DEEP EMBEDDINGS FOR RARE AUDIO EVENT DETECTION WITH IMBALANCED DATA

Vipul Arora [*]

Indian Institute of Technology Kanpur, India

Ming Sun and Chao Wang

Amazon Alexa, Cambridge MA, USA

ABSTRACT

In this paper, we present a method to handle data imbalance for classification with neural networks, and apply it to acoustic event detection (AED) problem. The common approach to tackle data imbalance is to use class-weights in the objective function while training. An existing more sophisticated approach is to map the input to clusters in an embedding space, so that learning is locally balanced by incorporating inter-cluster and inter-class margins. On these lines, we propose a method to learn the embedding using a novel objective function, called triple-header cross entropy. Our scheme integrates in a simple way with back-propagation based training, and is computationally more efficient than general hinge-loss based embedding learning schemes. The empirical evaluation results demonstrate the effectiveness of the proposed method for AED with imbalanced training data.

Index Terms— Data imbalance, embedding space, acoustic event detection, neural networks, classification

1. INTRODUCTION

Given an imbalanced dataset for classification, several methods have been proposed to make the learning effective [1–3]. One of the common approaches is to over-sample the under represented classes and/or under-sample the over represented classes by some factor such that in every batch or epoch, the learning algorithm sees the same number of instances from each class. Similar effect can be obtained by modifying the loss function so as to give higher weights to the samples of the under represented class. Data augmentation by generating synthetic data is another approach to handle data imbalance [4].

Another line of attack to this problem is by using embeddings [5–7]. The idea here is that instead of optimizing the classification loss function directly, we learn an embedded space with additional constraints to take care of problems like data imbalance. The embedded features thus obtained can be classified by a simple classifier. Huang *et al.* [6] proposed cluster based constraints to take care of data imbalance in image analysis. The embedding space is learned to optimize class-based as well as cluster-based margins with the help of a triple-header loss function that uses quintuplets sampled from the data. However, finding the quintuplets for each sample in every iteration, on the basis of pair-wise distances between points, is a time-consuming process, with time complexity proportional to the square of the number of samples. In this paper, we propose a back-propagation based scheme to learn the embedding. We propose a loss function that incorporates class-based as well as cluster-based constraints in an efficient way, without having to sample quintuplets or triplets from the data, reducing the time complexity to be linear w.r.t. the number of samples.

We apply this method to the problem of acoustic event detection (AED), which entails detecting specific events in the surroundings from the audio. AED has got great applications [8] in areas like home security, wildlife monitoring, self-driving cars, smart home appliances, etc. DCASE [8] is a popular challenge that has been pushing efforts on this problem. Various approaches have been proposed for AED, like hidden Markov models [9], non-negative matrix factorization [10], bag-of-features [11]. Neural network based approaches using recurrent neural networks are being widely used these days [12–15]. Standard machine learning methods assume a balanced distribution of data, e.g., in DCASE challenges. However, in practice, the data is highly imbalanced across different categories of sound events, e.g. AudioSet data [16]. The effort involved in data collection and annotation makes it extremely difficult to have balanced data in real applications, especially while dealing with rare events.

2. ACOUSTIC SCENE CLASSIFICATION WITH NEURAL NETWORKS

Given the acoustic features $\boldsymbol{x}_i$ of an audio sample $i \in \{1, 2, ..., I\}$, the goal is to label $\boldsymbol{x}_i$ with L acoustic event labels. Here, $\boldsymbol{x}_i \in \mathbb{R}^{X \times N}$, with X as the feature dimension at each time frame, and N as the number of time frames. A prediction model f is trained such that $f(\boldsymbol{x})$ predicts the corresponding acoustic event labels. The output of the model, $f(\boldsymbol{x}_i)$, is a vector $\boldsymbol{y}_i \in \mathbb{R}^L$, whose element y_{il} is the probability that the event l is present in the audio sample i. The model is trained in a supervised way. While training, the ground truth label vector $\boldsymbol{t}_i \in \{0, 1\}^L$ is given for each sample i.

We use a neural network as the prediction model. Various layers of the network successively perform non-linear trans-

[*]V. Arora performed this work while at Amazon Alexa, Cambridge MA. Thanks to Weiran Wang for useful discussions.

formations on the input $\boldsymbol{x}_i$. To obtain the final output $\boldsymbol{y}_i$, a sigmoid non-linearity is applied to the output of the last layer. The loss function that is optimized during training is typically the binary cross-entropy loss, given by

$$\mathcal{L}_{\text{bin}} = -\sum_{i,l} \{t_{il} \log y_{il} + (1 - t_{il}) \log(1 - y_{il})\} \tag{1}$$

The above loss works well for a balanced dataset. The number of samples having the label l is given by $I_l = \sum_i t_{il}$. For a balanced dataset the values I_l are approximately the same for all l, while this is not so in the case of imbalanced dataset. To handle imbalanced dataset, the loss function can be modified as

$$\mathcal{L}_{\text{wBin}} = -\sum_{i,l} \{w_l t_{il} \log y_{il} + (1 - t_{il}) \log(1 - y_{il})\} \tag{2}$$

where, $w_l \in \mathbb{R}$ is chosen to be inversely proportional to I_l. The above loss function ensures that the contribution of each class to the loss function is balanced.

3. PROPOSED METHOD

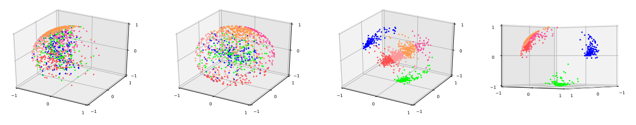
In order to handle the data imbalance problem in classification/detection, we learn an embedding with the help of constraints, which prevent any local imbalance in the data [6], while maintaining the class-discriminativeness of the embedded features. These embedded features can be further classified with a simple classifier, say a shallow neural network.

Given the input features $\boldsymbol{x}_i$ and ground truth labels $\boldsymbol{t}_i$, we learn an embedding $\mathcal{E}$, such that the embedded features $\mathcal{E}(\boldsymbol{x}_i)$ can readily be used to predict the labels $\boldsymbol{y}_i$.

We model the embedding with a neural network with the output layer as the embedding layer. We constrain the output $\mathcal{E}(\boldsymbol{x}_i)$ to be L2 normalized. Starting from an embedding initialized randomly or from a pre-trained network, we cluster all the data $\mathcal{E}(\boldsymbol{x}_i)$ into roughly equal size clusters such that each cluster contains data of a single class. Here, class refers to each distinct value of $\boldsymbol{t}_i$. We denote the class of the sample i with c_i The size of each cluster is set to be the size of the smallest class. For the class c with I_c samples, the number of clusters is $K_c = \lfloor I_c / \min_{c'}\{I_{c'}\}\rfloor$. Let $k_i \in \{1, ..., K\}$ denote the cluster assigned to the sample i, where $K = \sum_c K_c$. Since, each class has several clusters associated with it, let c_k denote the class associated with the cluster k. This clustering is performed using spherical K-means [17], i.e., K-means using cosine similarity rather than Euclidean distance, because the embedding space is a hypersphere.

3.1. Learning the embedding

To learn the embedding without local class imbalance, the basic idea is to map the input to clusters in the embedding space such that the same-class clusters are closer to each other



(a) Before learning (b) After 2 iterations (c) After 5 iterations (d) Another perspective of (c)

Fig. 1. Demonstration of Embedding learning with toy example with three classes, viz., blue, green and red (different shades of red for different clusters).

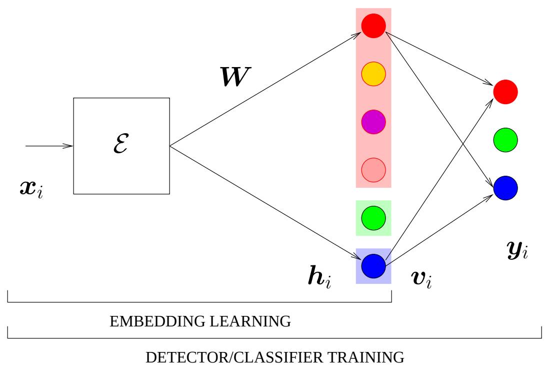


Fig. 2. Overall schematic of the proposed model

than the different-class clusters. To achieve this, Huang *et al.* [6] propose a triple-header hinge loss function, which requires finding a quintuplet for each sample. The creation of this quintuplet table is a computationally expensive step, with computational complexity $O(I^2)$. Here we propose a triple-header cross-entropy function to learn the embedding, reducing the computational complexity to $O(IK)$, with $K \ll I$.

Given a sample i, we can constrain its distance from the centroids of other clusters, rather than from specific points (as in [6]). Mathematically, denoting cosine similarity with $S(\cdot,\cdot)$ and centroid of cluster k as $\boldsymbol{W}_k$, we need to formulate the loss function such that

$$\begin{aligned} S(\mathcal{E}(\boldsymbol{x_i}), \boldsymbol{W}_{k_i}) &> \max_{k':k' \neq k_i, c_{k'} = c_i} S(\mathcal{E}(\boldsymbol{x}_i), \boldsymbol{W}_{k'}) \\ &> \max_{k'':k'' \neq k_i, c_{k''} \neq c_i} S(\mathcal{E}(\boldsymbol{x}_i), \boldsymbol{W}_{k''}) \end{aligned} \tag{3}$$

This equation implies that the sample embedding's cosine similarity with the centroid of cluster k_i (first term), is more than that with the centroid of the nearest same-class cluster (second term), which in turn, is more than that with the centroid of the nearest cluster of another class (third term).

Given the network $\mathcal{E}$, we add a dense layer of dimension $\sum_c K_c$ that maps the embedding features $\mathcal{E}(\boldsymbol{x_i})$ to a cluster one-hot vector $\bar{\boldsymbol{v}}_i$, such that $\bar{v}_{ik} = \delta(k - k_i)$. We define

$$\boldsymbol{h}_i = \boldsymbol{W}^\intercal \mathcal{E}(\boldsymbol{x_i}) \tag{4}$$

where $\boldsymbol{W}$ is the weight matrix of the dense layer. We note that $\mathcal{E}(\boldsymbol{x_i})$ is L2 normalized, i.e., $||\mathcal{E}(\boldsymbol{x_i})|| = 1$. If we also constrain each column of $\boldsymbol{W}$ to be L2 normalized, then $\boldsymbol{h}_i$ represents the cosine similarity between $\boldsymbol{x_i}$ and each column of $\boldsymbol{W}$. Now, if we apply a non-linearity (say softmax) to $\boldsymbol{h}_i$ and train it (say with categorical cross-entropy loss) to classify the sample to k_i, we are essentially maximizing the cosine similarity between $\mathcal{E}(\boldsymbol{x_i})$ and the k_ith column of $\boldsymbol{W}$, as compared to that between $\mathcal{E}(\boldsymbol{x_i})$ and any other column of $\boldsymbol{W}$. Thus, we can use this dense layer as a clustering function, with columns of $\boldsymbol{W}$ as the centroids. To incorporate the constraint mentioned in Eq. (3), we define the non-linearity as

$$v_{ik} = \frac{e^{\alpha h_{ik}}}{e^{\alpha h_{ik}} + e^{\alpha h_{ik'} - \beta'} + e^{\alpha h_{ik''} - \beta''}} \tag{5}$$

$$\text{where,} \quad k' = \underset{k':k' \neq k, c_{k'} = c_k}{\arg\max} \; h_{jk'} \tag{6}$$

$$k'' = \underset{k'':k'' \neq k, c_{k''} \neq c_k}{\arg\max} \; h_{jk''} \tag{7}$$

$\beta', \beta'' \in \mathbb{R}$ are constants, such that $\beta'' < \beta'$ to penalize the different-class samples more than the same-class samples, and $\alpha \in \mathbb{R}_{>0}$ is a parameter to allow $v_{ik} \subset (0, 1)$ to have values close to either extreme. Without α, $e^{h_{ik}}$ can lie only in $[e^{-1}, e]$, thereby making the clustering ineffective. For implementation, we combine α and L2 normalized $\boldsymbol{W}$ to make $\boldsymbol{W}$ unnormalized. For training the embedding, categorical cross-entropy loss can be used. The above training scheme ensures even sampling over clusters. To further ensure even sampling over classes, we weigh the clusters so that the loss function becomes

$$\mathcal{L}_{\text{wXEnt}} = -\sum_{i,k} \frac{\max_{k'}\{w_{k'}\}}{w_k} \delta(k - k_i) \log v_{ik} \tag{8}$$

$$\text{where,} \quad w_k = \sum_{k'} \delta(c_k - c_{k'}) \tag{9}$$

After training the embedding, we might discard the dense layer (Eq. (4)).

To show the effectiveness of embedding learning, we take a toy example with 1000 samples of $\boldsymbol{x}_i \in \mathbb{R}^{20}$ and 3 classes, blue, green and red, with data distribution 1:1:4, respectively. $\mathcal{E}$ is a single layer LSTM model with 3 nodes, whose output lies on a 3D-sphere. Fig. 1 shows the progress of learning until we get the final embedding where all three classes are separated, with red class embedded into 4 clusters.

3.2. Training the Detector

Once the embedding $\mathcal{E}$ is learned, we can train a simple classifier to detect the events. Since our embedding is a neural network, it is convenient to use a single dense layer to map the embedding features to final detection output $\boldsymbol{y}_i$.

Without discarding the dense layer of Eq. (4), we add a dense layer with L nodes as the output layer with sigmoid non-linearity. It is trained using weighted binary entropy function defined in Eq. (2). We experimented with two ways of training. First, we trained only the output dense layer while keeping the embedding layer fixed. Second, after tuning the output layer, we trained the whole model end-to-end allowing the embedding layers also to be updated. We call these two models as Proposed1 and Proposed2, respectively.

Algorithm 1 Overall Training Algorithm

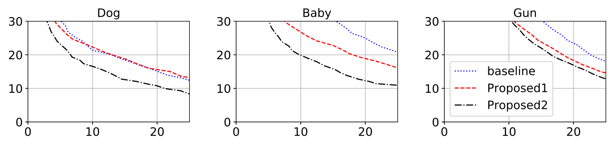
```
1. Learning the Embedding:
Initialize the neural network model $\mathcal{E}$
for a fixed number of iterations, $N_{\text{iter}}$ do
    Cluster $\mathcal{E}(\boldsymbol{x}_i)$ for each class $c$ into $K_c$ clusters
    Add a classification layer (Eq. (4)) to $\mathcal{E}(\boldsymbol{x}_i)$, initialized with cluster centroids
    Tune only the classification layer for 2 epochs (Eq. (8))
    Train end-to-end using Eq. (8) for $N_{\text{epoch}}$ epochs
end for
2. Training the Classifier:
Add an output sigmoid layer of size $L$
Train just the output layer using Eq. (2)
Train end-to-end using Eq. (2)
Select the model with the least validation loss
```

The overall algorithm is outlined in Algorithm 1. Also, Fig. 2 shows the overall architecture of the proposed system, using the cluster and class colors of the toy example of Fig. 1.

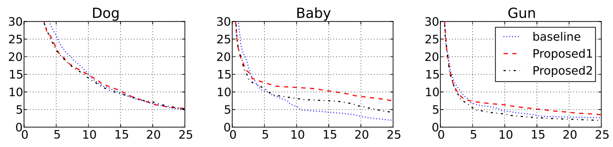
4. EXPERIMENTS

In this paper, we focus on the problem of detecting rare events, where we restrict each audio sample to have either single or no event label present in it. This assumption is also made in the DCASE 2017 challenge task 2 [18]. Our experiments focused on three rare events, viz., dog barking, baby cry and gun shot. As background, we took samples from various other classes. We got the data from AudioSet [16]. Our dataset contained 36.0K samples for background, and 13.5K, 2.3K and 4.1K for dog barking, baby crying and gun shot, respectively. The data ratio for the four classes used for training, namely, dog barking, baby crying, gun shot and background, respectively, was 6:1:2:16. The data was weakly labeled [19], i.e., the start and end times of each event were not given. The samples were partitioned into training (70%), validation (20%) and test (10%) sets.

Each sample had an audio duration of 10s. From each audio sample, we extracted 64 dimensional log mel filter bank energies using a frame length of 25ms and a hop size of 10ms. These features were normalized using the mean and variance of the training data. For embedding $\mathcal{E}$, we used two models - LSTM and CNN - for two sets of experiments. The LSTM model had single LSTM layer with 128 nodes. The CNN model had two layers. The first layer consisted of 32 convolutional 7×7 filters with ReLU non-linearity, followed by batch normalization and a 5×4 max pool, with a drop-out rate



(a) LSTM



(b) CNN

Fig. 3. DET curves for different models with data ratio 6:1:2:16; x-axis is FAR, y-axis is MR (%).

Class	LSTM			CNN		
	B	P1	P2	B	P1	P2
Dog	21.5	19.1	**18.3**	12.0	12.2	**11.8**
Baby	22.4	19.3	**15.5**	**8.0**	11.1	8.5
Gun	17.0	17.1	**13.6**	6.2	6.8	**5.3**
Overall	20.3	18.5	**15.8**	8.7	10.0	**8.5**

Table 1. EER for different models with data ratio 6:1:2:16. Training method B is baseline, P1 is Proposed1 and P2 is Proposed2. Best performance numbers in bold.

of 30%. The second layer had 64 filters of kernel size 7 with ReLU activation, batch normalization, 100×4 max pooling and 30% dropout. Each model was learned using Adam optimizer with an initial learning rate of 0.001. The batch size was fixed to 64 per GPU with 8 GPUs running in parallel. The learning proceeded for $N_{\text{iter}} = 5$, and N_{epoch} was set to 30 in all but last iteration, where it was set as 150. We set $\beta' = 5$ and $\beta'' = 0$. Same setting was used for training the detector, but the training stopped when the loss on the validation data stopped improving for more than 10 epochs.

As baseline, we used the same models, i.e., LSTM or CNN as used for the embedding, followed by dense output layers, but a different training algorithm, viz., the class-weighted loss function (Eq. (2)). The training used the same settings as those used for training the detector for the proposed model.

We evaluated the models by plotting the detection error tradeoff (DET) curves, i.e., the false alarm rate (FAR) vs the miss rate (MR). For both these measures, a lower value indicates a better model. Fig. 3 displays the DET curves for different sizes of the data. As seen in Fig. 3, the Proposed1 method, i.e. without end-to-end tuning, performs comparable to the baseline. Nevertheless, the end-to-end tuning, as in Proposed2, further improves the performance. Also, we tabulate the equal error rate (EER) for each method and for each class

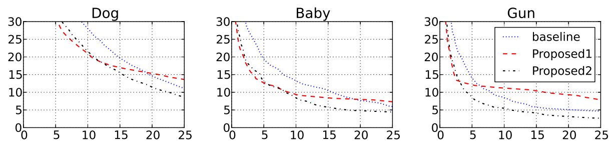


Fig. 4. DET curves for CNN trained with data ratio 2:2:1:26; x-axis is FAR, y-axis is MR (%).

Model	Dog	Baby	Gun	Overall
Baseline	17.3	12.0	9.0	12.8
Proposed1	16.5	9.6	11.1	12.4
Proposed2	**15.2**	**9.2**	**6.8**	**10.4**

Table 2. EER for CNN model trained with different methods with data ratio 2:2:1:26. Best performance numbers in bold.

in Table 1. The overall EER is computed with equal weight to each class. The proposed2 method mostly brings improvements in EER for all the events. Even if the Proposed2 CNN does not do the best for baby-crying, it brings improvement to the overall EER. The CNN model is deeper, has more parameters and is computationally slower as compared to the LSTM model. Clearly, the performance of CNN based embedding is better than the LSTM based embedding.

In another set of experiments, we downsampled the dog barking and gun shot classes in training and validation sets to make the data ratio 2:2:1:26. This data had more imbalance between the background and the classes of interest. The DET curves and EER values are shown in Fig. 4 and Table 2, respectively. Here too, Proposed2 method performs the best. The proposed method divides each class, especially the background in this case, into smaller clusters and enhances the distinguishability amongst classes.

5. CONCLUSION

We presented an embedding-based method to perform classification with data imbalanced over different classes, and applied it to rare audio event detection task. We map the input to an embedding space that ameliorates the data imbalance problem by splitting the large classes into uniformly sized clusters. Similar method has been proposed earlier for image classification task using triple header hinge loss. However, we propose a triple header cross-entropy loss, which makes the proposed scheme computationally faster than the triple header hinge loss based scheme, and is simpler to implement as it integrates with the general back-propagation based learning. The experiments substantiate improvements brought by the proposed method over the popular class-weights based training. Further, the proposed method is fairly general, and can well be applied to other classification problems.

6. REFERENCES

[1] R. Anand, K. G. Mehrotra, C. K. Mohan, and S. Ranka, "An improved algorithm for neural network classification of imbalanced training sets," *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 962–969, 1993.

[2] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[3] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority oversampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[5] J. Wang, M. Xu, H. Wang, and J. Zhang, "Classification of imbalanced data by using the smote algorithm and locally linear embedding," in *International Conference on Signal Processing*. IEEE, 2006, vol. 3.

[6] C. Huang, Y. Li, C. Change Loy, and X. Tang, "Learning deep representation for imbalanced classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5375–5384.

[7] A. Jansen, M. Plakal, R. Pandya, D.P.W. Ellis, S. Hershey, J. Liu, R.C. Moore, and R.A. Saurous, "Unsupervised learning of semantic audio representations," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 126–130.

[8] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M.D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, Oct 2015.

[9] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *18th European Signal Processing Conference (EUSIPCO)*. IEEE, 2010, pp. 1267–1271.

[10] C. V. Cotton and D.P.W. Ellis, "Spectral vs. spectro-temporal features for acoustic event detection," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2011, pp. 69–72.

[11] A. Plinge, R. Grzeszick, and G. A. Fink, "A bag-of-features approach to acoustic event detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3704–3708.

[12] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6440–6444.

[13] S. Hershey and *et al.*, "CNN architectures for large-scale audio classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[14] W. Wang, C.-C. Kao, and C. Wang, "A simple model for detection of rare sound events," in *INTERSPEECH*, 2018.

[15] C.-C. Kao, W. Wang, M. Sun, and C. Wang, "R-crnn: Region-based convolutional recurrent neural network for audio event detection," in *INTERSPEECH*, 2018.

[16] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[17] A. Banerjee, I. Ss Dhillon, J. Ghosh, and S. Sra, "Clustering on the unit hypersphere using von mises-fisher distributions," *Journal of Machine Learning Research*, vol. 6, no. Sep, pp. 1345–1382, 2005.

[18] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proc. Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp. 85–92.

[19] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1038–1047.