EXPLORATION INTO RARE SOUND DETECTION USING LSTM-RNN

By

Yikuan Chen

Senior Thesis in Electrical Engineering

University of Illinois at Urbana-Champaign

Advisor: Deming Chen

April 2018

Abstract

Rare Audio Event Detection (AED) plays a crucial role in domestic and public security applications. The goal of this research is to recognize key acoustic events using Long Short-Term-Memory Recurrent Neural Network (LSTM-RNN) based classifiers. We compared different existing methods on rare sound recognition, such as Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), zero-phase signal method and neural networks. Specifically, we investigated different neural network architectures, such as feedforward DNN, RNN, LSTM-RNN, bi-directional RNN etc. After experimenting with different neural network structures and different acoustic features, we propose a mixed neural network which consists of multiple subnets, each dedicated to recognizing one type of sound. Each subnet contains multi-input layers, feed forward layers, LSTM-RNN layer and output smoothing units. The final classification will be given based on the output of all subnets. Different acoustic features are fed into the network at different input layers to enhance the efficiency. Our model has exceeded the baseline performance of Detection and Classification of Acoustic Scenes and Events (DCASE) 2017 competition. However, there still exists a performance gap between our model and the current best model, and we are currently analyzing the advantages and drawbacks of our model and the top-ranking model.

Subject Keywords: LSTM; recurrent neural network; scream detection; gunshot detection; audio event detection

Contents

1. Introduction

The growing concern about public security has stimulated a plethora of research on automatic surveillance systems. One important function of an automatic surveillance system is the ability to perform unsupervised detection of certain target events which signals abnormality or danger. In a typical surveillance application, video/image data and/or sound data are collected from sensors (e.g. video cameras and microphones) and processed by a computer or specially designed hardware which implements a complete model which process the raw data, extract useful information and produces judgment about what events do the collected data represent.

For detecting certain events, such as a gunshot or baby crying, audio information is preferred to video information based on the following reasons: (1) computational burden and memory consumption is lower for audio than for video information, and thus it could be implemented on embedded computing system with lower cost; (2) audio monitoring does not require good illuminating condition; (3) camera has limitation of visible angle, but microphones could collect sound coming from any direction. Also, as study and application of neural networks become increasingly popular, researchers have investigated a variety of approaches, such as computer vision and automatic acoustic event detection/recognition, to performing security surveillance with the help of neural networks. Compared to traditional methods such as the Gaussian Mixture Model (GMM) and the Hidden Markov Model (HMM), neural network models not only are shown to be more robust on classification tasks, but also have shown the ability to perform end-to-end event detection. Therefore, in this research, we focus on sound event detection using neural networks. Specifically, our goal is to combine advantages of different types of neural network models, which are introduced below, to enhance the robustness of sound event detection.

1.1 Neural Network

A neural network with multiple input dimensions is a universal multiple variable continuous function approximator has the ability to learn high-order correlations between input data and output data [1]. It is proved to be very powerful for classification jobs such as image tagging and automatic speech recognition [2], [3]. A basic feedforward neural network (FFNN) can be viewed as a multivariable function:

$$\vec{y} = F(\vec{x}) \tag{1.1}$$

An FFNN contains one or several layers. Each layer takes a numerical input vector, multiple it with a weight matrix, add a bias vector and apply an activation function, such as sigmoid or tanh function, to yield a numerical vector output. Each placeholder to hold a number in each layer is called a “neuron”. If there are

more than one layer, then the output of a layer could be used as the input of its next layer. This process is called "forward propagation". The layers other than the final layer are called "hidden layers". The output vector of the last layer, $\vec{y}$, is the output of the entire function. The output may or may not be the desired correct value, so the network needs to be trained to be able to produce the desired output given any input. That is, each value in the weight matrices and bias vectors needs certain adjustment to make the function best approximate the underlying correlation between any input and corresponding desired output. These values are usually initialized with zero or random numbers and are updated using certain algorithms such as "gradient back propagation". After an iterative training process, the network will be able to approximate the desired output given any input in the training set. Usually, the network may not produce the desired output for a novel input (i.e. an input that is not in the training dataset), but the error can be reduced by increasing the amount of training data, which let the neural network to be exposed to more instances of relevant data and learn a better abstracted correlation between the input and output pairs. Therefore, preparing representative and comprehensive training data is a crucial job of neural network training.

For certain tasks which require as sequence-to-sequence mapping or temporal data processing, it has been shown that Recurrent Neural Network (RNN) and its variations, such as Long Short-Term Memory (LSTM) [4] usually outperforms basic FFNN. It has also been shown that the Convolutional Neural Network (CNN) has good capability to extract acoustic features from the frequency domain representation of sound [5]. Therefore, this research will explore the effectiveness of sound event detection using a mixed neural network which combines CNN and LSTM-RNN.

1.1.1 Long Short-Term Memory Recurrent Neural Network (LSTM-RNN)

Recurrent Neural Network (RNN), as its name suggests, adds temporal dimension to the network structure. For each hidden layer, the output, $\vec{h}_i(t)$, is computed using both input, $\vec{x}_i(t)$, and the previous output, $\vec{h}_i(t-1)$. For i^{th} hidden layer and timestep t, the output has the following form:

$$\vec{h}_i(t) = \sigma(W_{h,x,i}[\vec{h}_i(t-1), \vec{x}_i(t)] + \vec{b}_i) \tag{1.2}$$

where σ is the activation function, $W_{h,x,i}$ is the joint weight matrix for layer i, and b_i is the bias vector.

By including the previous output term as current state input, the network can keep a "working memory" of recent history data. Figure 1 shows the dataflow in a single neuron RNN layer. Notice that if all weights associated with $\vec{h}_i(t-1)$ are set to zero, the network becomes identical to an FFNN, so an RNN have full capacity of an FFNN plus additional ability to capture temporal correlation.

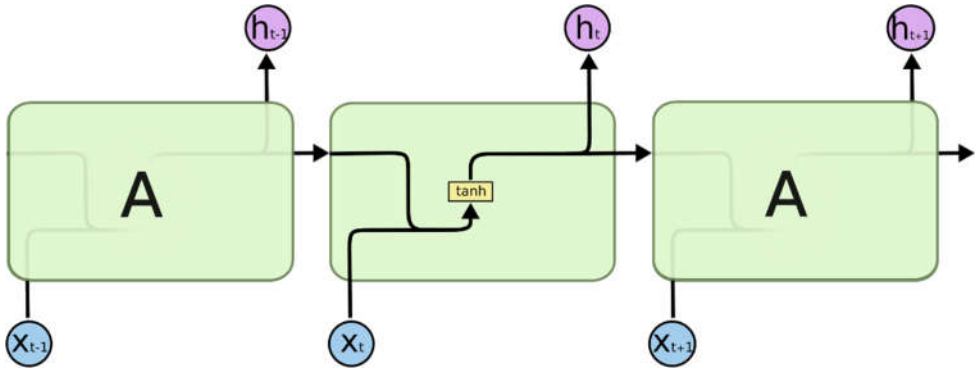

Figure 1 - Dataflow in a single neuron RNN layer [6]

One issue that regular RNN models have is called "vanishing gradient". This is because the error gradient must backpropagate through both layers and through timesteps. The gradient values gradually vanish out as they propagate into earlier timesteps. The most frequently used fix for the vanishing gradient issue is to gate the recurrent hidden state. One commonly used model is called Long Short-Term Memory (LSTM). For LSTM models, each recurrent hidden layer neuron is replaced with a mini-network, called "LSTM cell" [4]. Figure 2 shows the dataflow in a single neuron LSTM layer. The neuron is much more complicated than a RNN neuron shown in Figure 1.

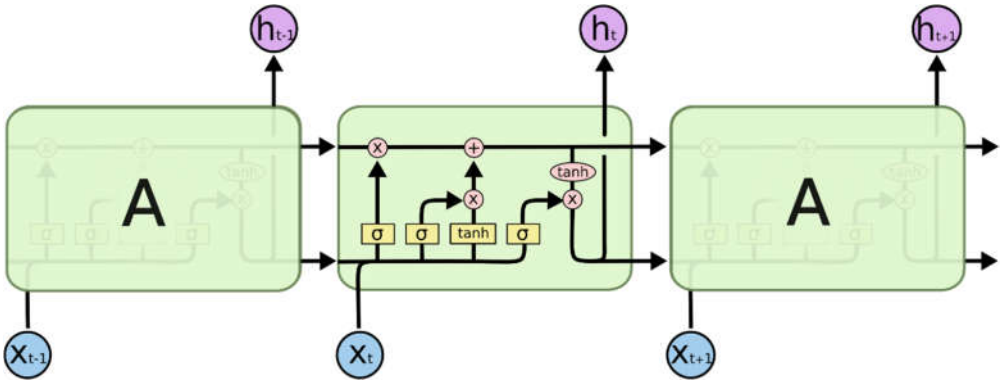

Figure 2 - Dataflow in a single neuron LSTM layer [6]

The following equations describe how the output vector is calculated:

$$\vec{f}(t) = \sigma_f\left(W_f\left[\vec{h}(t-1), \vec{x}(t)\right] + \vec{b}_f\right) \quad (1.3)$$

$$\vec{\imath}(t) = \sigma_i\left(W_i\left[\vec{h}(t-1), \vec{x}(t)\right] + \vec{b}_i\right) \quad (1.4)$$

$$\vec{m}(t) = \sigma_m\left(W_m\left[\vec{h}(t-1), \vec{x}(t)\right] + \vec{b}_m\right) \quad (1.5)$$

$$\vec{o}(t) = \sigma_f\left(W_o\left[\vec{h}(t-1), \vec{x}(t)\right] + \vec{b}_o\right) \quad (1.5)$$

$$\vec{c}(t) = \vec{\imath}(t) \cdot \vec{m}(t) + \vec{c}(t-1) \quad (1.6)$$

$$\vec{h}(t) = \vec{o}(t) \cdot \vec{m}(t) + \sigma_h(\vec{c}(t)) \quad (1.7)$$

We can see that each cell of the LSTM network is a mini-network by itself. The LSTM network prevents gradients from vanishing by introducing a path through the cell states $\vec{c}(t)$ that does not have exponential decaying factor [7].

1.1.2 Convolutional Neural Network (CNN)

Another important component of our proposed network is CNN. Unlike regular feedforward networks, where each neuron in a layer is connected with every neuron from the previous layer (i.e. fully connected), a CNN usually consists of one or more {convolutional layer, pooling layer} pairs. The convolutional layer (usually 2D) uses a bank of N convolutional kernels to convolve with the input data which is at least two dimensional. The output of a convolutional layer will be N matrices. A pooling layer will down sample these matrices using predefined rules (e.g. max-pooling layer with pool size (2,2) will keep the maximum value of each 2×2 submatrix of every input matrix). The purpose of pooling is to prevent overfitting. Compared with the fully connected layer, the convolutional layer has significantly fewer learnable parameters regardless of input size. For example, a bank of five 3×3 kernels only has 45 learnable parameters regardless of the size of the input matrix. Figure 3 shows an example network that uses CNN.

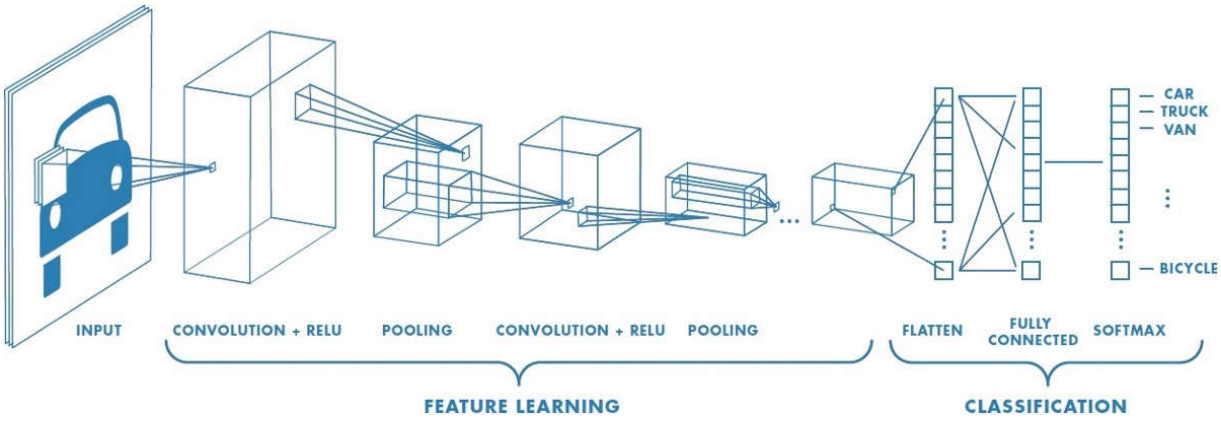

Figure 3 - Example of a network with many convolutional layers. Filters are applied to each image at different resolutions, and the convolutional layers. Filters are applied to each image at different resolutions, and the output of each convolved image is used as the input to the next layer [8].

Because of shared weight used for the moving convolutional kernels, local connectivity instead of full connectivity and 3D volumes of neurons (with height, width and depth), CNN has better generalization on computer vision tasks.

2. Literature Review

Researchers have attempted to construct robust models for rare sound event detection in the past decade. Gerosa et al. [9] proposed a method for choosing the composition of the feature vector and for evaluating dimension mixing filter selection criteria and wrapper validation of results. Their model uses GMM classifier and could adaptively choose the best input feature set from a bank of candidate features. They achieved 95% precision and 92% recall for combined results of screaming and gunshot detection.

Marchi et al. [10] proposed an unsupervised sound novelty detection method using bidirectional LSTM denoising autoencoder. Their approach relies on reconstruction error of the autoencoder which is trained to reconstruct "normal sound". They claimed that their model achieved 93.4% F-1 score. However, this autoencoder approach does not distinguish between different types of rare sound.

Laffitte et al. [11] trained a Deep Belief Network (DBN) and turned it into a Deep Neural Network (DNN). Their research targeted on detecting screaming and shouting from noisy environment in subway, which is a particularly challenging case because the network must be immune to noise. This paper reports confusion matrix of all classes, but it does not report the precision and the recall of each class. One problem with this model is that it has high confusion rate between conversation and shout. Another issue with this research is that its dataset was recorded in a reserved subway train where actors produced each class of sound according to plots. The data may not reflect the subway environment in real life, and it is difficult for other researchers to reproduce their results since the dataset is not an open source.

During DCASE Challenge 2017 [12], Lim et al. [13] won the first place in Task 2 (Detection of rare sound events). They used a 1D convolutional layer to extract the frame-wise acoustic feature from the MFCC spectrogram. The information is further processed by two stacked backward LSTM layers for more accurate onset time estimation. A noteworthy part of this model is its internal-external ensemble method. The internal ensemble is realized by averaging the probabilities at the same timestep given by a sliding LSTM window with a hop size of one frame. The external ensemble is achieved by majority voting by multiple individually trained networks. They achieved a high average F-1 score of 93.1% for all three target classes (gunshot, baby crying and glass break).

Despite different models, the above researchers have their own focus and advantages. One blank field is detecting multiple classes of event using a single model based on neural networks. In this research, we try to build an LSTM-RNN based ensemble model using a novel hybrid structure.

3. Description of Research Results

3.1 Proposed Hybrid Structure

The structure we propose is a variation of CRNN (Convolutional RNN) which adds an auxiliary input branch in parallel with the "CNN-LSTM" branch. Our system contains multiple networks, and each is trained to recognize one type of sound event. Figure 4 shows the case where the number of classes is 2.

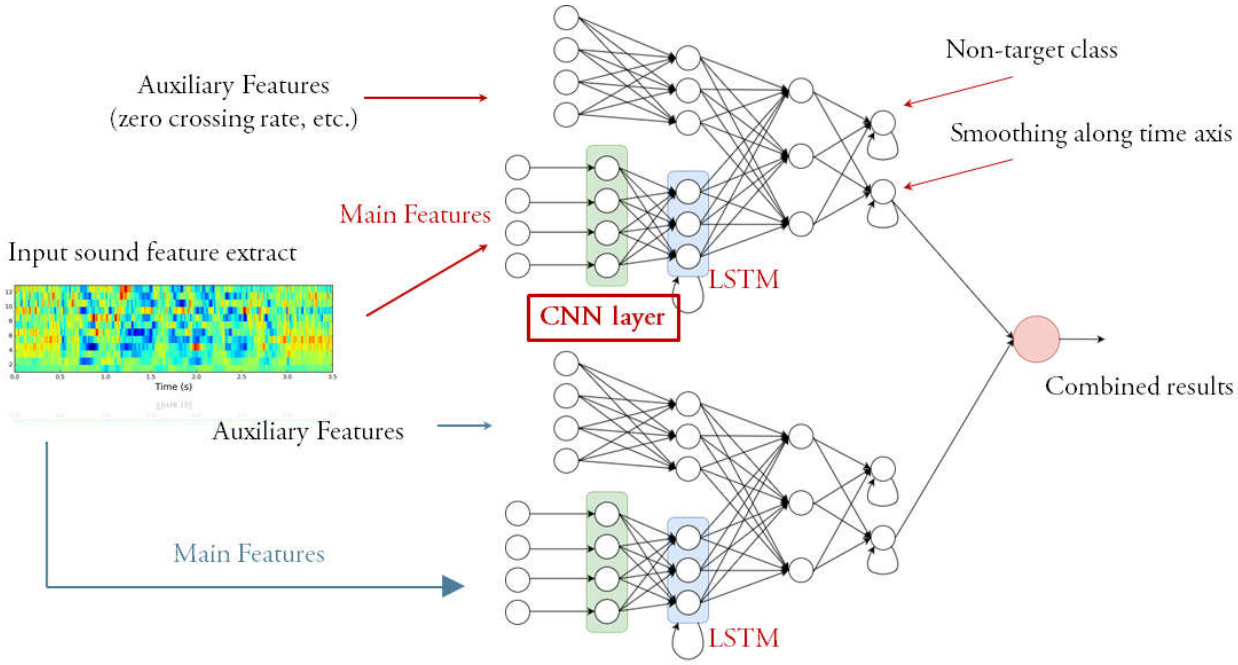

Figure 4 - Example of a proposed network where the number of classes is 2

3.1.1 Multi-input Layers

In order to let the neural network learn different features, we divide the input into different sets. For each training sample, we send the log Mel-energy spectrum of the sound into the CNN-LSTM branch because we want the network to learn the temporal correlation between frames that belong to the target class. For some features, such as zero crossing rate and first-order difference of log Mel-energy (delta feature), the temporal correlation between target frames is much weaker. Hence, we only send them through a feedforward net, thereby reducing the number of parameters needed by the network and speeding up the training and inference process.

The log Mel-energy spectrum is obtained by taking the Short Time Fourier Transform with a window size of 2048 and overlapping of 50% (resulting in 46 ms per frame), calculating the power of the spectrum and

multiply the Fourier Spectrum with 64 Mel-frequency filter banks. Finally, the log scale value is taken because it is similar to how human ear perceives sound [14].

3.1.2 2D-CNN Layer

Lim et al. [13] used a 1D convolutional layer to extract frame-wise information, and they claim that this would give better accuracy in terms of onset time evaluation. However, the length of each frame is only 46 ms, which is much shorter than the collar (0.5 s) for valid onset evaluation allowed by DCASE challenge. Moreover, a 1D CNN only considers features within one frame, which is less immune to random noise. Therefore, we use a 2D CNN layer with 64 filters, each with kernel width and height equal to (2,20). The time length for consecutive two frames is only 69 ms, which is also much less than 0.5 s, so the onset time estimation will not get much impact. By using 2D CNN, the network is more immune to random noise because information in two consecutive frames is considered.

A max pooling layer with the kernel size of (1,64) is used to only keep one representative feature per filter for each timestep. After that, batch normalization and dropout with 0.3 dropout rate is applied to the result to prevent overfitting.

3.1.3 LSTM Layer

In order to learn the temporal correlation between frames, a LSTM-RNN layer with truncated backpropagation length equal to 100 is stacked on top of the CNN layer. Like the CNN layer, dropout with dropout rate equal to 0.3 is also applied to the output of LSTM layer.

3.1.4 Auxiliary Branch

Beside the log Mel-energy, there are other useful features that are helpful for increasing the classification accuracy, such as zero crossing rate [9] and delta features. Unlike the log Mel-energy, these features will only pass through the feedforward layer either because the temporal correlation between target frames is much weaker or the time difference (i.e. differential change) of these features are more important. Dropout with 0.3 dropout rate is also applied to the output of the feedforward layer.

3.1.5 Recurrent Output Layer

The most frequently used output layer is a dense (fully connected) feedforward layer. However, for sound event detection, if one frame is the target frame, the probability that the next frame is also a target frame is higher than the cases where the current frame is a non-target (background noise) frame. Therefore, we use a recurrent output unit to smooth the output probability along the time axis.

3.1.6 Postprocessing

After we get the output from the output neuron, we apply thresholding on it. The threshold is twofold: (1) the output must be greater than a threshold value, which is empirically set to 0.5, for it to be considered as "positive"; (2) the output must be continuously above the threshold for a number of frames, which is empirically set to 5 for baby crying and 2 for glass break as well as gunshot, for it to be considered as "valid". Similarly, for an event to be considered as "negative" and "valid", the output must be lower than the threshold for several continuous frames. Figure 5 shows the spectrum of an example input audio clip and the corresponding label of that clip. Figure 6 shows the corresponding output given by the network.

After all the output are obtained from all networks, the final output will be the class with highest probability (which must also be "positive").

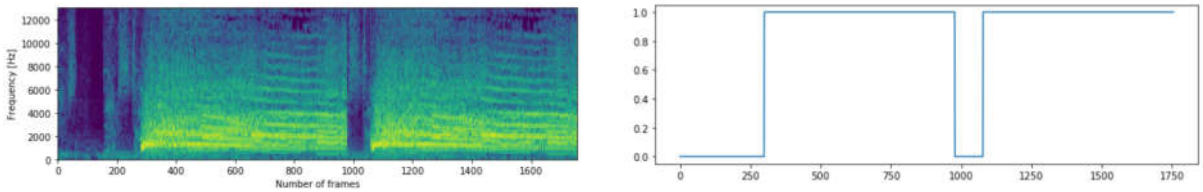

Figure 5 - Example of input sound spectrogram and label (1=positive, 0=negative)

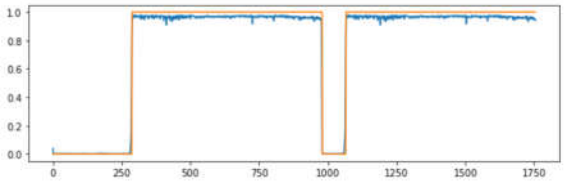

Figure 6 – Blue line is the probability given by the network, orange line is the output label

3.2 Dataset

In order to be able to compare the performance with previous works, we downloaded the dataset from "DCASE Challenge 2017 Task 2, detection of rare sound events" [15]. The pre-mixed dataset contains 1500 sound clips for training and 1500 clips for testing. Each class of sound has 500 clips for training and 500 for testing. Each sound clip is 30 seconds long. All sound clips are autogenerated by mixing a clean target sound file with noisy background sound clips which includes different scenes such as an office room and streets. The target sound is mixed with the background sound with {-6,0,6} dB ratio, each with 1/3 probability.

For improving the robustness of our model, we use the provided code to conduct data augmentation. The provided code allows us to randomly mix zero or one target event sound clip with a randomly chosen

background sound file. We generated 10000 sound clips for each event class. Each clip has the same length as the provided pre-mixed clips and contains target event with probability of 0.9. We increased the probability of target event sound because the event only last for as long as 4s in a 30s sound clip.

3.3 Evaluation Metric

For the sake of comparison, we used the same event-wise metric as the DCASE challenge [16]. After we compute the output using the model, an event list is generated in the following format:

$$Onset\_time \quad Offset\_time \quad Event\_category$$

A list is generated for both the prediction and the provided ground truth labels. The provided code [16] will compare the two list files and calculate evaluation results based on the following rules:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \qquad F_1 = \frac{2P * R}{P + R} \tag{3.1}$$

where

$$TP = True\ Positive \quad FP = False\ Positive \quad FN = False\ Negative\ (miss)$$

$$P = Precision\ Rate \quad R = Recall\ Rate \quad F_1 = F\text{-}1\ score$$

A predicted event is considered "true positive" if (1) there is an event in the ground truth event list that overlaps (or partially overlaps) with the predicted event and (2) if the predicted onset time is within $\pm$0.5 s of the actual onset time.

A predicted event is considered "false positive" if there is no event in the ground truth event list that overlaps (or partially overlaps) with the predicted event.

The predicted result is considered having "false negative" if there is an event in the ground truth event list that overlaps (or partially overlaps) with none of predicted event. That is, a target event is failed to be detected by the model.

3.4 Results

The initial network does not have the 2D CNN layer. We experimented with different sets of parameters including number of filter bank for input, width and height of convolution kernel, dropout rate and learning rate decay. The comparison between the baseline results and our initial results is shown in Table 1.

Table 1: F-1 score for all three classes of events with the initial structure

Class	Training Set F-1 (baseline)	Training Set F-1 (Our model)	Testing Set F-1 (baseline)	Testing Set F-1 (Our model)
Glass break	88.5%	82.0%	79.1%	80.0%
Baby crying	72.0%	71.9%	66.8%	66.8%
Gunshot	57.4%	59.2%	46.5%	52.1%

We observed that the F-1 score rises fast at the beginning but slows down quickly at around 10 epochs. After comparing our architecture with the previous works, we found that our model might not be good enough at extracting useful features from input. Therefore, we modified our network structure by adding the 2D CNN layer mentioned in section 3.1.2. After retraining, the performance of all three classes has improved. The comparison between the baseline results and our new results is shown in Table 2.

Table 2: F-1 score for all three classes of events with the modified structure

Class	Training Set F-1 (baseline)	Training Set F-1 (Our model)	Testing Set F-1 (baseline)	Testing Set F-1 (Our model)
Glass break	88.5%	87.7%	79.1%	84.1%
Baby crying	72.0%	75.2%	66.8%	73.8%
Gunshot	57.4%	67.0%	46.5%	63.2%

After we added the CNN layer, we observed noticeable improvement on our F-1 score on all three classes.

Because postprocessing is claimed to have significant contribution to improving the F-1 score for sound event detection, we recreated the model described in the technical report by Lim et al. [13] and implemented the internal-external ensemble method described in their report to analyze the contribution of this method. Additionally, we have an observation about the baby crying class. Unlike glass-breaking and gunshot, for human sounds like baby crying, although the onset might be clear, the offset is usually not. This is because the crying sound would usually last several seconds (e.g. 2 s) and it usually contains some brief pauses (e.g. 0.5 s) due to breathing. In the ground truth labels provided by DCASE, if the baby crying lasts for 5 seconds and consists of multiple distinguishable cries (e.g. 3 cries with pauses between each cry), the given label only shows one event, but our model would count them as three distinct events. According to the metric specified by DCASE, only one event will be considered true positive and the other two will be counted as false positives because their onset times are a few seconds later than the onset time given by the ground truth labels. Therefore, our precision score will decrease. Our strategy to solve this problem is to set different onset and offset threshold time. For example, we consider an event as valid positive if the output of the model is higher than 0.5 value for 13 continuous frames, but we require the output of the model to be lower than 0.5 for 28 continuous frames for the event to be considered as ended. Table 3 shows the F-1 score of "baby crying" class alone using different postprocessing methods.

Table 3: F-1 score for baby crying class on Testing dataset without different postprocessing

Class	(A) DCASE Baseline	(B) No Ensemble	(C) Internal Ensemble Only	(D) External Ensemble Only	(C)+(D)	(C)+(D)+Different Onset/Offset Time Threshold
Baby crying	66.8%	70.1%	76.3%	72.5%	78.2%	**81.8%**

We observed a significant increase in F-1 score when both internal ensemble method and external ensemble method are applied.

4. Conclusion and Outlook

In this research, we studied different strategies for automatic sound event detection. We proposed a LSTM-RNN based hybrid network. The unique feature of our model is that we included an auxiliary branch to treat different inputs sets with different processing techniques. We are also able to classify three types of sound events using an ensemble structure. Additionally, we conducted experiments on various postprocessing techniques and we found them helpful for improving the F-1 score. In order to further improve the performance of our model, we think there are three things we could do as our next step: (1) we believe that, in order to achieve high F-1 scores, one need to train potentially tens of models, each fine-tuned with augmented data and various other techniques that are yet to be experimented with, so that one thing we could do next (2) we could augment the training dataset to a larger size to improve the robustness of our model. (3) we could train some models with different timestep length to achieve better performance in event duration estimation as well as precise onset estimation. With these methods implemented and tested, we will attempt to integrate the methods that contribute the most to improving the F-1 score with our own model to enhance the robustness of our model.

References

[1] B. C. Csáji, "Approximation with Artificial Neural Networks;," Faculty of Sciences; Eötvös Loránd University, Hungary, 2001.

[2] Y. Geng et al. "A novel image tag completion method based on convolutional neural network," arXiv:1703.00586v2 , 2017.

[3] A. Zeyer et al. "A Comprehensive Study of Deep Bidirectional LSTM RNNs for Acoustic Modeling in Speech Recognition," arXiv:1606.06871v2 , 2017.

[4] J.Schmidhuber et al. "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735–1780, 1997.

[5] K. Piczak, "Environmental sound classification with convolutional neural networks," *Machine Learning for Signal Processing,* IEEE 25th International Workshop on., pp. 1–6, 2015.

[6] C. Olah, "Understanding LSTM Networks," 27 Aug 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 2 April 2018].

[7] Pascanu, Razvan et al. "On the difficulty of training recurrent neural networks," *ICML,* pp. 1310-1318, 28 3 2018.

[8] MathWorks, "Convolutional Neural Network," [Online]. Available: https://www.mathworks.com/discovery/convolutional-neural-network.html. [Accessed 20 04 2018].

[9] L. Gerosa, G. Valenzise and M. Tagliasacchi, "Scream and gunshot detection in noisy environments," *Signal Processing Conference,* no. 978-839-2134-04-6, 2007.

[10] E. Marchi et al. "A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks," *Acoustics Speech and Signal Processing (ICASSP) 2015 IEEE International Conference on,* pp. 1996-2000, 2015.

[11] P. Laffitte et al. "Deep neural networks for automatic detection of screams and shouted speech in subway trains," *Acoustics, Speech and Signal Processing (ICASSP),* 2016 IEEE International Conference on .

[12] "Detection and Classification of Acoustic Scenes and Events, An IEEE AASP Challenge," Mar to July 2017. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2017/index.

[13] H. Lim et al. "Rare Sound Event Detection Using 1D Convolutional Recurrent Neural Networks," 2017.

[14] S. Ravindran et al. "Speech recognition using filter-bank features".

[15] "DCASE 2017 Detection of rare sound events," [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-rare-sound-event-detection.

[16] A. M. Toni Heittola, "Evaluation toolbox for Sound Event Detection," [Online]. Available: https://github.com/TUT-ARG/sed_eval. [Accessed 20 04 2018].