Received: 27 November 2023 | Revised: 18 February 2024 | Accepted: 10 May 2024 | Published online: 10 June 2024

RESEARCH ARTICLE

Artificial Intelligence and Applications
2025, Vol. 3(2) 131–138
DOI: 10.47852/bonviewAIA42022164

BON VIEW PUBLISHING

Weakly Supervised Detection of Baby Cry

Weijun Tan[1,2,*], Qi Yao[2] and Jingfeng Liu[2]

[1]*LinkSprite Technologies, USA*

[2]*Deepcam Information Technologies, China*

Abstract: Detection of baby cry is an important task of baby monitoring application. Effective and real-time detection of baby detection makes the baby well cared for while releasing the care giver's pressure. Almost all existing methods for detection of baby cry use supervised support vector machines, CNN, or their varieties. In this work, we propose to use weakly supervised anomaly detection to detect baby cry in which baby cry is detected as an anomalous audio event. In this weak supervision framework, we only need weak annotation of if there is a cry in an audio file. We design a data mining technique using the pre-trained VGGish feature extractor and an anomaly detection network to obtain short audio files from long untrimmed audio files. The obtained dataset is used to train a delicately designed super lightweight CNN for cry/non-cry classification. This CNN is then used as a feature extractor in an anomaly detection framework to achieve better cry detection performance on untrimmed audio files or streams.

Keywords: baby cry, multiple-instance learning, audio classification, anomaly detection

1. Introduction

Baby monitoring is an important application of video surveillance, computer vision, and machine learning. It helps take better care of babies and reduces the burden of caregivers. Baby cry is a signal to communicate their needs including hunger, discomfort, or pain. It is used not only for caregiving but also for disease diagnosis.

The goal of is to detect the baby's cry and localize its starting and end position in an audio signal. It is a challenging task since the baby's cry sound may be mixed with different background noise in various environments, such as home and hospital.

In recent years, the detection of baby cries has been studied using both traditional machine learning and deep learning algorithms. The traditional algorithm, typically Support Vector Machine (SVM), works well on hand-crafted acoustic features in both the frequency and time domains. Typical examples include Mel frequency cepstral coefficients (MFCCs) and their varieties, pitch-related features, harmonic features, energy, zero crossing rate, etc. For a review of these approaches, the readers are referred to References [1–3].

The deep learning algorithms mostly use CNN [1–6], and some use other types of neural network [7]. In Reference [8], a CNN and other features are fused to improve detection performance. In Reference [9], it trains an LSTM-with-self-attention model on infant-cry samples automatically detected from the recorded audio through cluster analysis and Hidden Markov Model classification. It has been shown in References [1, 3] that the performance of CNN is much better than traditional machine learning. On the other hand, the complexity of the CNN may be high, preventing it be used in common embedded devices, like low-cost IP cameras or tablets. In this paper, we will address this issue by designing a super lightweight CNN.

Most, if not all, existing CNN methods use supervised learning. The work by Coro et al. [9] is defined as self-training but it generates a supervised dataset for the training of an LSTM model. Therefore, frame-level annotation is needed. This annotation is very time-consuming and prone to human mistakes. In the few datasets available online, some of the audio files [10] are trimmed and annotated, and some others [11] are untrimmed with only audio-level annotations. In this work, we propose to use weakly supervised anomaly detection [12] to detect baby cries on the audio signal. This weak supervision only requires weak annotation, i.e., if there is a cry in the audio file without frame-level annotation, therefore making the data annotation a lot easier.

Based on this weakly supervised anomaly detection, and a well-known pre-trained VGGish network [13] for audio feature extraction, we design a data mining technique to obtain frame-level datasets for supervised CNN classification. We use this dataset to train a delicately designed super lightweight CNN, which runs very fast on embedded devices. This CNN, as the feature extractor in an anomaly detection framework, gives better performance than the CNN alone.

After this work, we realized that there are multiple years of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE). For its 2024 Workshop, please refer to the website https://dcase.community/challenge2024/. Task 4 – Sound Event Detection with Heterogeneous Training Dataset and Potentially Missing Labels is most relevant to our work. Its 2024 technical reports [14–20] have a summary of all algorithms explored in Task 4.

The contribution of this paper is three-fold:

1) First, we propose to use anomaly detection to detect baby cries in audio signals. We deal with single-type audio event detection, rather than multiple-type audio event detection in the DCASE workshops.

*Corresponding author: Weijun Tan, LinkSprite Technologies, USA. Email: weijun.tan@linksprite.com

2) We design a data mining technique using the pre-trained VGGish [13] feature extractor and an anomaly detector. The obtained dataset has a similar performance to an annotated dataset on our lightweight CNN.
3) We design a super lightweight CNN. This CNN makes our framework possible to run on embedded devices.

2. Related Work

In this section, we first review the literature on baby cry detection. Then, we review the anomaly detection approaches on video signals. We borrow this anomaly detection idea and extend it to baby cry detection on audio signals.

2.1. Baby cry detection

The first step of baby cry detection is audio signal pre-processing. The main tasks are denoising and audio segmentation. The purpose of denoising is to filter out noise in unwanted frequency bands. Audio segmentation is to use a vocal activity detector to remove silent duration. In this work, we treat silent duration as non-cry segments and directly apply cry detection to it.

The signal processing features of audio signals can be categorized into cepstral domain, prosodic domain, time domain, image domain, and wavelet domain [2]. The cepstral domain is the most widely used. It includes the MFCCs, linear frequency cepstral coefficients (LFCCs), and the corresponding spectrogram. Please note that the spectrogram is 2D data, while the cepstral coefficients are several scalar data. Baby cry detection is essentially a binary classification task. A variety of classification techniques can be used, including 2-D CNN, 1-D CNN, SVM, KNN, and multiple-layer perceptron (MLP), LSTM. In previous work [3, 4, 9, 21–26], these different features and classification methods are used on their private datasets. Since their datasets are private, it is impossible to conclude which is better. However, in their comparison, a common recommendation is that CNN outperforms the traditional machine learning methods. For a complete review, please refer to References [1–3].

As a super-set of baby cry detection, audio anomaly detection typically uses unsupervised learning [27]. The work by Abbasi et al. [28] presents a large audio dataset for anomaly detection including baby cry detection. However, even though the term anomaly detection is used, in their detection algorithm, audio files are first cut into small segments then supervised learning is used to classify every segment. In Reference [5], baby cry detection in a real-world environment is explored. In DCASEs (see technical reports [14] and of previous years), multiple audio events are detected using supervised learning, self-supervised learning, or unsupervised learning.

2.2. Anomaly detection on videos

Weakly supervised anomaly detection only uses video-level annotation. This annotation only gives a binary label of abnormal or normal for a video. Sultani et al. [12] propose the MIL framework using only video-level labels and introduce the large-scale anomaly detection dataset, UCF-Crime. This work inspires quite a few follow-up studies [6, 29–35].

However, in the MIL-based methods, abnormal video labels are not easy to be used effectively. Typically, the classification score is used to tell if a snippet is abnormal or normal. This score is noisy in the positive bag, where a normal snippet can be mistakenly taken as the top abnormal event in an anomaly video. To deal with this problem, Zhong et al. [6] treat this problem as a binary classification under a noisy label problem and use a graph convolution neural (GCN) network to clear the label noise. In RTFM (robust temporal feature magnitude) [33], a temporal feature magnitude is used to select the most reliable abnormal snippets from the abnormal videos and the normal videos. They unify the representation learning and anomaly score learning by a temporal feature ranking loss, enabling better separation between normal and abnormal feature representations, and improving the exploration of weak labels compared to previous MIL methods. More details will be given later.

3. Proposed Methods

3.1. Anomaly detection in audios

The task of anomaly detection is to find and localize anomalous or abnormal events in videos. There are self-supervised methods trained only on normal datasets and weakly supervised methods trained on both abnormal and normal datasets annotated with video or audio-level labels. The weakly supervised anomaly detection in videos is first proposed in Reference [12]. It uses a multiple-instance learning (MIL) framework to find a segment in the positive (abnormal) or negative (normal) data sample whose classification scores are the maximum. Then the distance between the two segment scores is maximized for the best discriminability.

In this work, we extend the anomaly detection from the video signal to the audio signal. In videos, since the frame is 2D, therefore a segment of frames is 3D. In audio signal, a segment is a 1D audio signal and its spectrogram is 2D. So instead of a 3D CNN backbone, a 2D CNN backbone is needed.

Let V_a and V_n represent the segments in the abnormal and normal audio. The MIL expects to have the following objective function,

$$\max_{i\in B_a} f(V_a^i) > \max_{i\in B_n} f(V_n^i) \tag{1}$$

where B_a and B_n are the bags of segments in the abnormal and normal audio, f is the predicted anomaly score in the range of 0 and 1. The function *max* is taken over all instances in a bag. It is used because the segment-level annotation is not available. It is expected that in the positive bag, the highest-scored instance is a true abnormal segment. The highest-scored instance in the negative bag is the one most similar to the positive bag but is a negative instance. This makes the negative instance a hard one and therefore benefits the discriminability in the model training. To push the positive instance and negative instance further apart, the MIL ranking loss is defined as

$$l(B_a, B_n) = \max\left(0, 1 - \max_{i\in B_a} f(V_a^i) + \max_{i\in B_n} f(V_n^i)\right) \tag{2}$$

It is worth noting that this loss function looks similar to the contrastive loss function which is used to separate two or more classes as far as possible. Two regularization terms, the smoothness term and the sparsity term, are added to it. So the overall loss function is [12],

$$\begin{aligned} l = &\max\left(0, 1 - \max_{i\in B_a} f(V_a^i) + \max_{i\in B_n} f(V_n^i)\right) \\ &+ \lambda_1 \sum_i (f(V_a^{i+1}) - f(V_a^i))^2 + \lambda_2 \sum_i (f(V_a^i))^2 \end{aligned} \tag{3}$$

It is expected in Equation (1) that abnormal segments have higher scores than normal segments. However, this is not always true.

A few methods [30–35] have been studied how to improve the score quality so that the correct abnormal segment is chosen in the abnormal bag. In the work RTFM [33], a different approach is used. Instead of using the classification score as the criterion to choose the abnormal segment, the authors propose to use a feature magnitude, which they believe has better discriminability between abnormal and normal instances. Furthermore, they propose to use multiple instances whose feature amplitude is the largest and call them the top-k instances. In their approach, the MIL ranking loss is defined by the feature magnitude,

$$l_F(B_a, B_n) = \max(0, m - d(m(V_a^{top-k}), m(V_n^{top-k})) \quad (4)$$

where X_a^{top-k} and X_n^{top-k} are the top-k segments whose feature magnitudes are the largest k instances out of the abnormal and normal bag, f is the feature magnitude function, m is a predefined margin, and d is the defined distance function between the two sets of top-k features. In their implementation, this function is simply the square of the mean of the top-k feature magnitude.

The standard cross-entropy loss is used as the classification loss. However, it is applied to the top-k segments whose feature magnitude is the largest. If $k > 1$, the scores are averaged before feeding into the cross-entropy loss function,

$$l_S(B_a, B_n) = -y \log(f(X^{top-k}) - (1-y)\log(1 - f(X^{top-k})) \quad (5)$$

The same smoothness term and sparsity term are also used, so the overall loss function is,

$$l(B_a, B_n) = l_S + \alpha l_F + \lambda_1 \sum_i (f(V_a^{i+1}) - f(V_a^i))^2 + \lambda_2 \sum_i (f(V_a^i))^2 \quad (6)$$

where α, λ_1, and λ_2 are predefined weight factors.

In addition, a multi-scale (dilated convolution) non-local aggregation (MSNL) block is used [33] on the feature extracted from the pre-trained CNN backbone. This block is also important for the feature magnitude training. Without this block, the feature is fixed and cannot be learned. The MSNL is used in Reference [33], but other simpler networks, e.g., a few full-connection (FC) layers may also work.

3.2. Proposed anomaly detection framework

The overall block diagram of our proposed network is illustrated in Figure 1. The style of the figure is borrowed from Reference [12]. A framework similar to RTFM [33] is used, with all necessary modifications for anomaly detection of baby cries in audio.

The abnormal or normal audio signal is first divided into a certain number of equal-length segments. We use 16 in the figure as an example. Every segment is called an instance in the positive or negative bag of instances. All instances pass through a pre-trained CNN backbone, and CNN features are extracted. In Figure 1, the CNN backbone is called BlazeNet – our delicately designed super lightweight network. This CNN feature is fed into a feature refinement network and a second CNN feature is extracted. The features of the positive instance bag form the positive feature bag, the same is true for the negative feature bag. The top-k instances whose feature magnitudes are the largest among this bag are selected. The classification network is typically two or three FC layers. Through the backpropagation of the loss function in Equation (5), the feature magnitude and the classification are both learned at the same time.

In the implementation, when the audiofile is very short and the audio signal is divided into 16 segments, every segment may not belong enough for a frame. So CNN feature is extracted for every frame, and then, linear interpolation is used to generate features for 16 segments.

3.3. BlazeNet

Our goal of this study is to design a baby cry detection framework that can work efficiently on embedded devices.

So the CNN backbone network must be super lightweight, and at the same time, achieve good performance. We have tested the popular MobileNet, ShuffleNet, SqueezeNet, and find that they are still too large, no need to mention the popular VGGish-Net widely used in audio recognition. We take the backbone from the BlazeFace in Reference [36]. We make changes so that the input size is 64 × 64 and the output feature size is 224. We use 16 BlazeBlocks, where the 11th BlazeBlock output is classified by the first classifier *FC*1(88,2), and the 16th BlazeBlock output is classified by a second classifier *FC*2(96,6). The outputs of these two classifiers are flattened and concatenated and then classified

Figure 1
Anomaly detection of baby cry block diagram. The log-Mel-spectrum is not shown

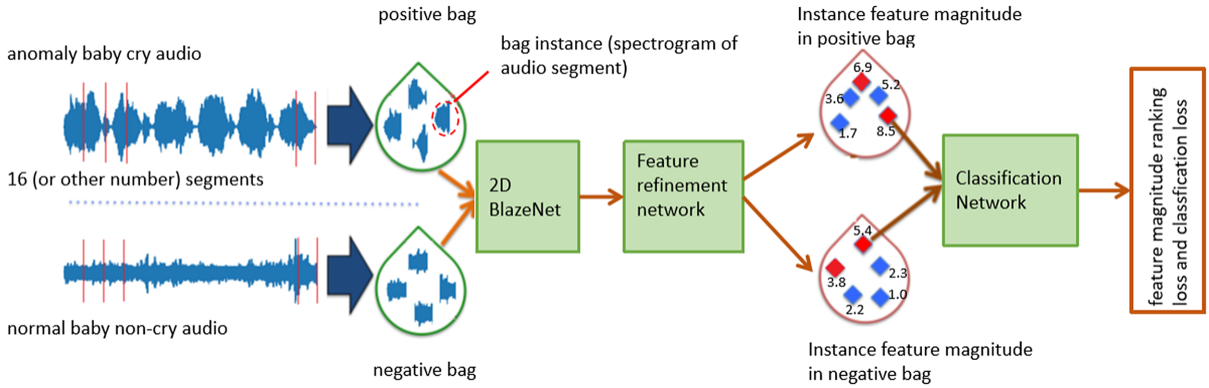
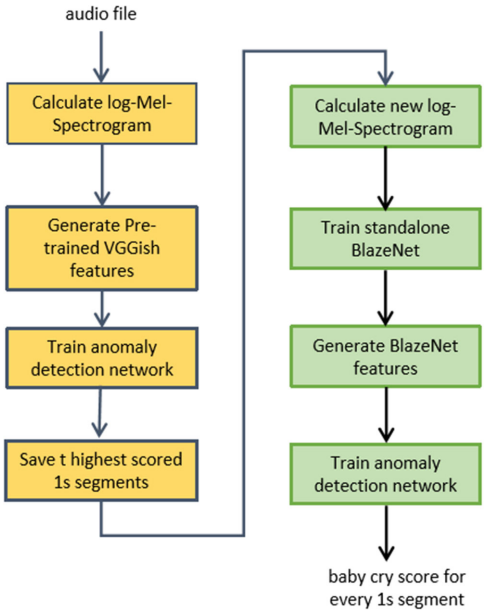


Table 1
Layers of our BlazeNet. The BlazeNet parameters are number of input channels, number of output channels, kernel size, and stride. The sequence of layers is from top to bottom, from left to right

BlazeBlock	Others
–	Conv2D(3,24,5,2)
BlazeBlock-1(24,24,3,1)	–
BlazeBlock-2(24,28,3,1)	–
BlazeBlock-3(28,32,3,2)	–
BlazeBlock-4 (32,36,3,1)	–
BlazeBlock-5(36,42,3,1)	–
BlazeBlock-6(42,48,3,2)	–
BlazeBlock-7(48,56,3,1)	–
BlazeBlock-8(56,64,3,1)	–
BlazeBlock-9 (64,72,3,1)	–
BlazeBlock-10(72,80,3,1)	–
BlazeBlock-11(80,88,3,1)	*FC*1(88,2)
BlazeBlock-12(88,96,3,2)	–
BlazeBlock-13(96,96,3,1)	–
BlazeBlock-14(96,96,3,1)	–
BlazeBlock-15(96,96,3,1)	–
BlazeBlock-16(96,96,3,1)	*FC*2(96,6)
–	cat(*FC*1,*FC*2)
–	*FC*(224,2)

Figure 2
Pipeline of data mining and anomaly detection for baby cry detection



by the final *FC*(224,2) classifier. The details of our BlazeNet are listed in Table 1. The total number of parameters of this model is 89,680 in PyTorch.

3.4. Data mining datasets for BlazeNet

The CNN backbone in Figure 1 is pre-trained and fixed when the anomaly detection network (feature refinement network and classification network) is trained. If this CNN and the anomaly detection network are trained together end-to-end, the GPU can be easily overflowed. For this reason, all previous anomaly detection methods in videos use pre-trained 3D CNN.

So we need to pre-train the CNN backbone before the whole anomaly detector framework can work to detect baby cries in audio. To do so, we need some training data (herein including validation and test dataset without confusion). One way to do so is to prepare some audio data manually, which we do in this work. More details will be given later. However, manually annotating audio data is very time-consuming and is prone to human mistakes. We propose a second way to mine training data from a weakly annotated dataset and a different pre-trained CNN backbone.

In Reference [11], the authors publish a large audio dataset called AudioSet. At the same time, they publish a pre-trained VGGish backbone [13] for CNN feature extraction. With their default settings, the log-Mel-spectrogram is used on a 0.96-s frame. The output CNN feature is 128-D. We use this VGGish network to exact CNN features for audio files, then apply the anomaly detection framework to it. So to make it clear, the 2D BlazeNet in Figure 1 is replaced with the pre-trained VGGish network. After the anomaly detection network (feature refinement network and classification network) is trained, the framework is set to inference mode, and all training, validation, and test datasets are processed. Please note that, in the inference mode, the audio files are not divided into 16 segments. Instead, the audio signal in the form of a 1s-frame is used. Then, the top t 1s segments whose classification scores are the largest are saved. In this way, we obtain the new training, validation, and test dataset to train the 2D BlazeNet.

The pipeline of the data mining and anomaly detection framework for baby cry is shown in Figure 2. The blocks on the left perform the data mining, and the blocks on the right perform the anomaly detection of baby cry detection. Please note only the training procedure of anomaly detection is plotted, and the testing procedure can be derived accordingly.

4. Experiments

4.1. Datasets

Even though there are quite some publications on baby cry detection, none of the used datasets are publicly available. We search online and organize the datasets from the sources listed in Table 2. Please note that the background of all these datasets is clean except for the AudioSet [11], whose background is very

Table 2
Dataset sources of baby cry we find online

Source	Length	Annotation	Cleaned number	Background
[10]	5s	Baby Cry, other 3	108, 324	clean
[12]	7s	Baby Cry only	482	clean
ESC-50 [10]	5s	Baby Cry, many others	40, 1960	clean
AudioSet [11]	Untrimmed	Baby Cry, many others	1364, a lot	noisy

noisy. We clean the datasets and filter out the ambiguous cases. The numbers of cleaned samples are listed in the table.

In total, we have 2624 baby cry audios and tremendous other audios. We use all other audios in References [10] and then collect randomly other audios in Reference [11]. The total number of other non-cry audios is about the same as the number of baby cry audio.

For the training of BlazeNet, we have two audio frame lengths, 5s, and 1s. If the length of an audio is longer than two times the frame length, then more than one frame can be cut from an audio file. After the cutting, we manually check if there is a segment of a baby cry in the audiofile. The total audio frames are randomly divided into training, validation, and test datasets with a ratio 8:1:1.

For anomaly detection, we use two audio lengths: one is 5s, and the other is the original audio file length. When the length is 5s, it is the same dataset as above. In this case, the frame length is only 1s. When the length is the original audio file length, the frame length is also 1s.

Until very recently, we found that a new baby cry dataset was released in Reference [5]. However, the link to where the dataset is saved is broken. So it is not publicly available.

4.2. Implementation details

For the BlazeNet as a standalone CNN classification network on baby cry detection, we implement it in PyTorch. The SGD is used as the optimizer with a starting learning rate of 0.001 and momentum of 0.9. The training runs 60 epochs. The learning rate is decayed by a factor of 0.1 every 20 epochs. A batch size of 32 is used. A single Nvidia 1080TI GPU card is used.

For anomaly detection, we use the RTMF codebase [33] in PyTorch. Every audio file is divided into 5 segments when the input audio file is 5s long. It is divided into 10 segments when the original audiofiles are used. The top-k is set to 2. Two dataset iterators, one for the abnormal data and the other for the normal data, are used. This way, the pairing of abnormal and normal data is random, even when the numbers of abnormal and normal samples are different. An initial training rate of 1E-3 is used, and the training runs 20000 steps (we do not use epochs because the abnormal data loader and the normal data loader are iterating). A batch size of 128 is used.

For the VGGish input, the default log-Mel spectrogram parameters are used, specifically, sampling rate = 16 K Hz, number of frames in batch = 96, number of Mel bands = 64, FFT window length = 0.025s, FFT hop length = 0.01s, min Mel frequency = 125 Hz, max Mel frequency = 7500 Hz, log offset = 0.01, example hop seconds = 0.96. The only change we make is for example window seconds = 1s so that we have a 96 × 64 log-Mel spectrogram output for every 1s of audio.

For the BlazeNet input, we use different log-Mel spectrogram parameters and we use the Librosa library. When the example window seconds = 1s, sampling rate = 8 K Hz (this is the audio signal sampling rate on most IP cameras), number of Mel bands = 64, FFT window length = 0.064s, FFT hop length = 0.01475, min Mel frequency = 0 Hz, max Mel frequency = 8000 Hz. Other default parameters are used. Please note that we use an FFT hop length such that the spectrogram out size is 64 × 64, which is required by BlazeNet. Resizing the Mel spectrogram array is not recommended since it causes performance loss. When the example window seconds = 5s, the FFT window length and FFThop length are adjusted accordingly so the spectrogram out size is 64 × 64.

Please note that, in all our performance evaluations, the data unit is the 1s audio segment. In practice, we give detection results for every 1s audio signal input.

4.3. Data mining using VGGish

We first test how VGGish [13] features work in the anomaly detection network. The performance must be good for the data mining to work well. From the standpoint of anomaly detection, positive instances must have a larger score than negative instances (see Equation (1)). In other words, the largest scored instances in the abnormal bag must be truly positive instances.

We do this test on both the trimmed 5s audio files and the untrimmed audiofiles as datasets. VGGish features are extracted for 1s segments sequentially without overlap in every audiofile. These features and their audio labels are used in training and testing the anomaly detection network.

The experiment results are listed in Table 3. We observe that the validation and test accuracy results are all higher than 0.90. We believe this good performance will make the data mining method work well, which will be verified later with the performance of the standalone BlazeNet classification.

4.4. BlazeNet classification results

We first train the BlazeNet on trimmed 1s audios. The trimmed 5s audios are prepared manually. To get a 1s audio dataset, every 5s audio is cut into 5 segments of 1s-long audio without overlap. When the 1s audios are used in training, there are two modes. In the first mode, all 5 segments of every 5s long audio file are used. In the second mode, only 2 randomly selected segments are used. This is for a fair comparison with the data mining method, where only the top 2 segments are saved as training datasets.

Please note that in all these experiments, only the training dataset changes, while the validation and test datasets stay the same. In the last experiment, long untrimmed audios are added to the training dataset. We argue that always using short 1s audio files as validation and test datasets is reasonable because, in practical applications, a decision per 1s audio signal is preferred to avoid long latency.

The experiment results are listed in Table 4, where all the accuracy results are taken at default threshold = 0.5. The accuracy result of using all 1s segments is the best, and the one using random 2 segments is a little bit worse. This is probably because the selected segments do not cover as many cases as using all 1s segments.

Considering the 5s audios are well annotated, any 1s segment should be good to be used. When top-2 1s segments are mined from 5s audios, we expect to have the same performance as the random 2 1s segments, however, the results are not so. The accuracy is worse than using the annotated data by 1%. We test further two cases, one using only mined positive data, and the other using only mined negative data. The results show that hard

Table 3
Performance of anomaly detection of baby cry using VGGish features. Accuracy is measured at default classification threshold = 0.5

Dataset	Val Acc	Test Acc
5s audios	0.9447	0.9312
Untrimmed audios	0.9370	0.9223

Table 4
Performance of BlazeNet as a standalone baby cry classifier. Accuracy is measured at default classification threshold = 0.5

Dataset	Val Acc	Test Acc
All 1s segments from 5s audios	0.8868	0.8820
Random 2 1s-segments from 5s audios	0.8784	0.8654
Mined top-2 1s-segments from 5s audios	0.8609	0.8542
Mined positive top-2 1s-segments from 5s audios	0.8748	0.8562
Mined negative top-2 1s-segments from 5s audios	0.8748	0.8622
Mined top-2 1s-segments from untrimmed audios	0.8142	0.8292

Table 5
Performance of anomaly detection of baby cry using BlazeNet features. Two trained BlazeNet backbones from Table 4 are used. Accuracy is measured at default classification threshold = 0.5

Dataset	Backbone	Val Acc	Test Acc
5s audios	Table 4. Line-1	0.9457	0.9302
Untrim audios	Table 4. Line-1	0.9141	0.9020
Untrim audios	Table 4. Line-6	0.8652	0.8473

negative samples with the highest scores in the mined negative data are preferred, while the positive samples with the highest scores are not preferred.

The performance of the mined data from the long untrimmed audio is even worse. This is understandable since some negative samples may be chosen as positive samples. However, as a backbone for an anomaly detection framework, the backbone does not need to be perfect. As in anomaly detection in videos, a pre-trained 3D CNN is used without training on the anomaly detection dataset [12, 33]. The experiment result will be shown in the next subsection.

4.5. Anomaly detection results

In this subsection, we test the anomaly detection framework shown in Figure 1 with the fixed BlazeNet, which has been trained in the previous subsection.

In the first experiment, we use this method on the manually annotated 5s audio files. The goal is to find the detection results on every 1s audio segment and measure the accuracy. In this experiment, since there are only 5 1s segments in a 5s audiofile, the number of segments in anomaly detection is set to 5, and the top-kis set to 2.

In the second experiment, long untrimmed audiofiles are used as training datasets, while 5s audio files are used as validation and test datasets. After looking at the distribution of the audiofile length, we set the number of segments in anomaly detection to 10. The top-k is still set to 2.

The experiment results are listed in Table 5. We observe that the performance is very close to that using the VGGish feature in Table 3, while the complexity of the BlazeNet is a lot lower than that of the VGGish network. For BlazeNet trained at Table 4 Line 6, the performance is worse than the one trained at Table 4 Line 1. So the quality of the BlazeNet feature does matter in the anomaly detection performance. So manually annotating some datasets for the BlazeNet is preferred.

4.6. Discussion: Anomaly detection vs. classification

Our goal is to find a solution for baby detection on embedded devices, so we ignore any results directly using the VGGish network in inference mode. When comparing the results of BlazeNet in Tables 4 and 5, we see that the performance of anomaly detection is already more than 2% better than that of the standalone BlazeNet.

Furthermore, we note that we only use the accuracy at default threshold = 0.5 in all these experiments (We use this threshold because it is dominantly used in the training of a binary classifier). So we do some analysis in terms of the max $F1$ score and the ROC curve.

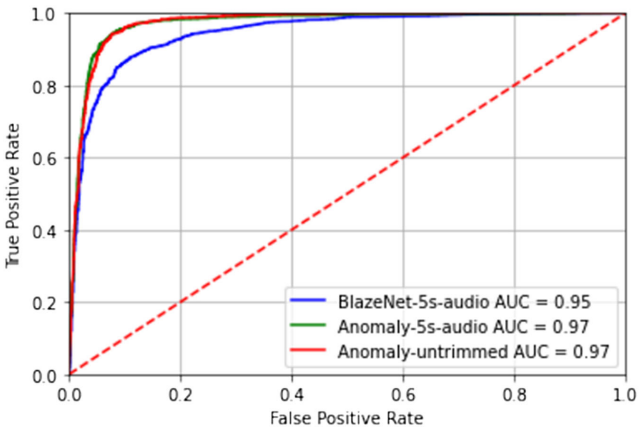
We collect prediction scores of all 1s audio segments in the test dataset and then calculate the max $F1$ score and the corresponding threshold. Finally, we calculate the test accuracy at this threshold. The results are listed in Table 6. It is observed that for the BlazeNet since it is a binary classification, the threshold to achieve the max $F1$ is near 0.5, and the test accuracy at this threshold is almost identical to the one in Table 4. While for anomaly detection, for its nature of MIL ranking loss, the threshold to achieve the max $F1$ is pushed to the 1.0 side.

The ROC curves of the same three cases in Table 6 are plotted in Figure 3. The gain from the anomaly detection is obvious, and so is the ROC AUC.

Table 6
Performance of anomaly detection of baby cry using BlazeNet features. Two trained BlazeNet backbones from Table 4 are used. Accuracy is measured at default classification threshold = 0.5

Dataset	Method	$F1$-max	Test Acc
5s audios	BlazeNet	0.8873	0.8836
5s audios	Anomaly	0.9241	0.9305
Untrim audios	Anomaly	0.9306	0.9360

Figure 3
ROC curves of standalone BlazeNet and the anomaly detection



5. Conclusion

In this paper, we studied baby cry detection in audio. We extend an anomaly detection framework RTMF [33] from video to audio and use it to detect baby cries in audio. Since anomaly detection only requires audio-level annotation, it reduces significantly the workload of annotating datasets for supervised training. Furthermore, to make the detector simple and run fast, we designed a super lightweight BlazeNet for baby cry/non-cry classification. We designed a few experiments and showed that anomaly detection can achieve better performance than the standalone BlazeNet classification with a little bit of extra complexity.

To overcome the lack of dataset problem, we proposed to use the anomaly detection framework with a pre-trained VGGish backbone to mine training data from the AudioSet [11]. Even though the BlazeNet trained with this data is not as well as the one using manually annotated data, using this BlazeNet in an anomaly detection framework still works relatively well.

This study demonstrates that weakly supervised anomaly detection is a promising solution for baby cry detection. We can reasonably believe that this method can extend to other single-type audio event detection.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in (1) GitHub at https://github.com/gveres/donateacry-corpus; (2) audioset at https://doi.org/10.1109/ICASSP.2017.7952261, reference number [11]; (3) ESC 50 at https://doi.org/10.1145/2733373.2806390, reference number [10].

Author Contribution Statement

Weijun Tan: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Qi Yao:** Resources, Data curation. **Jingfeng Liu:** Resources, Data curation, Project administration.

References

[1] Cohen, R., Ruinskiy, D., Zickfeld, J., IJzerman, H., & Lavner, Y. (2020). Baby cry detection: Deep learning and classical approaches. In W. Pedryca, & S. M. Chen (Eds.), *Development and analysis of deep learning architectures* (pp. 171–196). Springer. https://doi.org/10.1007/978-3-030-31764-5_7

[2] Ji, C., Mudiyanselage, T. B., Gao, Y., & Pan, Y. (2021). A review of infant cry analysis and classification. *EURASIP Journal on Audio, Speech, and Music Processing*, *2021*(1), 8. https://doi.org/10.1186/s13636-021-00197-5

[3] Torres, R., Battaglino, D., & Lepauloux, L. (2017). Baby cry sound detection: A comparison of hand crafted features and deep learning approach. In *Engineering Applications of Neural Networks: 18th International Conference*, 168–179. https://doi.org/10.1007/978-3-319-65172-9_15

[4] Ferretti, D., Severini, M., Principi, E., Cenci, A., & Squartini, S. (2018). Infant cry detection in adverse acoustic environments by using deep neural networks. In *2018 26th European Signal Processing Conference*, 992–996. https://doi.org/10.23919/EUSIPCO.2018.8553135

[5] Yao, X., Micheletti, M., Johnson, M., Thomaz, E., & de Barbaro, K. (2022). Infant crying detection in real-world environments. In *2022 IEEE International Conference on Acoustics, Speech and Signal Processing*, 131–135. https://doi.org/10.1109/ICASSP43922.2022.9746096

[6] Zhong, J. X., Li, N., Kong, W., Liu, S., Li, T. H., & Li, G. (2019). Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1237–1246.

[7] Lahmiri, S., Tadj, C., & Gargour, C. (2021). Biomedical diagnosis of infant cry signal based on analysis of cepstrum by deep feedforward artificial neural networks. *IEEE Instrumentation & Measurement Magazine*, *24*(2), 24–29. https://doi.org/10.1109/MIM.2021.9400952

[8] Zayed, Y., Hasasneh, A., & Tadj, C. (2023). Infant cry signal diagnostic system using deep learning and fused features. *Diagnostics*, *13*(12), 2107. https://doi.org/10.3390/diagnostics13122107

[9] Coro, G., Bardelli, S., Cuttano, A., Scaramuzzo, R. T., & Ciantelli, M. (2023). A self-training automatic infant-cry detector. *Neural Computing and Applications*, *35*(11), 8543–8559. https://doi.org/10.1007/s00521-022-08129-w

[10] Piczak, K. J. (2015). ESC: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia*, 1015–1018. https://doi.org/10.1145/2733373.2806390

[11] Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., …, & Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, 776–780. https://doi.org/10.1109/ICASSP.2017.7952261

[12] Sultani, W., Chen, C., & Shah, M. (2018). Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6479–6488.

[13] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., …, & Wilson, K. (2017). CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, 131–135. https://doi.org/10.1109/ICASSP.2017.7952132

[14] Schmid, F., Primus, P., Morocutti, T., Greif, J., & Widmer, G. (2024). Improving audio spectrogram transformers for sound event detection through multi-stage training. In *Detection and Classification of Acoustic Scenes and Events 2024*.

[15] Nam, H., Min, D., Choi, S., Choi, I., & Park, Y. H. (2024). Self training and ensembling frequency dependent networks with coarse prediction pooling and sound event bounding boxes. In *Detection and Classification of Acoustic Scenes and Events 2024*.

[16] Yue, H., Wang, Z., Mu, D., Sun, H., Jiang, Y., Zhanf, Z., & Yin, J. (2024). Local and global features fusion for sound event detection with heterogeneous training dataset and potentially missing labels. In *Detection and Classification of Acoustic Scenes and Events 2024*.

[17] Chen, W. Y., Lu, C. L., Chuang, H. F., Cheng, Y. H., & Chan, B. C. (2024). Sound event detection with heterogeneous training

dataset and potentially missing labels for DCASE 2024 task 4. In *Detection and Classification of Acoustic Scenes and Events 2024*.

[18] Son, S. W., Park, J., Kim, H. K., Vesal, S., & Lim, J. E. (2024). Sound event detection based on auxiliary decoder and maximum probability aggregation for DCASE Challenge 2024 Task 4. In *Detection and Classification of Acoustic Scenes and Events 2024*.

[19] Chen, J., Cai, X., Liu, Z., Zhang, H., Zuo, L., & Wu, M. (2024). Semi-supervised sound event detection based on pretrained models for DCASE 2024 task 4. In *Detection and Classification of Acoustic Scenes and Events 2024*.

[20] Huang, W., Han, B., Chen, X., Fan, P., Lu, C., Lv, Z., . . . , & Qian, Y (2024). Sound event detection enhanced by scene information for DCASE Challenge 2024 Task 4. In *Detection and Classification of Acoustic Scenes and Events 2024*.

[21] Chang, C. Y., & Tsai, L. Y. (2019). A CNN-based method for infant cry detection and recognition. In L. Barolli, M. Takizawa, F. Xhafa, & T. Enkido (Eds.), *Web, artificial intelligence and network applications* (pp. 786–792). Springer. https://doi.org/10.1007/978-3-030-15035-8_76

[22] Dewi, S. P., Prasasti, A. L., & Irawan, B. (2019). Analysis of LFCC feature extraction in baby crying classification using KNN. In *IEEE International Conference on Internet of Things and Intelligence System*, 86–91. https://doi.org/10.1109/IoTaIS47347.2019.8980389

[23] Gu, G., Shen, X., & Xu, P. (2018). A set of DSP system to detect baby crying. In *2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference*, 411–415. https://doi.org/10.1109/IMCEC.2018.8469246

[24] Lavner, Y., Cohen, R., Ruinskiy, D., & Ijzerman, H. (2016). Baby cry detection in domestic environment using deep learning. In *IEEE International Conference on the Science of Electrical Engineering*, 1–5. https://doi.org/10.1109/ICSEE.2016.7806117

[25] Lim, H., Park, J. S., Lee, K., & Han, Y. (2017). Rare sound event detection using 1D convolutional recurrent neural networks. In *Detection and Classification of Acoustic Scenes and Events 2017*, 80–84.

[26] Manikanta, K., Soman, K. P., & Manikandan, M. S. (2019). Deep learning based effective baby crying recognition method under indoor background sound environments. In *4th International Conference on Computational Systems and Information Technology for Sustainable Solution*, 1–6. https://doi.org/10.1109/CSITSS47250.2019.9031058

[27] Koizumi, Y., Kawaguchi, Y., Imoto, K., Nakamura, T., Nikaido, Y., Tanabe, R., . . . , & Harada, N. (2020). Description and discussion on DCASE2020 Challenge Task 2: Unsupervised anomalous sound detection for machine condition monitoring. *arXiv Preprint:2006.05822*.

[28] Abbasi, A., Javed, A. R. R., Yasin, A., Jalil, Z., Kryvinska, N., & Tariq, U. (2022). A large-scale benchmark dataset for anomaly detection and rare event classification for audio forensics. *IEEE Access*, *10*, 38885–38894. https://doi.org/10.1109/ACCESS.2022.3166602

[29] Degardin, B., & Proença, H. (2020). Human activity analysis: Iterative weak/self-supervised learning frameworks for detecting abnormal events. In *IEEE International Joint Conference on Biometric*, 1–7. https://doi.org/10.1109/IJCB48548.2020.9304905

[30] Feng, J. C., Hong, F. T., & Zheng, W. S. (2021). MIST: Multiple instance self-training framework for video anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14009–14018.

[31] Li, S., Liu, F., & Jiao, L. (2022). Self-training multi-sequence learning with transformer for weakly supervised video anomaly detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(2), 1395–1403. https://doi.org/10.1609/aaai.v36i2.20028

[32] Lv, H., Zhou, C., Cui, Z., Xu, C., Li, Y., & Yang, J. (2021). Localizing anomalies from weakly-labeled videos. *IEEE Transactions on Image Processing*, *30*, 4505–4515.

[33] Tian, Y., Pang, G., Chen, Y., Singh, R., Verjans, J. W., & Carneiro, G. (2021). Weakly-supervised video anomaly detection with robust temporal feature magnitude learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4975–4986.

[34] Wu, J., Zhang, W., Li, G., Wu, W., Tan, X., Li, Y., . . . , & Lin, L. (2021). Weakly-supervised spatio-temporal anomaly detection in surveillance video. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 1172–1178.

[35] Wu, P., & Liu, J. (2021). Learning causal temporal relation and feature discrimination for anomaly detection. *IEEE Transactions on Image Processing*, *30*, 3513–3527. https://doi.org/10.1109/TIP.2021.3062192

[36] Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., & Grundmann, M. (2019). BlazeFace: Sub-millisecond neural face detection on mobile GPUs. *arXiv Preprint:1907.05047*.

How to Cite: Tan, W., Yao, Q., & Liu, J. (2025). Weakly Supervised Detection of Baby Cry. *Artificial Intelligence and Applications*, *3*(2), 131–138. https://doi.org/10.47852/bonviewAIA42022164