Sound Event Detection

Yuan Feng, 521030910369
Shanghai JiaoTong University

1. Introduction

Our everyday environment consists of many sound sources that create a complex mixture audio signal. Human auditory perception is highly specialized in segregating the sound sources and directing attention to the sound source of interest. This phenomenon is called the cocktail party effect.

The goal of automatic sound event detection (SED) [1] methods is to recognize what is happening in an audio signal and when it is happening. In practice, the goal is to recognize at what temporal instances different sounds are active within an audio signal. This task is illustrated in Figure 1.

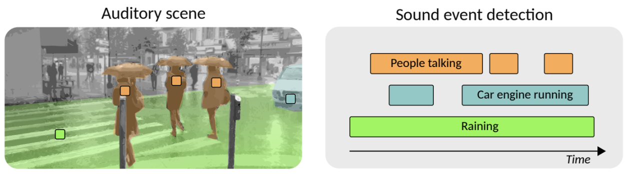

Figure 1. Sound Event Detection

The dataset we use is the DCASE18 (Detection and Classification of Acoustic Scenes and Events) dataset, which is weakly labelled. That is, there are only audio tags for each audio clip without the onset and offset times of sound events. This is one of the challenges facing the SED problem. [2]

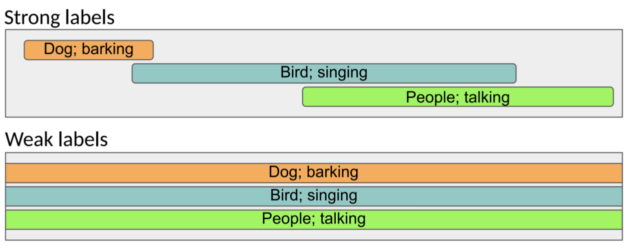

Figure 2. Strong Label vs. Weak Label

2. Experiment Tasks Overview

In this project, I primarily conducted the following experiments:

1. I understand code logic, being familiar with sound event detection models.
2. I understand the challenges of time axis prediction under weak supervision, as well as the principles behind baseline model design.
3. I implement the CRNN model according to the model framework, and make model and parameter modifications to achieve better performance.
4. I try some audio data augmentation methods, analyze and compare the results, and try to achieve better performance.
5. Provide the optimal model parameter configuration in the end.

3. CRNN

The receptive field of CNNs is limited, and therefore they cannot capture long-term dependencies in audio clips. However, some sound events have long-term dependencies. For example, some honking sounds may last for tens or even dozens of seconds, and long-term information is very important for SED.

Therefore, we need to introduce recurrent neural networks (RNNs), which are neural networks that can store historical information in hidden states, and thus capture the long-term dependencies of sequential data. However, the potential problem with conventional RNNs is that the weight gradients may vanish or explode during the training process.

Therefore, there is also long short-term memory (LSTM), which introduces constant error carousel units, input gates, output gates, and forget gates to avoid the problem of gradient explosion and vanishing. Gated recurrent units (GRUs) are an improved LSTM architecture that can reduce the parameters of LSTM and simplify the gating mechanism.

In this project, we applied bidirectional GRU (biGRU), which can work in both directions.

The baseline model of this project is the **CRNN** architecture. It performs several convolution and max-pooling operations on the input three-dimensional tensor (the three dimensions are $batch\_size$, $time_steps$, and num_freq), and then through the BiGRU layer, uses the fully connected layer to transform it into a three-dimensional tensor corresponding to ($batch\_size$, $time\_steps$, and $num\_classes$), to represent the probability of the occurrence of sound events over time.

Meanwhile, the classification task loss function used is $BCELoss$. The formula is as follows:

$$\mathcal{L}(y, \hat{y}) = -\hat{y}\log(y) + (1 - \hat{y})\log(1 - y)$$

The specific network architecture is shown in the Figure 3.

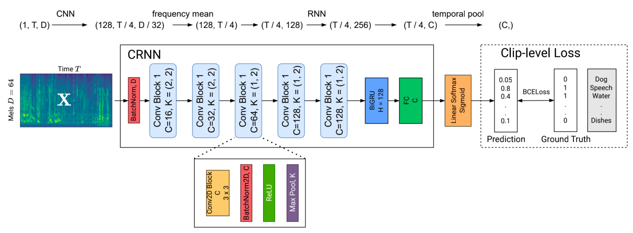

Figure 3. CRNN

4. Metrics

We use three different evaluation methods to evaluate the **F1-score** of a classification result.

1. Tagging-Based: This represents whether the audio event was correctly detected, i.e., whether the label was correctly applied and the classification was correct.

2. Segment-Based: This represents whether the location of the detected audio event is correct, with the criterion for correct location being that it is divided into individual segments, and each segment is considered separately.

3. Event-Based: This also represents whether the location of the detected audio event is correct, but unlike the Segment-Based method, it is only considered a correct detection if the entire event is detected, and the start and end positions are within a certain offset range.

The difference between the Segment-Based and Event-Based methods can be seen in Figures 4 and 5.

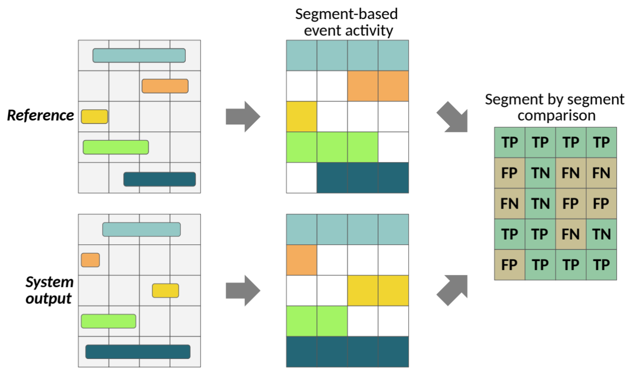

Figure 4. Segment-Based

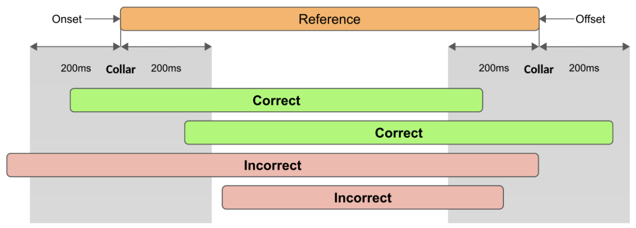

Figure 5. Event-Based

5. Experiments

5.1. Baseline

Results:

Table 1. Baseline Results

	f_measure	**precision**	**recall**
event_based	0.140607	0.146566	0.15417
segment_based	0.607592	0.641032	0.591055
tagging_based	0.666189	0.723808	0.634672
mAP	0.7049425304376002		

5.2. Data Augmentation

I have tried five data augmentation methods.

1. Add Noise (AN): Randomly adding white noise to the audio, with the specific implementation being adding a Gaussian noise with a 0.6 probability.

2. Time Shift (TS): Randomly shifting the audio forward or backward (for example, shifting forward by T frames, the last T frames of the audio will be set to 0, with no sound).

3. Freq Shift (FS): Applying the aforementioned time-domain shifting method to the frequency domain.

4. Time Mask (TM): Randomly masking a segment of the audio, blocking it out.

5. Freq Mask (FM): Applying the aforementioned time-domain shifting method to the frequency domain.

Other methods include changing the pitch, changing the audio speed, and stretching the audio duration, which have not been implemented here.

The comparative experimental results of the above five methods are shown in the Table 2:

Table 2. Data Augmentaion Results

	None	AN	TS	FS	TM	FM
mAP	0.70	**0.73**	0.69	0.71	0.70	0.69
Eve_F	0.14	0.15	0.12	0.14	0.13	**0.16**
Eve_P	0.15	0.15	0.13	0.13	0.13	**0.16**
Eve_R	0.15	0.15	0.14	0.16	0.14	**0.17**
Seg_F	<u>0.61</u>	<u>0.61</u>	0.59	<u>0.61</u>	<u>0.61</u>	0.60
Seg_P	0.64	**0.66**	0.63	0.63	0.64	0.64
Seg_R	0.59	0.57	0.57	**0.60**	0.58	0.58
Tag_F	<u>0.67</u>	<u>0.67</u>	0.63	0.66	0.65	0.65
Tag_P	<u>0.72</u>	<u>0.72</u>	0.70	0.71	<u>0.72</u>	0.71
Tag_R	0.63	**0.64**	0.59	0.63	0.61	0.61

For each metric, the values achieved by the method with the highest performance are **bolded**, and in some cases, there are multiple methods with the highest values, which are <u>underlined</u>.

It can be observed that some data augmentation methods can improve the model's performance, and the methods that can effectively enhance the performance are mainly those that enhance the frequency domain and add noise.

5.3. Different GRU Dimensions

I tried three different dimensions for the biGRU layer: 64, 128, and 256. The comparative experiment results are shown in Table 3:

Table 3. Different GRU Dimensions Results

	64	128	256
mAP	0.685	**0.705**	0.694
Eve_F	0.101	0.140	**0.144**
Eve_P	0.112	0.147	**0.149**
Eve_R	0.106	0.154	**0.162**
Seg_F	0.579	**0.608**	0.599
Seg_P	**0.674**	0.641	0.610
Seg_R	0.518	0.591	**0.603**
Tag_F	0.654	**0.666**	0.657
Tag_P	<u>0.724</u>	<u>0.724</u>	0.692
Tag_R	0.604	0.635	**0.636**

The result annotation method is the same as in Table 2. Here, the GRU dimension of 128 is the baseline. It can be observed that changing the GRU dimension does not improve all metrics, but overall, when the GRU dimension is 256, there is an improvement in many of the metrics.

6. Best Result

Based on the above analysis, I have chosen to use the **Add Noise** data augmentation method and set the GRU dimension to **256** as the optimal solution. The parameter configuration file can be found at `config/best.yaml`.

Results are shown in Table 4

Table 4. Best Results

	f_measure	**precision**	**recall**
event_based	0.1531	0.159015	0.164004
segment_based	0.614742	0.636503	0.600154
tagging_based	0.674325	0.708152	0.650378
mAP	0.7080823432171583		

This result does not achieve the best performance in all metrics, but most of the metrics show improvement. In real-world problems, the algorithm selection should be based on the specific metrics that need to be improved.

References

[1] Annamaria Mesaros, Toni Heittola, Tuomas Virtanen, and Mark D. Plumbley. Sound event detection: A tutorial. *IEEE Signal Processing Magazine*, 38(5):67–83, September 2021. 1

[2] Qiuqiang Kong, Yong Xu, Wenwu Wang, and Mark D. Plumbley. Sound event detection of weakly labelled data with cnn-transformer and automatic threshold optimization, 2020. 1