own [16]. However, CRNN has its limitation and problem which these are due to its architecture; the hidden states in CRNN need to be calculated one by one [17].

CRNN model can be used as a multi-scale squeeze-excitation using a pyramid model to help differentiate the sound events with different durations and recalibrate the channel-wise spectrogram [18]. The model will be influential towards data that needs to be more labeled and labeled. Therefore, by allowing SED to be implemented, these kinds of data can still achieve the desired result. This makes CRNN a versatile model that can be used for different data types. n a conventional CRNN, a high echo value sound is often underestimated. Thus, this makes it hard to be detected by the model. Figure 1 illustrates the difference between conventional CRNN and CRNN-LSTM, where the first line is the conventional CRNN, and the second line is CRNN-LSTM [19]. In Figure 1, the CRNN-LSTM model has a better high echo detection than conventional CRNN. The underestimation of the high echoes produces a trend that makes it disappear from the source when running the model. Because the gates in CRNN are created independently on input, hidden state, and sum fusion, this creates a situation in which the input and the hidden state do not interact to identify and preserve important information.

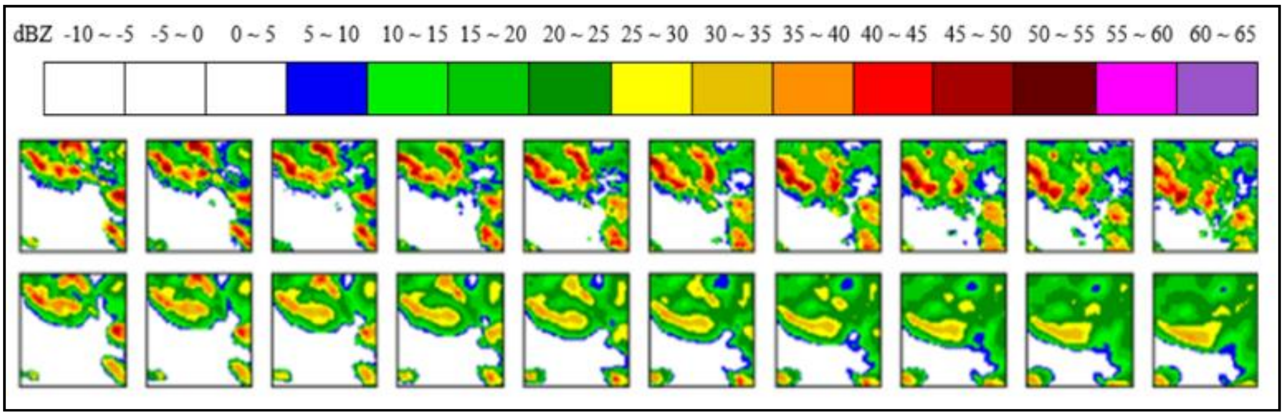


Fig. 1. The instance of radar echo map location

In the architecture of CRNN, it has difficulty capturing long-term dependencies. TCN can overcome this issue by utilizing dilated convolutional [20]. Compared to conventional CNN, TCN allows it to process the input in a parallel sequence rather than wait for the output from the previous time step in conventional CNN. This will help reduce computation time when using TCN in the architecture, as it can simultaneously process the input and output from the previous step.

3. Material and Methods

This section presents the materials and methods used in the evaluation. Figure 2 shows the three main phases, namely feature extraction, model construction, and evaluation settings. The former starts with the preparation of datasets. The dataset was based on the collection of 136 audio samples in tropical forests in Malaysia, as shown in Table 1. Some of the datasets were elaborated in [12]. This article focuses on six sound categories: chainsaw, machete, car, hatchet, ambiance, and bike. Next, the feature extraction phase and model construction are explained in A and B, respectively. Finally, evaluation settings are presented. We use the implementation of CRNN with LSTM together with the Tensor framework.

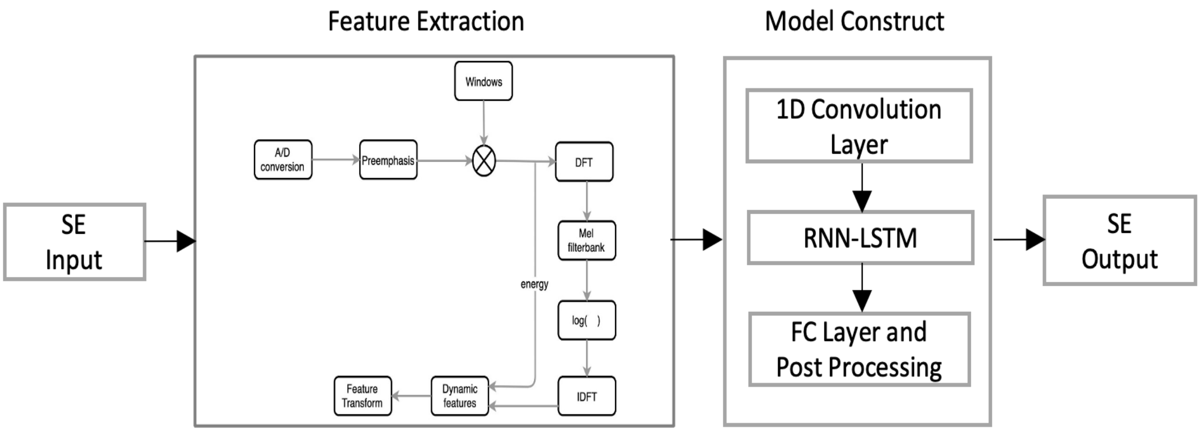


Fig. 2. Sound event detection architecture for CRNN-LSTM

Table 1
Real audio samples in tropical forests in Malaysia

Sound class	Train	Test	Total
Chainsaw	35	9	44
Machete	13	3	16
Car	6	2	8
Hatchet	35	9	44
Ambiance	10	2	12
Bike	10	2	12

3.1 Features Extraction

MFCC works by trying to copy the human ears in detecting sound. By making the audio sound more compatible with human ears' capability, we can eliminate and its variants due to the different hearing sensitivity between human ears and the audio recording equipment. We use MFCC as recommended [11]. It is calculated by defining the Short-Time Fourier Transform from the curves of individuals. A brief of the steps for MFCC is presented in the following explanations:

i. A/D conversion, Audio clips will be sampled and digitized, converting from an analog signal into discrete space.
ii. Pre-emphasis involves boosting the high frequency of the audio.
iii. Windowing, the audio waveform is sliced into sliding frames. However, the edge of the frames cannot simply be sliced because it will create noise in high frequency due to sudden-fallen amplitude. Therefore, it is better to slice them when the amplitude gradually drops [15].
iv. Discrete Fourier Transformation function could extract the frequency domain of the information.
v. Mel filter bank, the audio will be scaled based on mel frequency, and bark frequency since the hearing perception of humans and the measuring equipment might differ.
vi. The log output of the mel filterbank is logged out to reduce acoustic variants that are not important to audio recognition.
vii. Discrete cosine transform is used to obtain the MFCC coefficient c(n).
viii. Dynamic features, MFCC has an overall of 39 features.
ix. Cepstral mean and variance normalization, the features will be normalized by finding its mean and dividing it by its variance. It will allow the values to be adjusted to countermeasure the variance in the data.

3.2 Modeling

In this phase, the model uses CRNN architecture to detect intruders' sounds in a forest environment. Three steps are involved as demonstrated in Figure 3. The first step involves the design of 1D CNN components, the second step on RNN-LSTM architecture design and the final step is the FC layer design and post-processing.

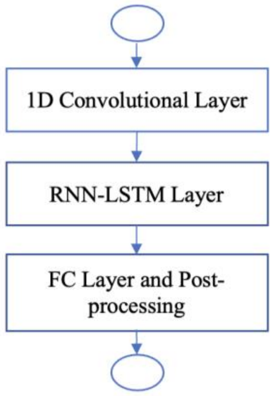


Fig. 3. Steps for CRNN-LSTM Model Construction

In 1D CNN, the loss function used in this study was sparse categorical entropy and adaptive momentum (ADAM). The learning rate of 0.0001 was used in the architecture. The CNN used in this study contains five components. They are the convolutional layer, BN, activation layer, and drop out. The 1D CNN used in this study had three layers. Convolutional was performed on the input signal in the kernels or filters layer. The number and size of kernels play a significant role in capturing relevant features from the signal. BN helps improve the reliability of the neural network. It works by normalizing the output from the previous layer by subtracting the batch mean and by dividing it by the batch standard deviation [18]. For the activation layer, we used Rectified Linear Unit [5]; meanwhile, in the pooling layer, the detected features were sampled down into the size set. The max-pooling was applied at each output to extract representative values.

For RNN-LSTM, the features collected from the Convolutional layer were fed to the RNN-LSTM layer. This research used two RNN layers, each containing 64 LSTM units using unidirectional backward RNN-LSTM. In both RNN layers, we used hyperbolic tangent (tanh) and a dropout rate equal to 0.3 for all layers. The features received from RNN-LSTM were added to the FC layer. BN and ReLU were applied similarly in the 1D CNN layer. A sigmoid unit was receiving the forwarded updated features in which the output represented the possibility of existing of the chosen sound event. Therefore, the values of possibility for every frame in the mel-spectrogram were calculated. The entire probability of the audio sequence audio was obtained using the sliding assemble method [18].

3.3 Evaluation

The model was tested and evaluated with different parameters, such as different epochs, batch sizes, filter sizes, and LSTM layers, to achieve the best performance. Five experiments were conducted to determine the best setting for the model to obtain the best performance. Table 2 shows the experimental setup for five experiments.

Table 2
Parameter settings

Type of experiment	Number of epochs	Batch sizes	1st CNN layer	2nd CNN layer	LSTM layer
Experiment 1	50, 100, 150	32	32	32	64
Experiment 2	100	16, 32, 64	32	32	64
Experiment 3	100	32	16, 32, 64	32	64
Experiment 4	100	32	32	16, 32, 64	64
Experiment 5	100	32	32	32	32, 64, 128

Accuracy and loss were taken into consideration as training phase performance indicators in order to select the best model for supervised learning. The model will perform better if its accuracy is raised. In order to calculate the accuracy, use Eq. (1).

Accuracy = Number of correct prediction / total number of prediction (1)

Accuracy and loss were considered as training phase performance indicators to select the best model for supervised learning. The model will perform better if its accuracy improves. In order to calculate the accuracy, use Eq.(1).

Accuracy = Number of correct predictions / total number of predictions (1)

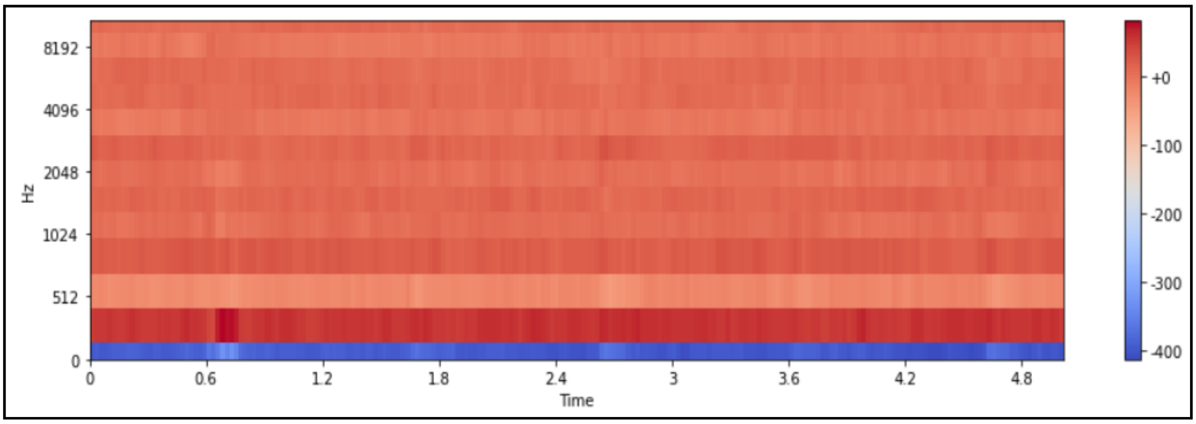
A loss is a penalty for making an incorrect prediction. In other words, the loss is a number that indicates how inaccurate the model's prediction was on a single example. The loss is zero if the model's prediction is perfect; otherwise, the loss is more significant. Assuming a sequence prediction task with LSTM and a sequence of accurate labels (y_1, y_2, ..., y_T) and corresponding predicted probabilities (p_1, p_2, ..., p_T) from the LSTM, the cross-entropy loss can be calculated as in Eq. (2).

$$L = -1/T * \Sigma(y_t * \log(p_t)) \quad (2)$$

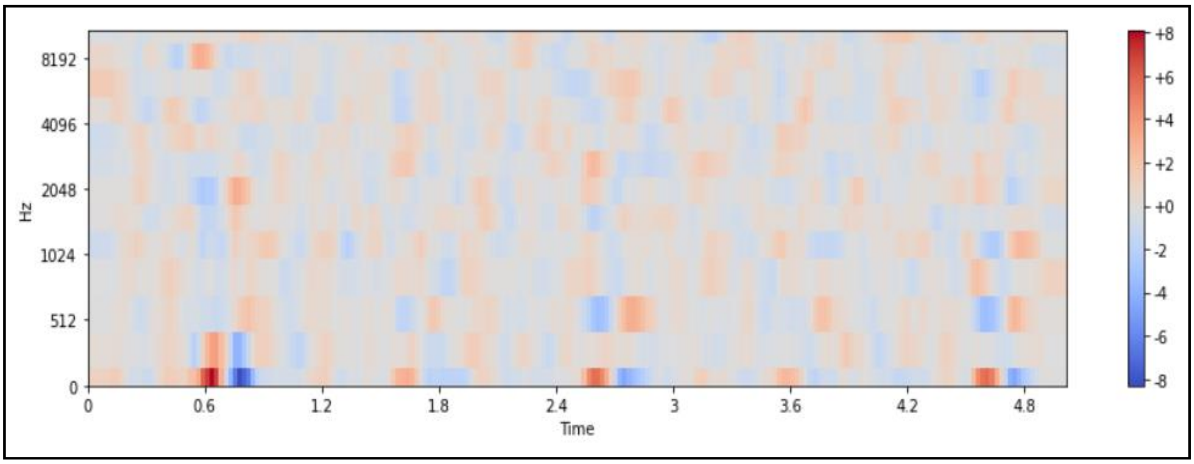
where:
T is the length of the sequence
y_t is the true label at time step t
p_t is the predicted probability for the true label at time step t
log represents the natural logarithm

4. Results for CRNN-LSTM

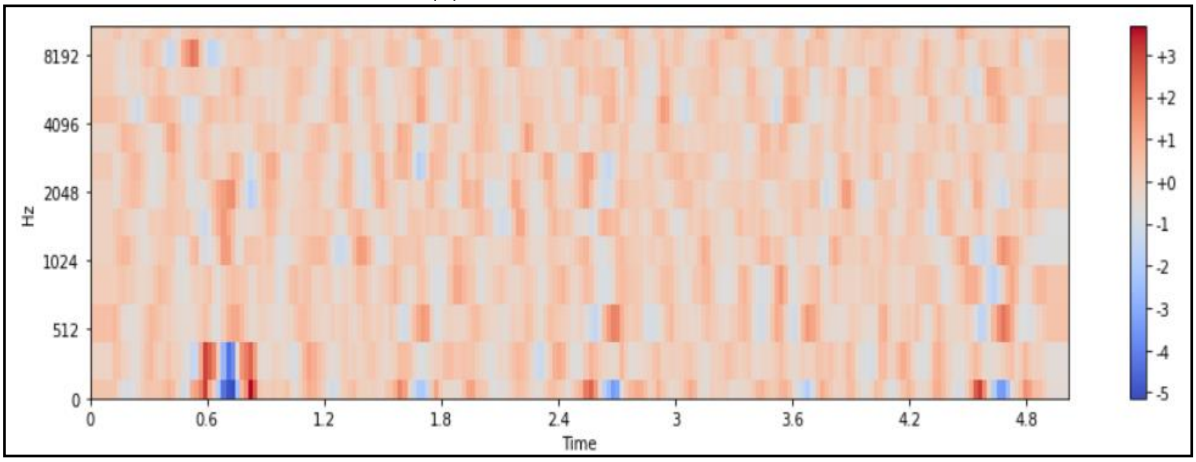
Results and findings obtained from the audio recognition of the audio data that represent sounds of intruders in the forest environment is explain. The experiment conducted using CRNN-LSTM with MFCC features extraction method. Figure 4 (a)-(c) shows the feature extracted from a 5-second sample of a chainsaw activity. Figure 4(a) is the MFCC extracted sample. The first derivative of MFCC and the second derivative can be seen in Figure 4 (b) and (c). In both results, the MFCC result was further derived to get the real value coefficient which was then saved into a JSON file for storage.



(a) The result of MFCC gain visualizes



(b) The first derivative of MFCC



(c) The second derivative of MFCC

Fig. 4. MFCC Extractions, first and second derivatives

4.1 Results for Experiment 1 based On the Number of Epochs

During training, the number of epochs determines the number of times the model will iterate over the entire dataset. Selecting the appropriate number of epochs to balance underfitting (the model hasn't learned enough) and overfitting (the model memorizes the training data) is crucial. Too few epochs may result in underfitting, in which the model fails to learn the underlying patterns and generalizes poorly to new data. An excessive number of epochs can result in overfitting, in which the model becomes overly specialized in the training data and performs poorly on new, unseen data. The optimal number of epochs depends on the task's difficulty, the training set's size, and the neural network's architecture. It is frequently determined through experimentation and monitoring the model's performance on a validation set.

The model was trained with 50, 100, and 150 epochs. The results are depicted in Table 3. Overall, an optimized number of epochs could have resulted in better accuracy. The accuracy of the test increased when the number of epochs used for training increased. The lowest accuracy was obtained for 50 epochs with 92.27 percent. Meanwhile, the highest accuracy obtained from the experiment was 98.16 percent when it was trained with 100 epochs. Furthermore, an optimized number of epochs can result in a better loss. The highest value for the experiment occurs for 50 epochs, which is 0.2446. Meanwhile, the lowest loss value for the experiment is 0.0492 for 150 epochs. Hence, it can be said that the loss value has shown a decrease with an optimized number of epochs.

Table 3
Result for Different Numbers of Epochs

Number of epochs	Accuracy (%)	Loss
50	92.27	0.2446
100*	98.16	0.0492
150	97.94	0.0730

4.2 Results for Experiment 2 based on the number of batches

Batch size is the number of training examples propagated throughout the network before updating the model's parameters. Choosing an appropriate batch size affects both the efficiency and generalization of training. The model can process multiple examples in parallel, so larger batch sizes can facilitate faster training. This can be advantageous when working with large datasets or computationally intensive models. Smaller batch sizes permit more frequent parameter updates, which may result in better convergence and enhanced generalization. However, due to the increased frequency of parameter updates, very small batch sizes may result in noisy gradients and slower training. The optimal batch size is determined by available computational resources, the size of the dataset, and the nature of the problem at hand.

In this experiment, an optimized number of batch sizes can result in the best accuracy. When the number of batch sizes used to train was reduced, the test's accuracy increased. Table 4 shows the results. In this experiment, the lowest accuracy obtained was 94.12 percent for 64 batch sizes. Meanwhile, when the experiment was trained with 32 batch sizes, the highest accuracy gain was 98.16 percent. According to the results of this experiment, the model's accuracy increased with the most optimized batch size; a higher or lower number of batch sizes did not simply increase the model's accuracy. Furthermore, the experiment's losses reflect the same as the accuracy, with the optimized number of batch sizes producing the lowest loss. The experiment's highest loss was 0.2171 when trained with 64 batch sizes, while the experiment's lowest loss was 0.0492 when trained with 32 batch sizes. The finding demonstrates loss value decrease with the most optimized batch size and increasing or decreasing the number of batch sizes will not simply increase the model's accuracy.

Table 4
Result for different numbers of batches

Batch size	Accuracy (%)	Loss
16	96.32	0.0914
32*	98.16	0.0492
64	94.12	0.2171

4.3 Results for Experiment 3 Based on The Filter Size at The First Convolutional Layer

This section demonstrates the outcomes of experiments involving varying numbers of filters in the initial convolutional layer. CNNs use kernels, or filters, to convolutionally process input data. The depth or dimensionality of convolutional layer output feature maps depends on the number of filters. More filters capture more complex data patterns and features. Too few filters may limit detail capture, resulting in underfitting, and too many filters can complicate and slow the model. If the model memorizes training data, it may overfit. The number of filters depends on the task complexity, input data size and type, and neural network depth. This experiment's accuracy and loss are shown in Table 5. In this experiment, the first convolutional layer's 16 filters achieved the lowest accuracy of 94.12 percent. The experiment yielded the highest accuracy of 98.16 percent when it was trained with 32 filters. Additionally, it was for the initial convolutional layer. This experiment demonstrated that increasing the filters in the first convolutional layer improved accuracy. Moreover, the experiment's losses were reduced with an optimized number of filters. When trained with 16 filters, the highest loss for the experiment was 0.2054 at the first layer. While training with 0.0492 loss at the 32 filters in the first layer resulted in the lowest loss for the experiment, the lowest loss was achieved with 0.0492 loss at the 32 filters in the second layer. Based on this experiment, the loss of the system would decrease as the number of filters in the first layer is optimized.

Table 5
Results for Different the Number of Filters in the First Convolutional Layer

Number of filters	Accuracy (%)	Loss
16	94.12	0.2054
32*	98.16	0.0492
64	96.32	0.1169

4.4 Results of Experiment 4 Based on The Number of Filters in The Second Convolutional Layer

This section presents the explanation of the results from the experiment of different numbers of filters for the second convolutional layer. Table 6 tabulates the result of different numbers of filters for second convolutional layer on the testing accuracy and loss of the model. In this experiment, the lowest accuracy gained was 32 filters for the second convolutional layer, with 93.01 percent. Meanwhile, the highest accuracy gain from the experiment was 98.16 percent when it was trained with 64 filters for the second convolutional layer. From this experiment, the accuracy of the model increased when the number of filters was in the second convolutional layer. Furthermore, the losses of the experiment decreased when the number of filters increased. The highest loss for the experiment was when trained with 32 filters r, which was 0.2429. While the lowest loss for the experiment was when trained with 64 filters from the second convolutional layer at about 0.0492 loss. Based on this experiment, the loss of the model decreased with an optimized number of filters at the second layer.

Table 6
Result of a different number of filters in the second convolutional layer

Number of filters	Accuracy (%)	Loss
32	93.01	0.2429
64*	98.16	0.0492
128	97.06	0.0908

4.5 Results for Experiment 5 Based on The Number of Filters Used in The LSTM Layer

This section reports the results from the experiment of using different numbers of filters in the LSTM layer. Table 7 shows the result of a different number of filters in the LSTM layer on the testing accuracy and loss of the model. In this experiment, the lowest accuracy gained was 32 filters in the LSTM layer, with 93.01 percent. Meanwhile, the highest accuracy gain from the experiment was 98.16 percent when it was trained with 64 filters in the LSTM layer. From this experiment, the accuracy of the system increased with an optimized number of filters in the LSTM layer. Furthermore, the losses of the experiment decreased with an optimized number of filters in the LSTM layer. The highest loss for the experiment was when trained with 32 filters in the LSTM layer, which was 0.2429. While the lowest loss for the experiment was when trained with 64 filters in the LSTM layer with 0.

0.0492 in a loss. From this experiment, it could be observed that the loss of the model decreased with an optimized number of filters in the LSTM layer.

Table 7
Result of based on the number of filters used in the LSTM layer

Number of filters	Accuracy (%)	Loss
32	93.01	0.2429
64*	98.16	0.0492
128	97.06	0.0908

4.6 Results for CNN, CNN-RF, and RF

This section presents the experiment results for CNN, Convolution Neural-Network-Random Forest (CNN-RF), and RF. Table 8 displays each model's testing accuracy and loss. CNN outperforms CNN-RF and FR in terms of accuracy, scoring 95.52 with a loss of 0.0266. However, it performs less well than the proposed CRNN_LSTM.

Table 8
Result of CNN, CNN-RF and RF

Number of filters	Accuracy (%)	Loss
CNN	95.52	0.0266
CNN-RF Hybrid	79.69	-
RF	72.09	-

5. Discussion

From the experiments, it is interesting to note that the best number of epochs used is 100, which has an accuracy of 98.16 percent and a loss of 0.0492. The same result is from 32 batch sizes, 32 numbers of filters in the first convolutional layer, 64 numbers for the second convolutional layer, and 64 numbers of filters in the LSTM layer. Compared to the previous result with RF, CNN and CNN-RF using similar datasets, CRNN-LSTM has significantly improved performance. It is interesting to note that all the experimental results of CRNN-LSTM outperformed RF, CNN, and CNN-RF. We also compare with CNN using different convolutional and fully connected layers. The result is highlighted in Table 9. From the CNN model configuration, the highest accuracy is 98.00 percent which is at par with CRNN-LSTM, but the loss value reported is a little bit higher. Regarding this, more evaluation metrics, such as false positives and sensitivity analysis, should be considered.

Table 9
Result of CRNN

Convolutional Layers		Accuracy (%)	Loss
1st	2nd		
32	32	95.52	0.0266
64*	32	98.00	0.0602
128	64	95.49	0.1463

6. Conclusions

This study offers a workable solution for the SED of real-world sound datasets from a tropical forest setting. Additionally, we discovered from the relevant literature that CNNs, RNNs, and CRNNs can deliver promising detection performance for reliable data. CNNs, RNNs, and CRNNs can deliver encouraging detection performance for sound data. As a result, a CRNN-LSTM model was created and then evaluated using a variety of parameters, such as epoch, batch sizes, and the number of filter layer selections. The overall results are much more favorable than those obtained earlier. The lowest loss value for ambiance sound identification is 0.0492 percent, and the highest accuracy level is 98.16 percent. More evaluation and testing on this algorithm are suggested, mainly on utilizing the various types of intruder sound events. Testing, which can be done on real-time data, and implementation in the real world could be part of future work. In addition, another hybrid algorithm, such as embedded with computational optimization methods, would enhance the model capability, and use another performance measure like a confusion matrix for a detailed performance check.

Acknowledgement
The authors would like to thank the provider of this research grant, Malaysian Technical Standards Forum Bhd (MTSFB), Research Management Center, Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), College Computing, Informatics and Media, Universiti Teknologi MARA, Shah Alam, Malaysia for providing essential supports and knowledge to the authors.

References
[1] Adavanne, Sharath, Pasi Pertilä, and Tuomas Virtanen. "Sound event detection using spatial features and convolutional recurrent neural network." In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 771-775. IEEE, 2017. https://doi.org/10.1109/ICASSP.2017.7952260
[2] Ahmad, Sheikh Fahad, and Deepak Kumar Singh. "Automatic detection of tree cutting in forests using acoustic properties." *Journal of King Saud University-Computer and Information Sciences* 34, no. 3 (2022): 757-763. https://doi.org/10.1016/j.jksuci.2019.01.016
[3] Alshalan, Raghad, and Hend Al-Khalifa. "A deep learning approach for automatic hate speech detection in the saudi twittersphere." *Applied Sciences* 10, no. 23 (2020): 8614. https://doi.org/10.3390/app10238614
[4] Amiriparian, Shahin, Maurice Gerczuk, Sandra Ottl, Lukas Stappen, Alice Baird, Lukas Koebe, and Björn Schuller. "Towards cross-modal pre-training and learning tempo-spatial characteristics for audio recognition with convolutional and recurrent neural networks." *EURASIP Journal on Audio, Speech, and Music Processing* 2020 (2020): 1-11. https://doi.org/10.1186/s13636-020-00186-0
[5] Cakır, Emre, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. "Convolutional recurrent neural networks for polyphonic sound event detection." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25, no. 6 (2017): 1291-1303. https://doi.org/10.1109/TASLP.2017.2690575
[6] Choi, Keunwoo, György Fazekas, Mark Sandler, and Kyunghyun Cho. "Convolutional recurrent neural networks for music classification." In *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pp. 2392-2396. IEEE, 2017. https://doi.org/10.1109/ICASSP.2017.7952585
[7] Ciaburro, Giuseppe, and Gino Iannace. "Improving smart cities safety using sound events detection based on deep neural network algorithms." In *Informatics*, vol. 7, no. 3, p. 23. MDPI, 2020. https://doi.org/10.3390/informatics7030023
[8] Guirguis, Karim, Christoph Schorn, Andre Guntoro, Sherif Abdulatif, and Bin Yang. "SELD-TCN: Sound event localization & detection via temporal convolutional networks." In *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 16-20. IEEE, 2021. https://doi.org/10.23919/Eusipco47968.2020.9287716
[9] Huang, Lin, Chitralekha Gupta, and Haizhou Li. "Spectral features and pitch histogram for automatic singing quality evaluation with crnn." In *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 492-499. IEEE, 2020.
[10] Kong, Qiuqiang, Yong Xu, Wenwu Wang, and Mark D. Plumbley. "Sound event detection of weakly labelled data with cnn-transformer and automatic threshold optimization." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020): 2450-2460. https://doi.org/10.1109/TASLP.2020.3014737
[11] Li, Jiaojiao, Ruxing Cui, Bo Li, Rui Song, Yunsong Li, and Qian Du. "Hyperspectral image super-resolution with 1D–2D attentional convolutional neural network." *Remote Sensing* 11, no. 23 (2019): 2859. https://doi.org/10.3390/rs11232859
[12] Medhat, Fady, David Chesmore, and John Robinson. "Masked conditional neural networks for sound classification." *Applied Soft Computing* 90 (2020): 106073. https://doi.org/10.1016/j.asoc.2020.106073

[13] Shen, Fanlin, Siyi Cheng, Zhu Li, Keqiang Yue, Wenjun Li, and Lili Dai. "Detection of snore from OSAHS patients based on deep learning." *Journal of Healthcare Engineering* 2020 (2020). https://doi.org/10.1155/2020/8864863

[14] Saha, Sumit. "A comprehensive guide to convolutional neural networks—the ELI5 way." *Towards data science* 15 (2018): 15.

[15] Tay, Yi, Luu Anh Tuan, and Siu Cheung Hui. "Cross temporal recurrent networks for ranking question answer pairs." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1. 2018. https://doi.org/10.1609/aaai.v32i1.11973

[16] Ykhlef, Hadjer, Farid Ykhlef, and Sylia Chiboub. "Experimental design and analysis of sound event detection systems: case studies." In *2019 6th International Conference on Image and Signal Processing and their Applications (ISPA)*, pp. 1-6. IEEE, 2019. https://doi.org/10.1109/ISPA48434.2019.8966798

[17] Yusoff, Marina, and Amirul Sadikin Md. Afendi. "Acoustic Surveillance Intrusion Detection with Linear Predictive Coding and Random Forest." In *Soft Computing in Data Science: 4th International Conference, SCDS 2018, Bangkok, Thailand, August 15-16, 2018, Proceedings 4*, pp. 72-84. Springer Singapore, 2019. https://doi.org/10.1007/978-981-13-3441-2_6

[18] Shi, Ziqiang, Liu Liu, and Rujie Liu. "Hodge and Podge: Hybrid supervised sound event detection with multi-hot MixMatch and composition consistence training." In *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 1-5. IEEE, 2021. https://doi.org/10.23919/Eusipco47968.2020.9287700

[19] Luo, Chuyao, Xutao Li, Yongliang Wen, Yunming Ye, and Xiaofeng Zhang. "A novel LSTM model with interaction dual attention for radar echo extrapolation." *Remote Sensing* 13, no. 2 (2021): 164. https://doi.org/10.3390/rs13020164

[20] Guirguis, Karim, Christoph Schorn, Andre Guntoro, Sherif Abdulatif, and Bin Yang. "SELD-TCN: Sound event localization & detection via temporal convolutional networks." In *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 16-20. IEEE, 2021. https://doi.org/10.23919/Eusipco47968.2020.9287716