Shanghai Jiao Tong University
AI3611

Sound Event Detection

Yan Han (519030910404)

June 12, 2022

1 Introduction

For humans, it is common to recognize and analyse the sound around. However, for machines, it is not easy. The goal of automatic **sound event detection** (SED) 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 [2].

Section 2 describes the general SED model and the challenges in weak supervision. Section 3 shows some experiments about adjusting hyper parameters in detail. Section 4 investigates ways of data augmentation. Section 5 research on methods of post processing. And 6 draws a conclusion. The source code is available at `https://github.com/wolfball/Intellisense-Cognitive-Practice`.

2 Model and Challenge Description

In 2.1, we will introduce the general design of a sound event detection model and describe one network as baseline. In 2.2 , we will briefly discuss the challenges in the prediction of the span of the sound under weak supervision.

2.1 Baseline Model

[2] shows a pipeline of SED (See Figure 1). In the learning state, we use a feature extractor to get a feature matrix that represents the input audio, and encodes the labels to produce the target outputs. Then, by supervise learning, we can train an Acoustic model that can detect the sound events in an audio clip in the test stage.

Here we choose CRNN [3] as our baseline acoustic model (See Figure 2). The input are pre-processed feature matrix $X \in \mathbb{R}^{T\times D}$ which contains time-frequency information. The first layer in CRNN is a batch normalization. Then, there are three blocks, where each consists of a convolution layer, batch normalization, ReLU activation and maxpooling layer. Next, for extracting temporal information, the data will be processed by a bidirectional GRU followed by a fully-connected layer and a Sigmoid layer that produce the raw outputs

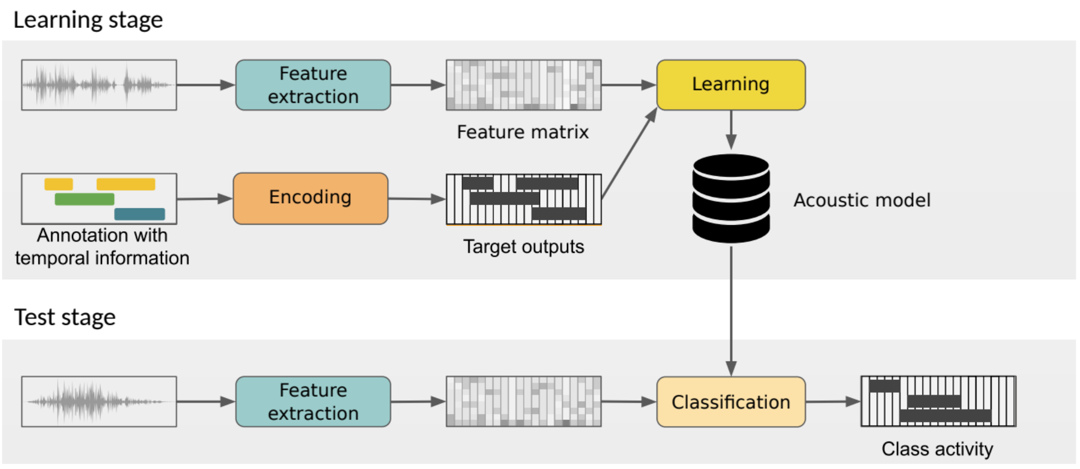


Figure 1: Pipeline of SED.

$y'_t(e) \in [0,1], t \in \{1,\ldots,T'\}, e \in \{1,\ldots,E\}$ where T' is the size in time domain (after pooling) and E is the number of classification. To recover the original temporal information, we apply interpolation to get $y_t(e), t \in \{1,\ldots,T\}$. Moreover, to get a clip-level prediction, we exclusively utilize linear softmax (LinSoft):

$$y(e) = \frac{\sum_{t=1}^{T} y_t(e)^2}{\sum_{t=1}^{T} y_t(e)} \tag{1}$$

Note that the usage of maxpooling and interpolating can smooth the results and keep the continuity in time domain.

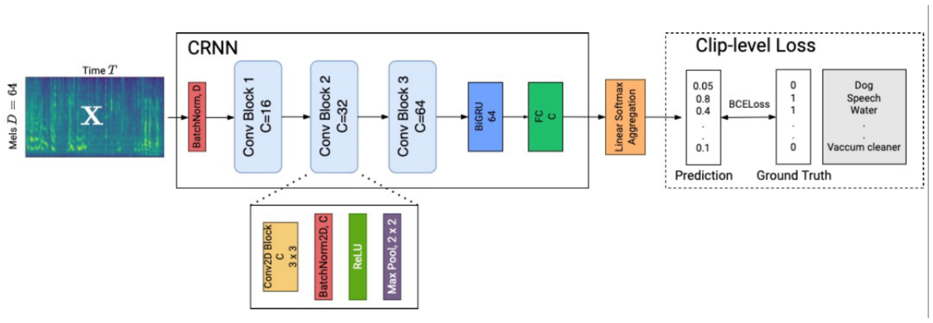


Figure 2: Baseline model.

2.2 Challenges under Weak Supervision

In general, there are three types of data(See Figure 3 for the first two types): **data with strong labels**, has the classification of sound events with their start and end times; **data with weak labels**, has only the classification of sound events; **data without labels**.

The main challenge in weak supervision is the absence of the corresponding onset and offset times of the audio events, which is difficult for models to localize the events due to the loss of ground truth. In this report, we mainly focus on this **Weak-Supervised Sound Event Detection**(WSSED).

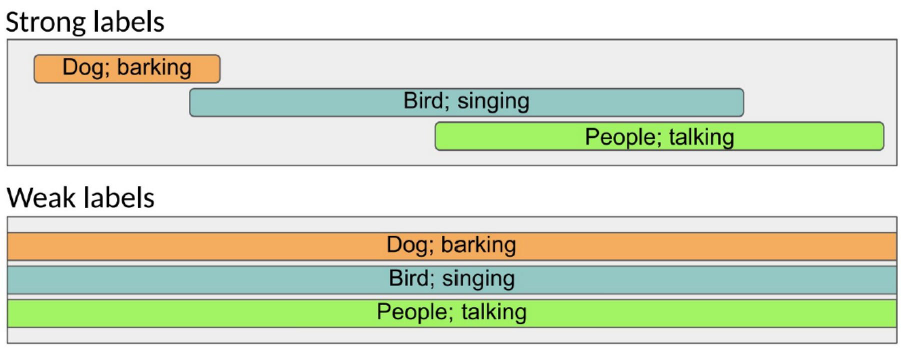


Figure 3: Strong labels and weak labels.

3 Experiments and results

In this section, we will do some experiments with different hyper-parameters. We use Balanced Audioset as our dataset, which contains ten events (e.g. Cat, Speech). Due to the low memory footprint and excellent performance, we use log-Mel power spectrograms (LMS) as the feature of the audio clip with $D = 64$. The model with its parameters is shown in Figure 2. Note that 64D biGRU feature is actually 128D because of two directions. For the output $y(e)$, we use binary cross entropy (BCE) as the loss function. A batch size of 32 is utilized for all experiments. During training, zero padding to the longest sample-length within a batch is applied. For optimizer, training uses AdamW with learning rate 0.001, and the learning rate will be reduced as long as the performance of the model has no longer improved within 10 epochs. For smoothing outputs, evaluation uses median filtering (See 5 for more post-processing methods) to process $y_t(e)$ with window size 1 and threshold 0.5. All experiments are executed with 100 epochs.

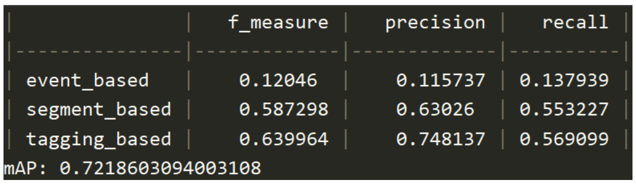

```
|                | f_measure  |  precision  |  recall  |
|----------------|------------|-------------|----------|
| event_based    |  0.12046   |   0.115737  | 0.137939 |
| segment_based  |  0.587298  |   0.63026   | 0.553227 |
| tagging_based  |  0.639964  |   0.748137  | 0.569099 |
mAP: 0.7218603094003108
```

Figure 4: Evaluation result of baseline.

For evaluation, we will introduce three types of metrics as shown in Figure 4:

1. Event based: measures on- and off-set overlap between prediction and ground truth and describes a model's capability to estimate a duration;

2. Segment based: gives an objective measure of a given model's sound localization capability by the segment-level overlap between ground truth and prediction. This can be seen as a coarse localization metric since precise timestamps are not required;

3. Tagging based: measures the models' capability to correctly identify the presence of an event within an audio clip.

Number of blocks. We change the number of the block from two to five (See Table 1). The mAP increases with more blocks. 4-block CRNN scores the best in event based metrics while less or more blocks will reduce the performance. In segment based metrics, when increasing blocks, F score and precision increases, but recall decreases. In tagging based metrics, model with more blocks generally performs better. This may due to the better capability to extract features.

Feature dimension of BiGRU. We try 32, 64 and 128 dimension of BiGRU feature (See Table 1). The higher dimension the BiGRU feature is, the higher F scores for all three types of metrics will be. This may due to the higher capability of representing.

Table 1: Experimental results. Model with (*) represents the baseline. mAP represents the mean of the average precision in clip-level. "Eve-", "Seg-", "Tag-", "F", "P" and "R" mean event based, segment based, tagging based, F measure, precision, and recall, respectively. Baseline has 3 Blocks, 64D biGRU feature.

model	mAP	Eve-F	Eve-P	Eve-R	Seg-F	Seg-P	Seg-R	Tag-F	Tag-P	Tag-R
2-Blocks	0.689	0.065	0.054	0.104	0.571	0.584	**0.574**	0.619	0.713	0.563
3-Blocks*	0.723	0.116	0.111	0.132	0.587	0.634	0.549	0.639	**0.751**	0.568
4-Blocks	0.728	**0.144**	**0.156**	**0.146**	0.589	0.676	0.537	0.662	0.741	0.611
5-Blocks	**0.743**	0.124	0.137	0.130	**0.596**	**0.692**	0.534	**0.688**	0.750	**0.643**
BiGRU 32D	0.698	0.131	**0.125**	0.148	0.581	**0.645**	0.535	0.627	0.745	0.561
BIGRU 64D*	**0.723**	0.116	0.111	0.132	0.587	0.634	0.549	0.639	**0.751**	0.568
BiGRU 128D	0.721	**0.132**	0.122	**0.166**	**0.601**	0.612	**0.605**	**0.654**	0.739	**0.600**

4 Data Augmentation

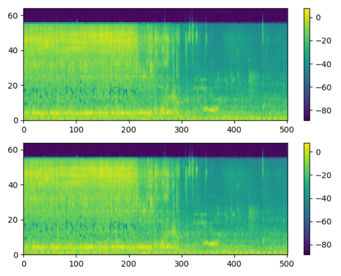
In this section, we will investigate and survey methods of data augmentation in audio learning. In general, there are many ways to do data augmentation, such as time stretching, pitch shifting, dynamic range compression, sub-frame time shifting, block-mixing and so on. Similar to [1], we mainly try the following five data augmentation methods (See Figure 5 for visualization).

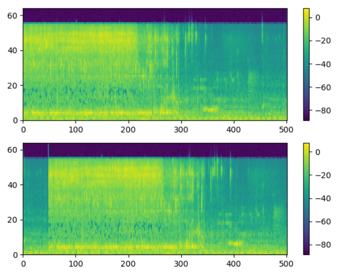
Table 2: Experimental results with different data augmentations (DA). mAP represents the mean of the average precision in clip-level. "Eve-", "Seg-", "Tag-", "F", "P" and "R" mean event based, segment based, tagging based, F measure, precision, and recall, respectively. All models have 4 Blocks, 64D biGRU feature.

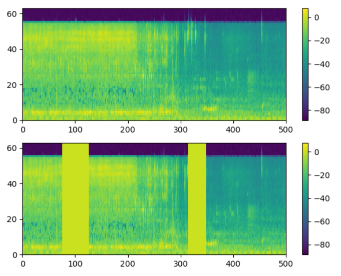
DA	mAP	Eve-F	Eve-P	Eve-R	Seg-F	Seg-P	Seg-R	Tag-F	Tag-P	Tag-R
No DA	0.728	0.144	0.156	0.146	0.589	**0.676**	0.537	0.662	0.741	0.611
NA	0.726	0.148	0.161	0.153	0.578	0.671	0.520	0.655	0.742	0.601
TS	**0.748**	0.142	0.152	0.149	**0.610**	0.665	0.571	0.674	0.735	0.633
TM	0.733	**0.166**	**0.166**	**0.185**	0.591	0.653	0.550	0.653	0.718	0.615
FS	0.728	0.144	0.136	0.169	0.609	0.629	**0.606**	**0.678**	0.725	**0.644**
FM	0.720	0.145	0.144	0.163	0.606	0.645	0.585	0.664	**0.745**	0.608

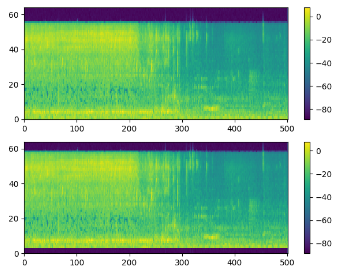
1. Noise Augmentation (NA): For every element in audio feature matrix $X \in \mathbb{R}^{T\times D}$, we add noise $x_n \sim \mathcal{N}(0, 1e-5)$ to it with probability of 0.5.

2. Time Shift Augmentation (TS): First, we choose a shift step $s \sim int(\mathcal{N}(0, 50))$. Then, we roll the feature matrix in the time domain.


(a) Noise augmentation


(b) Time shift augmentation


(c) Time mask augmentation


(d) Freq shift augmentation

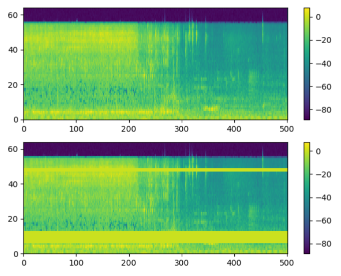

(e) Freq mask augmentation

Figure 5: Visualization of data augmentations. The top one is the original feature, and the bottom one is the transformed feature. Each color represents a type of sound event.

3. Time Mask Augmentation (TM): First, we randomly choose a integer mask duration d from [0,60). Then we randomly choose a start timestep ts and set feature value to zero within $[ts, ts + p)$. This operation will be execute for twice.

4. Freq Shift Augmentation (FS): First, we choose a shift step $s \sim int(\mathcal{N}(0, 4))$. Then, we roll the feature matrix in the frequency domain.

5. Freq Mask Augmentation (FM): First, we randomly choose a integer mask duration d from [0,8). Then we randomly choose a start timestep ts and set feature value to zero within $[ts, ts + p)$. This operation will be execute for twice.

The experimental comparisons are shown in Table2. In general, data augmentation for time domain will get higher scores in event-based metrics for the improvement of the robustness of temporal information. And data augmentation for frequency domain will contribute to higher scores in tag-based metrics for the improvement of the robustness of frequency information. In other words, data augmentation for time domain betters the capability of localizing while for frequency domain improves the ability of detecting.

5 Post Process

Faced with WSSED, we can utilize post-processing to help smooth the outputs. Similar to [1], here we mainly introduce three methods of post-processing. See Table 3 for experimental results. The usage of post-processing contributes to higher score in event-based metrics, but it will also remove some potential events (See Figure 6 for visualizations).

Median First, we binary the prediction with threshold 0.5. Then, we apply median filter with a sliding window sized 11.

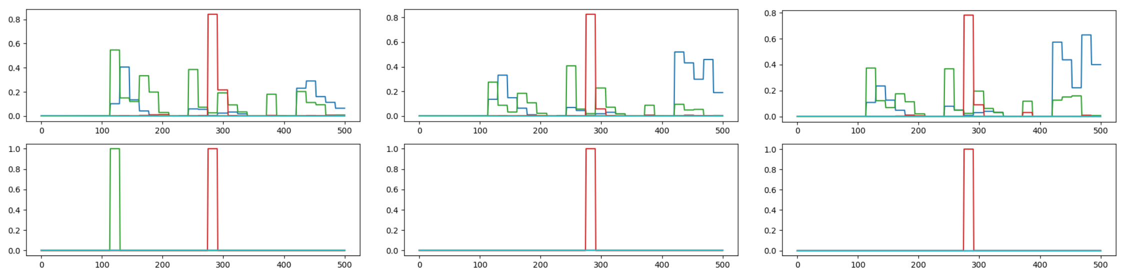

(a) Median (b) Double threshold (c) Triple threshold

Figure 6: Visualization of post-processing. The top one is the original output, and the bottom one is the processed result.

Table 3: Experimental results with different post-processing methods. mAP represents the mean of the average precision in clip-level. "Eve-", "Seg-", "Tag-", "F", "P" and "R" mean event based, segment based, tagging based, F measure, precision, and recall, respectively. All models have 4 Blocks, 64D biGRU feature.

method	mAP	Eve-F	Eve-P	Eve-R	Seg-F	Seg-P	Seg-R	Tag-F	Tag-P	Tag-R
baseline	0.728	0.144	0.156	0.146	**0.589**	0.676	**0.537**	0.662	0.741	0.611
median	**0.733**	0.149	0.162	0.152	0.589	0.683	0.533	**0.666**	**0.747**	**0.618**
double	0.732	**0.211**	0.260	**0.192**	0.580	0.709	0.509	0.657	0.740	0.609
triple	0.506	0.211	**0.265**	0.187	0.570	**0.712**	0.493	0.657	0.741	0.604

Double Threshold This method is defined via two thresholds $\phi_l = 0.2, \phi_h = 0.75$. The algorithm first sweeps over an output probability sequence and marks all values larger than ϕ_h as being valid predictions. Then it enlarges the marked frames by searching for all adjacent, continuous predictions (clusters) being larger than ϕ_l. An n arbitrary point in time t belongs to a cluster between two time steps t_1, t_2 if the following holds:

$$cluster(t) = \begin{cases} 1, & \exists(t_1, t_2) s.t. t_1 \leq t \leq t_2 \\ & \text{and } \forall t_0 \in [t_1, t_2], \phi_l \leq y_{t_0} \leq \phi_h \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

So, the double threshold algorithm $(\mathrm{dt}(\cdot)) is:$

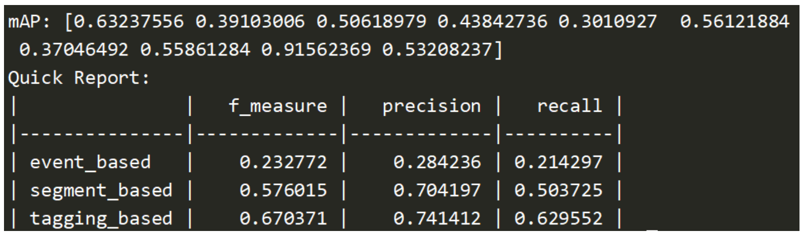
This post-processing is less susceptible to duration variations. However, it relies on robust predictions from the underlying model.

Triple Threshold Based on double threshold, the triple threshold incorporats an additional threshold on clip-level named $\phi_c = 0.5$. This threshold first removes all frame-level $y_t(e)$ probabilities which are not predicated on $y(e)$. The equation is as below:

$$y_t(e) = \begin{cases} y_t(e), & \text{if } y(e) > \phi_c \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$\hat{y_t}(E) = dt(y_t(e)) \tag{4}$$

This method considers the clip-level output probability which can use more information.



```
mAP: [0.63237556 0.39103006 0.50618979 0.43842736 0.3010927  0.56121884
 0.37046492 0.55861284 0.91562369 0.53208237]
Quick Report:
|               |   f_measure |   precision |   recall |
|---------------|-------------|-------------|----------|
| event_based   |    0.232772 |    0.284236 | 0.214297 |
| segment_based |    0.576015 |    0.704197 | 0.503725 |
| tagging_based |    0.670371 |    0.741412 | 0.629552 |
```

Figure 7: Best result with 4-blocks, Time Masking Augmentation and Triple Threshold.

6 Conclusion

In summary, we have learnt the task of SED with its general model architecture. Plus, we have tried some experiments about adjusting hyper parameters, such as the number of blocks and the dimension of the feature vector. Moreover, we investigate the data augmentation and post processing methods, based on which, we design many experiments to see their influences. There are lots of metrics used to measure the performance. Considering Event-based F score, we list the best result in Figure 7.

References

[1] Heinrich Dinkel, Mengyue Wu, and Kai Yu. Towards duration robust weakly supervised sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:887–900, 2021.

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

[3] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, 2015.