
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2024 SCPE. Volume 25, Issues 5, pp. 3796–3811, DOI 10.12694/scpe.v25i5.3031

# DIFFCRNN: A NOVEL APPROACH FOR DETECTING SOUND EVENTS IN SMART HOME SYSTEMS USING DIFFUSION-BASED CONVOLUTIONAL RECURRENT NEURAL NETWORK

MARYAM M. AL DABEL *

**Abstract.** This paper presents a latent diffusion model and convolutional recurrent neural network for detecting sound event, fusing advantages of different networks together to advance security applications and smart home systems. The proposed approach underwent initial training using extensive datasets and subsequently applied transfer learning to adapt to the desired task to effectively mitigate the challenge of limited data availability. It employs the latent diffusion model to get a discrete representation that is compressed from the mel-spectrogram of audio. Subsequently a convolutional neural network (CNN) is linked as the front-end of recurrent neural network (RNN) which produces a feature map. After that, an attention module predicts attention maps in temporal-spectral dimensions level, from the feature map. The input spectrogram is subsequently multiplied with the generated attention maps for adaptive feature refinement. Finally, trainable scalar weights aggregate the fine-tuned features from the back-end RNN. The experimental findings show that the proposed method performs better compared to the state-of-art using three datasets: the DCASE2016-SED, DCASE2017-SED and URBAN-SED. In experiments on the first dataset, DCASE2016-SED, the performance of the approach reached a peak in $F1$ of 66.2% and $ER$ of 0.42. Using the second dataset, DCASE2017-SED, the results indicate that the $F1$ and $ER$ achieved 68.1% and 0.40, respectively. Further investigation with the third dataset, URBAN-SED, demonstrates that our proposed approach significantly outperforms existing alternatives as 74.3% and 0.44 for the $F1$ and $ER$.

**Key words:** Sound event detection, latent diffusion model, spectrogram, deep neural network.

**1. Introduction.** The objective of sound event (SE) detection is to provide devices with the capability to identify and classify acoustic environments. It can be characterized as the process of discerning the presence of both overlapping and non-overlapping sound events, as well as determining their respective initiation and duration intervals [44]. A distinct auditory occurrence that may be recognized as a distinct notion is referred to as a sound event [18]. In our everyday lives, we often encounter many forms of sound events as an example bird cries, dog barking, and human speech. In a real-world acoustic environment, the occurrence of these sound events may not be sequential but rather exhibit a tendency to regularly overlap. The SE detection systems may enhance the capabilities of current security applications, smart home systems and surveillance systems when applied jointly. In addition, they can be used in industrial environments to detect deficiencies in equipment and machinery.

Different approaches have been used to perform the SE detection task. There are two fundamentals to boost the overall classification performance of SE models: i) the extraction of acoustic features with robust characterization abilities, and ii) efficient classification techniques. The widely used features are linear predictive coding [32], linear predictive cepstral coefficients, discrete wavelet transform, mel frequency cepstral coefficients [32] and log-mel spectrograms. Turning to conventional classifiers, examples include support vector machines [13], Gaussian mixture models [15], hidden Markov models [11], multi-layer perceptron [42]. Such conventional models, however, are only useful to single acoustic events and small datasets [31]. These conventional classification models are less likely to satisfy the classification needs due to the large dataset size and audio complexity. The advances of machine learning has made it possible for neural network classification models to outperform more conventional classifiers, such as feedforward neural networks, recurrent neural networks [36], convolutional neural networks [22] and convolutional recurrent neural networks [2, 12, 21, 29]. Most SE research in recent years has employed deep learning-based classification models [4, 1]. While neural network-

*Department of Computer Science and Engineering, College of Computer Science and Engineering, University of Hafr Al Batin, Saudi Arabia (maldabel@uhb.edu.sa).

based classification models have been widely used in the field of acoustics, difficulties with sound detection is still exist include the following: i) the SE model has more parameters, more feature space dimensions, and larger datasets; ii) the temporal-frequency structure of sounds is very complex and may be continuous, abrupt, or periodic; and iii) inconsistency and ambiguous duration of sounds impact model classification performance. The main contributions of this paper are summarized as follows.

- Instead of choosing a random combination, as in earlier efforts, we take into account the pressure levels of the audio pairings when mixing them for data augmentation by applying the Latent Diffusion Model. This makes sure that the combined audio accurately represents both of the source audio.
- Combining the convolutional recurrent neural networks and an attention module in a unified framework that connect both the convolutional neural network layer and recurrent neural network layer.
- Conducting a series of comparative experiments to evaluate the performance of the proposed models.

The rest of this paper is set up as follows. Section 2 discusses and reviews previous related work. Section 3 introduces the proposed framework. Sections 4, 5 and 6 report and analyze the experimental results. Section 8 summarizes the work.

**2. Related Work.** Early work in SE detection typically aims at identifying only the dominating sound event among the overlapping sound events and their associated onset-offset periods. However, this strategy is less appropriate for applications that need the simultaneous detection of several sound events.

Widely known classifiers were used for such task including the combined Gaussian mixture model-hidden Markov model [16], non-negative matrix factorization [17], convolutional neural networks [48, 38], and recurrent neural networks [37, 47] networks. In [16], for instance, the combined Gaussian mixture model-hidden Markov model was employed to detect the overlapping sound events based on multiple restricted Viterbi passes. Whereas in [17], the combined Gaussian mixture model-hidden Markov model was designed to better identified the overlapping sound events by a preprocessing stage, in which a non-negative matrix factorization method was implemented as a stage to get multiple streams of source separated audio.

As deep learning methods advanced, many deep neural network-based solutions for the SE challenges were proposed. A multi-class multi-label feed-forward deep neural networks was applied in [6] such that each input frame was produced by concatenating multiple temporal-frames of the feature. This technique outperformed the best SE technique previously reported in [17]. Individual Gaussian mixture models are trained for each sound class when using generative classifiers like Gaussian mixture model. The sound class is determined during inference based on the greatest probable outcomes of the Gaussian mixture model. In [5], for each sound class in the dataset, several feed-forward deep neural networks classifiers were similarly trained. The cumulative outcomes of the various single-class feed-forward deep neural networks classifiers were used for the SE task during inference. The findings indicated that the multiple single-class technique performed slightly poor when compared to the multi-class multiple label approach.

Recently, in an attempt to enhance classification performance, a study based on the attention mechanism has also been conducted in the area of SE research. For instance, the Convolutional Long Short-Term Memory and Deep Neural Networks model incorporates the temporal attention mechanism that was first presented in [14]. The system can look at every time step and try to identify the high impact one so that it can be given more weight. Another model was suggested in [27] using an attention-based multi-stream network model. The attention weight is calculated based on the degree of energy change in the spectrogram. The authors in [49] noted that not all frame-level characteristics can affect environmental sound performance equally. In particular, there are other time frames, such as silent frames, noisy frames, can cause the robustness of the classification model to degrade and will also result in errors in the classification. Based on this assumption, It is crucial to record the primary temporal segment of the sound stream. While the aforementioned techniques do help with classification performance, they did not take into account the variation of the frequency bands and their effect on the process. In addition, the method in [46] was developed to stack multiple attention network to get robust features. A temporal attention mechanism was suggested in [28] for convolutional layers to boost the representative ability of convolutional neural networks by re-weighting the convolutional neural networks feature maps using dot-product operation along the time dimension from input spectrogram.

Deep learning models have the ability to acquire effective representations from raw data without the need for manual intervention. Convolutional neural networks (CNNs) can automatically extract feature maps through

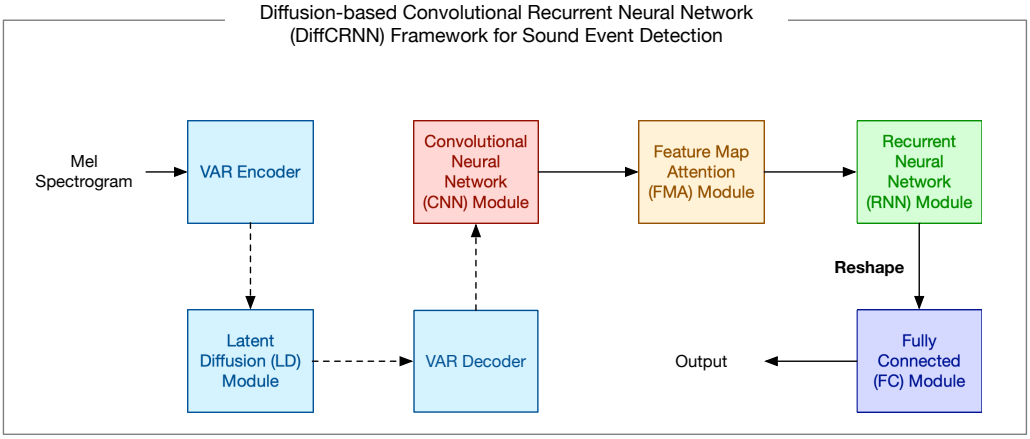


Fig. 3.1: The architecture of the proposed Diffusion-based Convolutional Recurrent Neural Network (DiffCRNN) system.

the convolution process, enabling them to capture the spatial features of input data [22, 39]. Furthermore, weight sharing significantly reduces the number of parameters in a convolutional neural network (CNN), thereby facilitating the training process of a CNN model compared to an equivalent dense neural network. Nevertheless, CNN-based models encounter challenges in capturing temporal dependencies when the input consists of time series data [20, 3]. Recurrent neural networks (RNNs) are extensively employed in various tasks, including text classification and speech recognition [10]. However, RNN-based models are limited in their ability to effectively extract features from raw data and face challenges with gradient vanishing and exploding when processing long time sequences [4]. Thus, this paper utilizes deep convolutional recurrent neural networks (namely DiffCRNN) to detect DiffCRNN by combining CNNs and RNNs. The DiffCRNN model utilizes convolutional layers to extract spatial features from raw data, while the recurrent layers are responsible for capturing the sequence information.

**3. DiffCRNN: Framework Design.** The architecture of the proposed Diffusion-based Convolutional Recurrent Neural Network (DiffCRNN) framework is illustrated in Figure 3.1. The framework has five main modules which are the latent diffusion based module, the convolutional neural networks (CNN) based module, the feature map attention based module, the recurrent neural network (RNN) based module and, finally, the fully connected layer based module.

In particular, the latent diffusion based module has three primary sub-modules: (i) encoder, (ii) latent diffusion model, and (iii) audio variational auto-encoder. The encoder is responsible for encoding the input description of the audio. Next, the process of reverse diffusion is used to construct a latent representation of the audio or audio prior from Gaussian noise, utilizing the textual representation. The audio variational auto-encoder subsequently employs the latent audio representation to yield a mel-spectogram. The primary objective of the CNN is to extract a multi-dimensional and higher-order features from the input spectrogram. Further, the FM-attention module learns the importance of each dimensions in a dynamic way, in which important feature map information is extracted and unimportant dimensions are discounted. The RNN module then attempts to acquire contextual information and anticipate both the start and offset times of sound events in a precise way. Finally, the output characteristics of the RNN serve as the input for the fully connected layer in order to get the classification score of the DiffCRNN system.

This section described the architecture in more detail. The latent diffusion based module is described in Section 3.1. The CNN module is reviewed in Section 3.2. Then, Section 3.3 explains the feature map attention based module. Finally, in Section 3.4, the RNN module is represented.

**3.1. Latent Diffusion Based Module.** The latent diffusion based module (LD) consists of three primary parts: the encoder, latent diffusion model, and audio variational auto-encoder.

**3.1.1. The encoder sub-module:.** The encoder ($E_\tau$) is the pre-trained large language models using FLAN-T5 [9] to obtain text encoding $\tau$. The token count and token-embedding size are $L$ and $d_\tau$, respectively. The use of gradient descent, which emulates the process of imitating characteristics, is of significant importance in the task of learning the relationship between textual and auditory concepts, without the need for fine-tuning the $E_\tau$, by treating each input sample as a distinct job. Enhanced pretraining techniques have the potential to enable the $E_\tau$, however, to prioritize essential information with less interference and enhanced contextual understanding. Therefore, the $E_\tau$ is held constant, on the assumption that the reverse diffusion process may acquire knowledge of the audio inter-modality mapping prior to its generation.

**3.1.2. The latent diffusion sub-module:.** The purpose of this sub-module is motivated by [40, 30] with the aim to produce the audio prior $s_0$ under the direction of text encoding $\tau$. This basically comes down to parameterized $p_0(s_0|\tau)$ via approximating the correct prior $q(s_0|\tau)$.

The mechanisms of forward and reverse diffusion allow the sub-module to accomplish the aforementioned. The forward diffusion consists of a series of Markov of Gaussians with predetermined noise parameters $0 < \delta_1 < \delta_2 < \cdots < \delta_N < 1$ to get more distorted iterations of the samples, $s_0$ as follows;

$$q(s_n|s_{n-1}) = \mathcal{N}(\sqrt{1-\delta_n}s_{n-1}, \delta_n\mathbf{I}), \tag{3.1}$$

$$q(s_n|s_0) = \mathcal{N}(\sqrt{\overline{\kappa_n}}s_0, (1-\overline{\kappa_n})\mathbf{I}), \tag{3.2}$$

such that $N$ denotes the quantity of forward diffusion iterations, $\kappa_n = 1 - \delta_n$, and $\overline{\kappa_n} = \prod_{i=1}^{n} \kappa_i$.

A more direct sampling of $s_n$ from sample noisier versions can be applied through re-parametrization using as follows;

$$s_n = \sqrt{\overline{\kappa_n}}s_0 + (1-\overline{\kappa_n})\epsilon, \tag{3.3}$$

such that the noise sample $\epsilon \in \mathcal{N}(\mathbf{0}, \mathbf{I})$. The last stage of the forward procedure yields $s_N \in \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The reverse method uses noise estimation ($\hat{\epsilon}_\theta$) to denoise and recover $s_0$ using loss function as follows;

$$\Omega = \sum_{n=1}^{N} \lambda_n \mathbb{E}_{\epsilon_n \in \mathcal{N}(\mathbf{0},\mathbf{I})} \|\epsilon_n - \hat{\epsilon}_\theta^{(n)}(s_n, \tau)\|_2^2. \tag{3.4}$$

such that $s_n$ is sampled from equation. 3.3 based on $\epsilon_n$ and $\lambda_n$ which are the standard normal noise and the weight of reverse step $n$, respectively. The $n$ is considered to be a measure of signal-to-noise ratio in respect to $\kappa_{1:N}$.

**3.1.3. The augmentation sub-module:.** In this sub-module, we synthesis more text-audio pairings by superimposing existing audio pairs and concatenating their captions. To avoid overpowering low-pressure samples, the pressure level of audio $R$ is considered. Audio sample ($x_1$) weight is determined as a relative pressure level:

$$p = (1 + 10^{\frac{R_1 - R_2}{20}})^{-1}, \tag{3.5}$$

such that $R_1$ and $R_2$ denotes the pressure levels of two used audio samples $y_1$ and $y_2$. This guarantees accurate depiction of the two audio samples after mixing.

In addition, the square of a sound wave's amplitude determines how much energy it has [45]. As a results, $y_1$ and $y_2$ were mixed as follows;

$$\text{mix}(y_1, y_2) = \frac{py_1 + (1-p)y_2}{\sqrt{p^2 + (1-p)^2}}. \tag{3.6}$$

**3.1.4. The free guidance sub-module:.** This sub-module is a classifier-free in which the input $\tau$ is used to rebuild the $s_0$ by directing the reverse diffusion. The contribution of text guidance to the noise level $\hat{\epsilon}_\theta$ is managed by a guidance scale $\upsilon$ with respect to unguided estimation throughout inference:

$$\hat{\epsilon}_\theta^{(n)}(s_n, \tau) = \upsilon \epsilon_\theta^{(n)}(s_n, \tau) + (1 - \upsilon)\epsilon_\theta^{(n)}(s_n). \tag{3.7}$$

**3.1.5. The decoder sub-module:.** In this sub-module, we implement the audio variational auto-encoder to converts the mel-spectogram of an audio sample into a $s_0$. The latent diffusion sub-module re-builts the $\hat{s}_0$ based on the $\tau$. The encoder and decoder are formulated of ResUNet blocks and are trained via maximizing evidence lower-bound and minimizing adversarial loss [25].

**3.2. CNN Module.** Assuming that $\boldsymbol{h}^{n-1}$ is the feature map of size $C^{n-1} \times P^{n-1} \times Q^{n-1}$ from the $(n-1)$-th layer, such that $C^{n-1}$ is the channel number and $P^{n-1} \times Q^{n-1}$ is the size of the feature map at the time and frequency axes, the result of the $n$-th convolutional layer is defined as

$$\boldsymbol{h}_j^n = \sum_{i=1}^{C^{n-1}} \boldsymbol{w}_{ij}^n * \boldsymbol{h}_i^{n-1} + b_j^n, \tag{3.8}$$

where $\boldsymbol{h}_j^n$ denotes the $j$-th channel of $\boldsymbol{h}^n$, $\boldsymbol{w}_{ij}$ denotes the $(i, j)$-th convolutional kernel, $*$ is the convolutional operation, and $b_j^n$ represents the bias at the $j$-th channel. In order to accelerate convergence, convolutional layers are typically followed by batch normalization and a ReLU activation function. Batch normalization can also increase the stability of CNN [8].

In order for the CNN model to function properly, the three-dimensional feature map that includes the channel, time frame, and feature vector must be transformed into a classification vector. It is possible, as mentioned in the previous section, to immediately flatten the feature map into a vector in order to reduce the number of dimensions. Flattening, on the other hand, could result in a sub-optimization due to the fact that it might preserve duplicate information. As a result, the time-frequency attention pooling will be covered here to produce a vector that is more compact and has less information that is redundant than the one that is generated by flattening.

The temporal-frequency global attention (TFGA) pooling in CNNs decreases the dimensionality of a feature map through measuring the contribution of each temporal-frequency unit. It is composed of two sub-modules: an attention sub-module, and a classification sub-module, which come typically after a set of convolutional layers and local average pooling layers. The attention sub-module has a two-dimensional convolutional layer with an output channel number equal to the number of classes $K$, and a kernel size of $1 \times 1$, which results in an attention tensor $A$. An activation function (softmax or sigmoid) is applied after the convolutional layer to yield a tensor $A^*$ with values in the range $[0, 1]$. Next, the tensor $A^*$ is normalized using

$$P_{kpq} = \frac{A^*_{kpq}}{\sum_{p=1}^{P_w} \sum_{q=1}^{Q_w} A^*_{kpq}}, \tag{3.9}$$

such that $P$ denotes the probability tensor. Moving to the classification sub-module, the feature map is transformed into a new one $C$ with the channel number of $K$ using an additional two-dimensional convolutional layer with a kernel size of $1 \times 1$. After that, the resultant classification tensor $C$ is multiplied by $P$ to determine the probability of each class by applying the following

$$p_k = \sum_{p=1}^{P_w} \sum_{q=1}^{Q_w} C_{kpq} \odot P_{kpq}, \tag{3.10}$$

Additionally, to complete a classification task, a softmax or log-softmax function is employed to operate on $C$ or $p$. In order to make more accurate predictions, the the time-frequency attention pooling can assess the contribution of each time-frequency bin to classification [20].

**3.3. Feature-map Attention Module.** In the Feature Map (FM-attention) algorithm, the multi-dimensional feature map $\boldsymbol{h}$ is acquired from CNN module, such that $C$ is the channel number and $T \times F$ represents at the time and frequency axes the size of the feature map. Then the high-order feature map $\boldsymbol{h}$ was input to the FM-attention model. The FM-attention has a Sigmoid activation layer and fully connected feedforward layer in order to compute the high impact weight of each feature dimension of $\boldsymbol{h}$ of size $C \times T \times F$. The high impact weight $U$ is the outputs of the Sigmoid layer, which is assigned to different feature dimensions. First, $\boldsymbol{h}$ is permuted into 3-dimensional tensor $\boldsymbol{h}'$ of size $T \times C \times F$. Subsequently, $\boldsymbol{h}'$ is flattened as a 2-dimensional tensor $\boldsymbol{h}''$ by fixing the dimension $T$.

Next, the input to the feedforward layer is $\boldsymbol{h}''$. The number of hidden units in this layer is set to $CF$. The dimension of weights $U$ is $M = CF$, which can be written as:

$$U = \{U_1, U_2, \ldots, U_d, \ldots, U_M\}, \tag{3.11}$$

where $U_m$ influences the $m$th dimensional feature of $\boldsymbol{h}''$, the expression of $U_m$ is:

$$U_m = \frac{\exp(O_m)}{\sum_{j=1}^{j=m} \exp(O_j)}, \tag{3.12}$$

The dimension of $\boldsymbol{h}''$ is $M$. The $j$th dimensional output of the Sigmoid activation layer is $O_j$. The high impact weight $U$ is repeated $T$ times, and its dimension $U$ results in $T \times C \times F$. The $U$ is reshaped to form $U'$, FM-attention vector, of size $T \times C \times F$. The outputs of the FM-attention module can be written as:

$$\boldsymbol{h}_{att} = U' \odot \boldsymbol{h}', \tag{3.13}$$

where "$\odot$" denotes the Hadamard product. Also, the outputs $\boldsymbol{h}_{att}$ of FM-attention module are fed into the RNN module.

**3.4. RNN Module.** The hidden state $h_t$ at the time step $t$, $t = 1, \ldots, T$, can be represented as

$$h_t = \sigma_h \left( \boldsymbol{w}_h x_t + \boldsymbol{u}_h h_{t-1} + b_h \right), \tag{3.14}$$

such that $\boldsymbol{w}_h$ and $\boldsymbol{u}_h$ denote the weights, $T$ represents the total number of time steps, $b_h$ denotes the bias, $h_{t-1}$ represents the previous hidden state at the time step $t-1$, $x_t$ denotes the input vector at the time step $t$, and $\sigma_h$ represents an activation function. In classification tasks, the final recurrent layer's hidden states are often merged into a single vector and sent on to a fully connected layer. Typically, a vector can be generated as the fully connected layer's input by either computing the average of the hidden states or extracting the hidden state at the most recent time step.

This simple RNN, however, is unable to process long-term context information owing to the exploding and vanishing gradient problem. For this reason, the Long Short-Term Memory (LSTM) RNN structure [19] and Gated Recurrent Units (GRU) RNN structure [50] were suggested to address such problem. The neurons in the simple RNN model is changed to memory blocks in the LSTM-RNN model, such that the memory blocks are connected recurrently. The LSTM, [19], is employed by replacing Equation 3.14 with the following steps: At the $t$-th time step, an LSTM unit comprises of an input gate $i_t$, an output gate $o_t$, a forget gate $f_t$, and a cell state $c_t$. The procedure of an LSTM unit is implemented as follow;

$$i_t = \sigma(\boldsymbol{w}_i x_t + \boldsymbol{u}_i h_{t-1} + b_i), \tag{3.15}$$

$$f_t = \sigma(\boldsymbol{w}_f x_t + \boldsymbol{u}_f h_{t-1} + b_f), \tag{3.16}$$

$$o_t = \sigma(\boldsymbol{w}_o x_t + \boldsymbol{u}_o h_{t-1} + b_o), \tag{3.17}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(\boldsymbol{w}_c x_t + \boldsymbol{u}_c h_{t-1} + b_c), \tag{3.18}$$

$$h_t = o_t \odot \tanh(c_t), \tag{3.19}$$

where $\odot$ denotes the element-wise multiplication. $i, f, o$ denote the input, forget and output gates' activation vectors, and $c, h$ denote cell and hidden states vectors.

A GRU-RNN structure, [50], comprises a reset gate $r_t$ and an update gate $z_t$ at the $t$ time step, unlike an LSTM cell. A GRU is established by

$$r_t = \sigma(\boldsymbol{w}_r x_t + \boldsymbol{u}_r h_{t-1} + b_r), \tag{3.20}$$

$$z_t = \sigma(\boldsymbol{w}_z x_t + \boldsymbol{u}_z h_{t-1} + b_z), \tag{3.21}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tanh(\boldsymbol{w}_h x_t + \boldsymbol{u}_h(r_t \odot h_{t-1}) + b_h), \tag{3.22}$$

In fact, a GRU has fewer parameters than an LSTM cell because it contains two gates in a single unit.

**4. Experimental Setup.** This section describes the experimental datasets in Section 4.1, evaluation metrics in Section 4.2 and experimental settings in Section 4.3 in the domain of SED. Experiments are run on publicly available datasets to verify the model's efficacy and the outcomes of this study's method are compared to those of previously published methods.

**4.1. Datasets.** The settings for real-time sound event detection system must be designed and customized to mimic the real-life noisy environments. This should be done by using equipments for recording at a number of different points and the sound sources are within a distance around the microphone points to generalize dataset with various recording environments. The system should also detect sound events regardless of position of the user.

To overcome the time-consuming issue of real-time sound event detection system, we present our results on three datasets namely, DCASE2016-SED [34], DCASE2017-SED [7] and URBAN-SED [41] that mimic the real-life noisy environments including everyday ambient noises that are separated into inside and outdoor settings.

**4.1.1. The DCASE2016-SED dataset.** The task3 of the DCASE2016 dataset [34] was utilized in this work to assess the performance of the DiffCRNN model. It includes everyday ambient noises that are separated into inside and outdoor settings. The DCASE2016 dataset's audio is mono and has a 44.1 kHz sample rate. A development set makes up 70% of the entire sample in both the DCASE2016 dataset, while an evaluation set makes up 30%. The four-fold cross-validation approach is employed in this work to train and test.

**4.1.2. The DCASE2017-SED dataset.** The task3 of the DCASE2017 dataset [7] was utilized in this work to assess the performance of the DiffCRNN model. It consists of everyday ambient noises that are separated into inside and outdoor settings. More street noises and human voices from authentic recordings may be found in the DCASE2017 collection. The sample frequency and duration of each audio file in the DCASE2017 dataset are both 44.1 kHz. Two typical settings are included in the DCASE2017: an inside residence and an outdoor residential neighborhood. A development set makes up 70% of the entire sample in the DCASE2017 dataset, while an evaluation set makes up 30%. The four-fold cross-validation approach is employed in this work to train and test.

**4.1.3. The URBAN-SED dataset.** The URBAN-SED [41] is a publicly available dataset for SED in urban environments. It is accompanied by detailed annotations, including onset and off-set times for each sound event, along with human generated accurate annotations.

**4.2. Evaluation Metrics.** We compare the performance using the commonly used metrics for SED presented in [33]. The segment-based $F1$-score ($F1$) and the error rate ($ER$) are used as assessment metrics in the experiment. Furthermore, $F1$ is the harmonic average of recall ($R$) and precision ($P$), which accept values between 0 and 1. The computation procedure is described as follows;

$$F1 = \frac{2P \cdot R}{P + R}, \tag{4.1}$$

Table 4.1: The structure of the neural settings in the DiffCRNN model.

| Layer Type | Configurations |
|---|---|
| Output | The output shape is (256, 6) |
| Recurrent | The number hidden unit is 32 |
| Recurrent | The number hidden unit is 32 |
| Merge | The mode is 'mul' |
| Repeat and Reshape | The output shape is (256, 128, 2) |
| Softmax activation | None |
| Feedforward | The number hidden unit is 256 |
| Reshape | The output shape is 256 & 256 |
| Permute | The output shape is 256, 128 & 2 |
| Max pooling | The sub-sampling rate is 2 |
| ReLU activation | None |
| Convolution | The filter number and kernel size is 128 & (3,3) |
| Max pooling | The sub-sampling rate is 2 |
| ReLU activation | None |
| Convolution | The filter number and kernel size is 128 & (3,3) |
| Max pooling | The sub-sampling rate is 5 |
| ReLU activation | None |
| Convolution | The filter number and kernel size is 128 & (3,3) |
| Merge | The mode is 'TF-Attention' |
| Multiply on the T/F direction | the mode is 'T-Attention' and 'F-Attention' |
| Softmax activation | None |
| Convolution | The filter number and kernel size is 1 & (1,1) |
| ReLU activation | None |
| Convolution | The filter number and kernel size is 32 & F(1,3) $\times$ 254/T(2,1) $\times$ 39 |
| Input | The input shape is (256,40) |

such that

$$P = \frac{\sum TP}{\sum TP + \sum FP'}, \tag{4.2}$$

and

$$R = \frac{\sum TP}{\sum TP + \sum FN'}, \tag{4.3}$$

where $TP$, $FP$, and $FN$ represent true positive, false positive, and false negative. The $ER$ denotes the number of samples classified incorrectly. The $ER$ is computed as;

$$ER = \frac{\sum_{t=1}^{T} S(t) + \sum_{t=1}^{T} I(t) + \sum_{t=1}^{T} D(t)}{\sum_{t=1}^{T} N(t)}, \tag{4.4}$$

in which $T$ represents how many audio events there are in segment $t$. Substitution events $S(t)$ represent the number of times the model incorrectly labels a sound event as a sound event. The term insertion event $(I(t))$ refers to an event $A$ that is currently not occurring in the tag annotation but is only identified in the model output. Deleted events, often known as $D(t)$, are sound events that were there but went undetected. The sum of the acoustic events from the annotations is $N(t)$.

**4.3. Experimental Settings.** All audio datasets used in this study are mono wave files at 44.1 kHz, and the dimension of the Log-Mel spectrograms is $40 \times 256$ where $(T = 256, F = 40)$. The overlapping frames are 50%, and the frame size is 40 ms.

Table 5.1: The performance comparison of the baseline and DiffCRNN with Latent Diffusion (+LD) and without (-LD).

| | DCASE2016-SED | | DCASE2017-SED | | URBAN-SED | |
|---|---|---|---|---|---|---|
| Method | $F1$ | $ER$ | $F1$ | $ER$ | $F1$ | $ER$ |
| CRNN | 50.4% | 0.36 | 53.2% | 0.38 | 62.3% | 0.40 |
| DiffCRNN(+LD) | 66.2% | 0.42 | 68.1% | 0.40 | 74.3% | 0.44 |
| DiffCRNN(-LD) | 60.4% | 0.45 | 59.3% | 0.55 | 64.2% | 0.41 |

The latent diffusion model is then given the characteristics. The Stable Diffusion U-Net architecture serves as the foundation for the 866M parameters that make up the diffusion model. In the U-Net model, we employ 8 channels and a cross-attention dimension of 1024. We train the AdamW optimizer with a linear learning rate scheduler and a 3e-5 learning rate. On the basis of the AudioCaps dataset, we train the model across 40 iterations, and we present the results for the checkpoint with the best validation loss, which we attained at iteration 40.

The Adam optimizer [24], which has a learning rate of 0.001, is used to feed the optimized features into CNNs. Total epochs are 100 and the learning rate ramp up during the first 20 epochs and ramp down during the remaining epochs. Batchsize is set to 64. A maximum of 3000 iterations are chosen through experiments to improve CNNs. Pytorch is used to build together the CNN architectures. Three CNN topologies - AlexNet [26], VGG-4 [43], and Net-4 - were used in the experiment. In order to reduce the efficacy of local max pooling layers, the Net-4 is a CNN structure with a stride of size 2 between the convolution layers. This Net-4, which is positioned between AlexNet and VGG-4, has a kernel size of 5×5. This is carried out to examine the impact of kernel size on performance and identify an ideal kernel size. The three-dimensional feature maps are converted into one-dimensional tensors via a global pooling layer that comes after the convolutional layers. As a result, fewer feature dimensions exist. Table 4.1 demonstrates the specific neural parameter settings for the DiffCRNN.

RNN, like CNN, is a highly effective neural network that is also utilized in SED tasks. The LSTM is a modified version of the RNN. Unlike standard RNN, LSTM can resolve the issue of long-term dependencies. Nevertheless, the interdependencies within time series data pose a challenge when attempting to utilize LSTM for parallel computation. The computation speed is significantly lower than that of the CNN. The GRU model is a distinct variant of RNN models. The accuracy of the detection task using the GRU model will be slightly affected while ensuring high speed for the DiffCRNN.

**5. Main Results.** The performance of the DiffCRNN model was assessed under the following experimental scenarios:
**(1):** with/without LD,
**(2):** with/without FM strategy,
**(3):** different pooling methods for CNNs classifiers,
**(4):** different RNNs classifiers,
**(5):** with/without Fine-tuning,
**(6):** with/without data augmentation, and
**(7):** with the other state-of-the-art SED methods.
We designed these experiments on the DCASE2016-SED dataset, DCASE2017-SED dataset and URBAN-SED dataset in which the baseline system is CRNN.

**5.1. Comparison of DiffCRNN With/Without Latent Diffusion.** The assessment results of the development set for DCASE2016-SED and DCASE2017-SED, comparing DiffCRNN with and without LD, are shown in Table 5.1. The used features were Log-Mel spectrograms. During the experimental phase, the CRNN method was used as the baseline to assess the classification performance while using LD.

LD demonstrated superior performance in terms of both $F1$ and $ER$ values when compared to the two situations. During the study conducted on the DCASE2016-SED dataset, the LD achieved a peak $F1$ score

Table 5.2: The performance comparison of the baseline and DiffCRNN with Feature Mapping Attention Algorithm (+FM) and without (-FM).

| | DCASE2016-SED | | DCASE2017-SED | | URBAN-SED | |
|---|---|---|---|---|---|---|
| Method | $F1$ | $ER$ | $F1$ | $ER$ | $F1$ | $ER$ |
| CRNN | 50.4% | 0.36 | 53.2% | 0.38 | 62.3% | 0.40 |
| DiffCRNN(+FM) | 66.2% | 0.42 | 68.1% | 0.40 | 74.3% | 0.44 |
| DiffCRNN(-FM) | 55.3% | 0.48 | 56.1% | 0.51 | 65.1% | 0.48 |

Table 5.3: The performance comparison of various pooling methods for CNNs.

| | DCASE2016-SED | | DCASE2017-SED | | URBAN-SED | |
|---|---|---|---|---|---|---|
| Classifier (+Pooling Type) | $F1$ | $ER$ | $F1$ | $ER$ | $F1$ | $ER$ |
| AlexNet (+GM) | 58.1% | 0.42 | 63.2% | 0.51 | 66.4% | 0.50 |
| AlexNet (+GA) | 57.8% | 0.45 | 58.5% | 0.55 | 67.2% | 0.48 |
| AlexNet (+TFGA) | 66.2% | 0.42 | 68.1% | 0.40 | 74.3% | 0.44 |
| VGG-4 (+GM) | 58.5% | 0.43 | 63.7% | 0.52 | 65.5% | 0.46 |
| VGG-4 (+GA) | 59.0% | 0.46 | 63.2% | 0.57 | 67.1% | 0.49 |
| VGG-4 (+TFGA) | 60.2% | 0.40 | 65.9% | 0.42 | 69.2% | 0.48 |
| Net-4 (+GM) | 57.2% | 0.40 | 62.9% | 0.45 | 67.2% | 0.50 |
| Net-4 (+GA) | 56.2% | 0.41 | 57.9% | 0.50 | 66.7% | 0.47 |
| Net-4 (+TFGA) | 60.3% | 0.43 | 64.5% | 0.47 | 67.3% | 0.45 |

of 66.2% and a $ER$ value of 0.42. The DCASE2017-SED dataset yielded a $F1$ score of 68.1% and an error rate ($ER$) of 0.40. The experiment on the URBAN-SED dataset, the LD reached a peak $F1$ score of 74.3% and a $ER$ value of 0.44. The experimental findings demonstrate that the use of LD significantly improved the classification performance.

**5.2. Comparison of DiffCRNN With/Without Feature Mapping Attention Algorithm.** The findings of evaluating the development set for DCASE2016-SED and DCASE2017-SED for comparing DiffCRNN With/Without FM approach are shown in Table 5.2. Log-Mel spectrograms were used as the features. In the course of the study, the classification impact of using FM method was compared using the same CRNN model as the baseline.

The $F1$ and $ER$ values were enhanced by the FM technique in comparison to the two cases. The FM method performed best in tests using the DCASE2016-SED dataset, with a maximum $F1$ of 66.2% and $ER$ of 0.42. Its $F1$ and $ER$, using the DCASE2017-SED dataset, were 68.1% and 0.40, respectively. During the study conducted on the URBAN-SED dataset, the FM achieved a peak $F1$ score of 74.3% and a $ER$ value of 0.44. The use of FM approach improved the classification performance, according to experiment data.

**5.3. Comparison of Different Pooling Methods for CNNs Classifiers in the DiffCRNN Model.** Table 5.3 shows the results of the evaluation of the development set for DCASE2016-SED, DCASE2017-SED and URBAN-SED. We can see that almost every one of our pooling models does better than the other. The TFGA model works better at AlexNet than the GM and GA models, and it was used to make CNN. But at VGG-4, the TFGA model gives way to GM. One reason might be that the larger number of hyper parameters in VGG-4 with TFGA pooling leads to overfitting. When it comes to the Net-4 model, the developed CNN gets the best results. This means that CNNs with a kernel size of five and no GM between convolutional layers seem to be better suited for this task of classifying acoustic scenes. Also, the developed CNN gets 56.2% and 60.3% accuracy for the DCASE2016-SED, 57.9% and 64.5% accuracy for DCASE2017-SED, and 66.7% and

Table 5.4: The performance comparison of LSTM-RNNs and GRU-RNNs of the DiffCRNN Model.

| | DCASE2016-SED | | DCASE2017-SED | | URBAN-SED | |
|---|---|---|---|---|---|---|
| Method (+RNN Classifie) | $F1$ | $ER$ | $F1$ | $ER$ | $F1$ | $ER$ |
| CRNN | 50.4% | 0.36 | 53.2% | 0.38 | 62.3% | 0.40 |
| DiffCRNN(+GRU) | 66.2% | 0.42 | 68.1% | 0.40 | 74.3% | 0.44 |
| DiffCRNN(+LSTM) | 60.4% | 0.45 | 59.3% | 0.55 | 71.4% | 0.42 |

Table 5.5: The performance comparison between fine-tuned and non fine-tuned models on the development set.

| | DCASE2016-SED | | DCASE2017-SED | | URBAN-SED | |
|---|---|---|---|---|---|---|
| Method | $F1$ | $ER$ | $F1$ | $ER$ | $F1$ | $ER$ |
| CRNN | 50.4% | 0.36 | 53.2% | 0.38 | 62.3% | 0.40 |
| DiffCRNN(+Finetuning) | 66.2% | 0.42 | 68.1% | 0.40 | 74.3% | 0.44 |
| DiffCRNN(-Finetuning) | 60.1% | 0.45 | 65.2% | 0.44 | 69.1% | 0.46 |

67.3% accuracy for URBAN-SED.

**5.4. Comparison of LSTM-RNNs and GRU-RNNs of the DiffCRNN Model.** Table 5.4 represents the results of the evaluation of the development set for DCASE2016-SED, DCASE2017-SED and URBAN-SED by comparing of different RNN classifiers. The used features was Log-Mel spectrograms. During the experimentation procedure, the efficacy of using various RNN classifiers for classification was compared using the same CRNN method as the baseline.

The experimental findings of DCASE2017-SED provide the mean accuracy on the 4-fold partitioned development set, as determined by the official evaluation metrics. Both RNN models consist of three recurrent layers with output channels of 256, 1024, and 256. Compared with the two scenarios, the GRU-RNNs classifiers improved $F1$ and $ER$ values. In experiments on the DCASE2016-SED dataset, the performance of the GRU-RNNs classifiers reached a maximum $F1$ of 66.2% and $ER$ of 0.42. Using the DCASE2017-SED dataset, its $F1$ and $ER$ were 68.1% and 0.40, respectively. Moving to the study on the URBAN-SED dataset, the performance of the GRU-RNNs classifiers reached its peak with $F1$ of 74.3% and $ER$ of 0.44. The outcomes of the studies show that the performance of classification was improved by the usage of GRU-RNNs. When training is terminated at various epochs, the performances of LSTM-RNNs and GRU-RNNs on a set of feature sets are compared.

**5.5. Comparison of DiffCRNN With/Without Fine-tuning.** Table 5.5 demonstrates the results of the evaluation of the development set for DCASE2016-SED, DCASE2017-SED and URBAN-SED for comparing of DiffCRNN With/Without Fine-tuning.

The results of experiments indicate that the use of Fine-tuning enhanced the classification performance. Nevertheless, it is crucial to acknowledge that achieving greater results on the restricted sample of the training dataset does not always imply superior overall performance. A model that has the ability to create wider ranges of sounds may have worse performance on the development set, but having superior generalization capabilities.

**5.6. Comparison of DiffCRNN With/Without Data Augmentation.** Table 5.6 demonstrates the results of the evaluation of the development set for DCASE2016-SED, DCASE2017-SED and URBAN-SED for comparing of DiffCRNN With/Without data augmented.

The results of experiments show that the use of data augmented increased the classification performance. For data augmentation, AudioGen employs an approach called mixup, where it combines pairs of audio samples and concatenates their processed text captions. This results in the creation of fresh paired data, which leads to improved performance overall.

Table 5.6: The performance comparison between data augmented and non-data augmented models on development set.

| | DCASE2016-SED | | DCASE2017-SED | | URBAN-SED | |
|---|---|---|---|---|---|---|
| Method | $F1$ | $ER$ | $F1$ | $ER$ | $F1$ | $ER$ |
| CRNN | 50.4% | 0.36 | 53.2% | 0.38 | 62.3% | 0.40 |
| DiffCRNN(+AudioGen) | 66.2% | 0.42 | 68.1% | 0.40 | 74.3% | 0.44 |
| DiffCRNN(-AudioGen) | 59.1% | 0.49 | 60.3% | 0.46 | 67.2% | 0.45 |

Table 5.7: Summary of the State-of-the-art SED Methods Used for Comparsion.

| SED Approach | Description |
|---|---|
| Log-Mel+CaspNet [23] | It is based on Capsule Neural Networks (CaspNet), the input feature is Log-Mel spectrograms, and it is the winning model for DCASE2016-SED. |
| Log-Mel-CRNN [2] | It is based on CRNN, the input feature is Log-Mel spectrograms, and it is the winning model for DCASE2017-SED. |
| CRNN-CWin [35] | It utilizes the Transformer encoder, which consists of multiple self-attention modules, the input feature is Log-Mel spectrograms, and it is the state-of-the-art model for URBAN-SED. |

Table 5.8: The performance comparison between DiffCRNN Model and the state-of-the-art SED methods

| | DCASE2016-SED | | DCASE2017-SED | | URBAN-SED | |
|---|---|---|---|---|---|---|
| Method | $F1$ | $ER$ | $F1$ | $ER$ | $F1$ | $ER$ |
| Log-Mel+CaspNet [23] | 47.8% | 0.81 | - | - | - | - |
| Log-Mel-CRNN [2] | - | - | 41.7% | 0.79 | - | - |
| CRNN-CWin [35] | - | - | - | - | 65.7% | 0.71 |
| Our DiffCRNN | 66.2% | 0.42 | 68.1% | 0.40 | 74.3% | 0.44 |

**5.7. Comparison of the DiffCRNN Model with the State-of-the-art SED Methods.** The DiffCRNN model was then compared with advanced SED methods. Other compared models are specified in Table 5.7 where the baselines and the winning models are outlined.

The experimental results in Table 5.8 show that the proposed DiffCRNN model outperforms other methods for both the baselines and the winning models.

**6. Ablation Study.** We conduct ablation experiments on DCASE2017 Task3 to study DiffCRNN in detail. All experiments use the pre-trained ResUNet backbone features for training and inference without further specification. The encoder and decoder are formulated of ResUNet blocks and are trained via maximizing evidence lower-bound and minimizing adversarial loss

**6.1. Ablation Study on Diffusion Strategy.** Diffusion Strategy Due to the inherent iteration based design with the decoder, we discuss and compare two diffusion strategies: (i) Noisy event latents in the continuous space (CS) (referred as DiffCRNN-CS, our model). (ii) Noisy event latent event in the discrete space (DS) (referred as DiffCRNN-DS). In addition, we distort the event latents using random shuffle as the noise in the forward diffusion step. In order to assess the impact of the diffusion strategy through experimentation, we

Table 6.1: Effect of the number of iteration on the performance for DCASE2016-SED, DCASE2017-SED and URBAN-SED Test set on noisy event latents in the continuous space (CS) and noisy event latent event in the discrete space (DS).

| Method | # Iteration | DCASE2016-SED $F1$ | DCASE2017-SED $F1$ | URBAN-SED $F1$ |
|---|---|---|---|---|
| DiffCRNN-CS | 10 | 63.2% | 64.6% | 71.3% |
| | 20 | 65.1% | 66.1% | 72.1% |
| | 30 | 65.3% | 67.2% | 72.6% |
| | 40 | **66.2%** | **68.1%** | **74.3%** |
| | 50 | 64.2% | 66.3% | 73.4% |
| DiffCRNN-DS | 10 | 57.1% | 63.3% | 69.2% |
| | 20 | **64.3%** | **66.3%** | 70.1% |
| | 30 | 59.1% | 65.3% | **71.4%** |
| | 40 | 58.2% | 65.7% | 68.2% |
| | 50 | 60.2% | 64.0% | 68.9% |

Table 6.2: Effect of scaling the noise factor on the performance for DCASE2016-SED, DCASE2017-SED and URBAN-SED Test set on noisy event latents in the continuous space (CS) and noisy event latent event in the discrete space (DS).

| Method | Noise scale | DCASE2016-SED $F1$ | DCASE2017-SED $F1$ | URBAN-SED $F1$ |
|---|---|---|---|---|
| DiffCRNN-CS | 0.1 | 64.1% | 66.2% | 70.1% |
| | 0.2 | 64.6% | 66.1% | 71.1% |
| | 0.3 | 65.2% | 66.8% | 71.9% |
| | 0.4 | **66.2%** | **68.1%** | **74.3%** |
| | 0.5 | 63.2% | 66.3% | 70.4% |
| DiffCRNN-DS | 0.1 | 60.1% | 58.8% | 69.0% |
| | 0.2 | 59.3% | 62.3% | **70.1%** |
| | 0.3 | 61.4% | 63.3% | 67.4% |
| | 0.4 | **64.8%** | **66.3%** | 69.0% |
| | 0.5 | 60.7% | 64.0% | 68.9% |

conduct tests on both variants using varying numbers of iteration. Table 6.1 shows that both variants achieve the best performance at the 40 iteration for the DiffCRNN-CS.

**6.2. Ablation Study on Signal Scaling.** The signal scaling factor controls the noise scaling of the diffusion process. We study the influence of scaling factors. The results in Table 6.2 illustrate that the scaling factor of 0.4 reaches the highest performance in $F1$ metric for DiffCRNN-CS, whereas for DiffCRNN-DS the best performance is obtained for a scaling factor of 0.2 in URBAN-SED whilst achieving the best $F1$ score for a scaling factor of 0.4 in both DCASE2016-SED and DCASE2017-SED. This implies a correlation between optimal scaling and the diffusion strategy.

**7. Discussion.** While the DiffCRNN method offers numerous benefits, its utilization also poses certain challenges. The following are the primary difficulties associated with DiffCRNN: The DiffCRNN has a high computational complexity, particularly when compared to less complex models such as CNNs. This can render them difficult to train and implement on low-power devices. The architectural design of DiffCRNN presents challenges that necessitate thorough consideration of the arrangement and integration of forward and reverse diffusion, convolutional, and recurrent layers. Selecting exemplary architecture can be a long and tedious

task. Training DiffCRNN can pose challenges, particularly when dealing with large datasets. The model may experience issues such as over-fitting, which occurs when the model becomes too closely aligned with the training data and fails to effectively apply its knowledge to new data. The DiffCRNN, like other diffusion models and deep learning models, presents limited interpretability, making it difficult to understand and explain its inner workings. Comprehending the rationale behind a model's specific predictions can pose challenges and hinder certain applications. The aforementioned challenges can be overcome with careful experimental settings that we implement in Section 4.3.

**8. Conclusions.** In this study, we combine the benefits of several networks to provide a latent diffusion model and convolutional recurrent neural network for sound event detection to enhance security applications and smart home systems. To overcome the problem of data scarcity, the system was first trained on large datasets and then used transfer learning to adjust to the target job. The suggested detection framework first trains a discrete representation compressed from the audio mel-spectrogram using the latent diffusion model. Next, a CNN is integrated as the front-end of a RNN. Next, the back-end RNN receives the feature map that the front-end CNN has learnt. Following that, an intermediate feature map is used by an attention module to forecast attention maps in two different dimensions: temporal and spectral. The input spectrogram is then multiplied by the attention maps in order to perform adaptive feature refining. Ultimately, the refined characteristics from the rear-end RNN are combined using trainable scalar weights. The experimental results demonstrate that the proposed method outperforms both the state-of-the-art and the baseline CRNN. Using the DCASE2016-SED dataset as an example, the system's performance peaked at 66.2% $F1$ and 0.42 $ER$. Its $F1$ and $ER$, using the DCASE2017-SED dataset, were 68.1% and 0.40, respectively. Further investigation with the URBAN-SED dataset shows that our proposed method outperforms existing alternatives with 74.3% and 0.44 for the $F1$ and $ER$.

Our future work will design a DiffCRNN system based on mobile terminal devices considering the fact that people use mobile terminals as internet access devices most of the time in daily life. We will adopt the client/server structure in order to allow the mobile device as the end-user to record and collect the user's voice signal. Then, it can be sent to the desktop computer as a server for neural network calculation, and finally, the result of event sources is returned to the user terminal.

## REFERENCES


[1] O. O. Abayomi-Alli, R. Damaševičius, A. Qazi, M. Adedoyin-Olowe, and S. Misra, *Data augmentation and deep learning methods in sound classification: A systematic review*, Electronics, 11 (2022), p. 3795.

[2] S. Adavanne and T. Virtanen, *A report on sound event detection with different binaural features*, arXiv, arXiv:1710.02997 (2017).

[3] N. Akhtar and U. Ragavendran, *Interpretation of intelligence in CNN-pooling processes: a methodological survey*, Neural computing and applications, 32 (2020), pp. 879–898.

[4] J. Baumann, P. Meyer, T. Lohrenz, A. Roy, M. Papendieck, and T. Fingscheidt, *A new dcase 2017 rare sound event detection benchmark under equal training data: Crnn with multi-width kernels*, in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 865–869.

[5] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, *Multi-label vs. combined single-label sound event detection with deep neural networks*, in 2015 23rd European Signal Processing Conference (EUSIPCO), 2015, pp. 2551–2555.

[6] ———, *Polyphonic sound event detection using multi label deep neural networks*, in IEEE International Joint Conference on Neural Networks, 2015, pp. 1–7.

[7] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, *Convolutional recurrent neural networks for polyphonic sound event detection*, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 25 (2017), pp. 1291–1303.

[8] E. Chai, M. Pilanci, and B. Murmann, *Separating the effects of batch normalization on cnn training speed and stability using classical adaptive filter theory*, in 2020 54th Asilomar Conference on Signals, Systems, and Computers, 2020, pp. 1214–1221.

[9] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, et al., *Scaling instruction-finetuned language models*, arXiv preprint arXiv:2210.11416, (2022).

[10] D. De Benito-Gorrón, D. Ramos, and D. Toledano, *A Multi-Resolution CRNN-Based Approach for Semi-Supervised Sound Event Detection in DCASE 2020 Challenge*, IEEE Access, 9 (2021), pp. 89029–89042.

[11] N. Degara, M. E. Davies, A. Pena, and M. D. Plumbley, *Onset event decoding exploiting the rhythmic structure of polyphonic music*, IEEE Journal of Selected Topics in Signal Processing, 5 (2011), pp. 1228–1239.

[12] H. Dinkel and K. Yu, *Duration robust weakly supervised sound event detection*, in 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 311–315.
[13] G. Guo and S. Li, *Content-based audio classification and retrieval by support vector machines*, IEEE Trans. Neural Netw., 14 (2003), pp. 209–215.
[14] J. Guo, N. Xu, L. Li, and A. Alwan, *Attention based cldnns for short-duration acoustic scene classification*, in Interspeech, 2017, pp. 469–473.
[15] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, *Audio context recognition using audio event histograms*, in 18th European Signal Processing Conference, 2010, pp. 1272–1276.
[16] ———, *Context-dependent sound event detection*, EURASIP Journal on Audio, Speech, and Music Processing, (2013), pp. 1–13.
[17] T. Heittola, A. Mesaros, T. Virtanen, and A. Eronen, *Sound event detection in multisource environments using source separation*, in Machine Listening in Multisource Environments, 2011, pp. 36–40.
[18] T. Heittola, A. Mesaros, T. Virtanen, and M. Gabbouj, *Supervised model training for overlapping sound events based on unsupervised source separation*, in IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 8677–8681.
[19] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation, 9 (1997), pp. 1735–1780.
[20] H. Ide and T. Kurita, *Improvement of learning for cnn with relu activation by sparse regularization*, in 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2684–2691.
[21] K. Imoto, S. Mishima, Y. Arai, and R. Kondo, *Impact of sound duration and inactive frames on sound event detection performance*, in 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 860–864.
[22] I.-Y. Jeong, S. Lee, Y. Han, and K. Lee, *Audio event detection using multiple-input convolutional neural network*, in Detection and Classification of Acoustic Scenes and Events, 2017, pp. 51–54.
[23] W. Jin, J. Liu, M. Feng, and J. Ren, *Polyphonic sound event detection using capsule neural network on multi-type-multi-scale time-frequency representation*, in 2022 IEEE 2nd International Conference on Software Engineering and Artificial Intelligence (SEAI), 2022, pp. 146–150.
[24] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).
[25] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, arXiv preprint arXiv:1312.6114, (2013).
[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, Communications of the ACM, 60 (2017), pp. 84–90.
[27] X. Li, V. Chebiyyam, and K. Kirchhoff, *Multi-stream network with temporal attention for environmental sound classification*, in Interspeech, 2019, pp. 3604–3608.
[28] X. Li, V. Chebiyyam, and K. Kirchhoff, *Multi-stream network with temporal attention for environmental sound classification*, arXiv, arXiv:1901.08608 (2019).
[29] H. Lim, J.-S. Park, and Y. Han, *Rare sound event detection using 1d convolutional recurrent neural networks. in proceedings of the detection and classification of acoustic scenes and events 2017, munich, germany, 16 november 2017; pp. 80–84.*, in Detection and Classification of Acoustic Scenes and Events, 2017, pp. 80–84.
[30] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley, *Audioldm: Text-to-audio generation with latent diffusion models*, arXiv preprint arXiv:2301.12503, (2023).
[31] Y. Liu, J. Tang, Y. Song, and L. Dai, *A capsule based approach for polyphonic sound event detection*, in 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2018, pp. 1853–1857.
[32] L. Luo, L. Zhang, M. Wang, Z. Liu, X. Liu, R. He, and Y. Jin, *A System for the Detection of Polyphonic Sound on a University Campus Based on CapsNet-RNN*, IEEE Access, 9 (2021), pp. 147900–147913.
[33] A. Mesaros, T. Heittola, and T. Virtanen, *Metrics for polyphonic sound event detection*, Applied Sciences, 6 (2016), p. 162.
[34] A. Mesaros, T. Heittola, and T. Virtanen, *Tut database for acoustic scene classification and sound event detection*, in 24th European Signal Processing Conference (EUSIPCO), 2016, pp. 1128–1132.
[35] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, *Weakly-supervised sound event detection with self-attention*, in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 66–70.
[36] G. Parascandolo, H. Huttunen, and T. Virtanen, *Recurrent neural networks for polyphonic sound event detection in real life recordings*, in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 6440–6444.
[37] ———, *Recurrent neural networks for polyphonic sound event detection in real life recordings*, in IEEE international conference on acoustics, speech and signal processing, 2016, pp. 6440–6444.
[38] H. Phan, L. Hertel, M. Maass, and A. Mertins, *Robust audio event recognition with 1-max pooling convolutional neural networks*, in INTERSPEECH, 2016.
[39] Z. Ren, Q. Kong, J. Han, M. Plumbley, and B. Schuller, *CAA-Net: Conditional atrous CNNs with attention for explainable device-robust acoustic scene classification*, IEEE Transactions on Multimedia, 23 (2020), pp. 4131–4142.
[40] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, in IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10684–10695.
[41] J. Salamon, C. Jacoby, and J. P. Bello, *A dataset and taxonomy for urban sound research*, in Proceedings of the 22nd ACM international conference on Multimedia, 2014, pp. 1041–1044.
[42] P. Sidiropoulos, V. Mezaris, I. Kompatsiaris, H. Meinedo, M. Bugalho, and I. Trancoso, *On the use of audio events for improving video scene segmentation*, in Analysis, Retrieval and Delivery of Multimedia Content, Springer, 2013,

pp. 3–19.
[43] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv, arXiv:1409.1556 (2014).
[44] B. T. Szabó, S. L. Denham, and I. Winkler, *Computational models of auditory scene analysis: a review*, Frontiers in Neuroscience, 10 (2016), p. 524.
[45] Y. Tokozume, Y. Ushiku, and T. Harada, *Learning from between-class examples for deep sound recognition*, arXiv preprint arXiv:1711.10282, (2017).
[46] J. Wang and S. Li, *Self-attention mechanism based system for dcase2018 challenge task1 and task4*, Proc. DCASE Challenge, (2018), pp. 1–5.
[47] Y. Wang, L. Neves, and F. Metze, *Audio-based multimedia event detection using deep recurrent neural networks*, in IEEE international conference on acoustics, speech and signal processing, 2016, pp. 2742–2746.
[48] H. Zhang, I. McLoughlin, and Y. Song, *Robust sound event recognition using convolutional neural networks*, in IEEE international conference on acoustics, speech and signal processing, 2015, pp. 559–563.
[49] Z. Zhang, S. Xu, S. Zhang, T. Qiao, and S. Cao, *Attention based convolutional recurrent neural network for environmental sound classification*, Neurocomputing, 453 (2021), pp. 896–903.
[50] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang, *Machine health monitoring using local feature-based gated recurrent unit networks*, IEEE Transactions on Industrial Electronics, 65 (2017), pp. 1539–1548.

*Edited by:* Kavita Sharma
*Special issue on:* Recent Advance Secure Solutions for Network in Scalable Computing
*Received:* Dec 11, 2023
*Accepted:* Apr 24, 2024