Unsupervised Feature Learning Based on Deep Models for Environmental Audio Tagging

Yong Xu, *Member, IEEE*, Qiang Huang, *Member, IEEE*, Wenwu Wang, *Senior Member, IEEE*, Peter Foster, Siddharth Sigtia, Philip J. B. Jackson, and Mark D. Plumbley, *Fellow, IEEE*

***Abstract*—Environmental audio tagging aims to predict only the presence or absence of certain acoustic events in the interested acoustic scene. In this paper, we make contributions to audio tagging in two parts, respectively, acoustic modeling and feature learning. We propose to use a shrinking deep neural network (DNN) framework incorporating unsupervised feature learning to handle the multilabel classification task. For the acoustic modeling, a large set of contextual frames of the chunk are fed into the DNN to perform a multilabel classification for the expected tags, considering that only chunk (or utterance) level rather than frame-level labels are available. Dropout and background noise aware training are also adopted to improve the generalization capability of the DNNs. For the unsupervised feature learning, we propose to use a symmetric or asymmetric deep denoising auto-encoder (syDAE or asyDAE) to generate new data-driven features from the logarithmic Mel-filter banks features. The new features, which are smoothed against background noise and more compact with contextual information, can further improve the performance of the DNN baseline. Compared with the standard Gaussian mixture model baseline of the DCASE 2016 audio tagging challenge, our proposed method obtains a significant equal error rate (EER) reduction from 0.21 to 0.13 on the development set. The proposed asyDAE system can get a relative 6.7% EER reduction compared with the strong DNN baseline on the development set. Finally, the results also show that our approach obtains the state-of-the-art performance with 0.15 EER on the evaluation set of the DCASE 2016 audio tagging task while EER of the first prize of this challenge is 0.17.**

***Index Terms*—DCASE 2016, deep neural networks, deep denoising auto-encoder, environmental audio tagging, unsupervised feature learning.**

I. Introduction

As SMART mobile devices are widely used in recent years, huge amounts of multimedia recordings are generated and uploaded to the web every day. These recordings, such as music, field sounds, broadcast news, and television shows, contain sounds from a wide variety of sources. The demand for analyzing these sounds is increasing, e.g. for automatic audio tagging [1], audio segmentation [2] and audio context classification [3], [4].

For environmental audio tagging, there is a large amount of audio data on-line, e.g. from YouTube or Freesound, which are labeled with tags. How to utilize them, predict them and further add some new tags on the related audio is a challenge. The environmental audio recordings are more complicated than the pure speech or music recordings due to the multiple acoustic sources and incidental background noise. This will make the acoustic modeling more difficult. On the other hand, one acoustic event (or one tag) in environmental audio recordings might occur in several long temporal segments. A compact representation of the contextual information will be desirable in the feature domain.

In traditional methods, a common approach is to convert low-level acoustic features into "bag of audio words" [5]–[9]. K-means, as an unsupervised clustering method, has been widely used in audio analysis [5] and music retrieval [6], [7]. In [8], Cai *et al.* replaced K-means with a spectral clustering-based scheme to segment and cluster the input stream into audio elements. Sainath *et al.* [9] derived an audio segmentation method using Extended Baum-Welch (EBW) transformations for estimating parameters of Gaussian mixtures. Shao *et al.* [7] proposed to use a measure of similarity derived by hidden Markov models to cluster segment of audio streams. Xia *et al.* [10] used Eigenmusic and Adaboost to separate rehearsal recordings into segments, and an alignment process to organize segments. Gaussian mixture model (GMM), as a common model, was also used as the official baseline method in DCASE 2016 for audio tagging [11]. Recently, in [12], a Support Vector Machine (SVM) based Multiple Instance Learning (MIL) system was also presented for audio tagging and event detection. The details of the GMM and SVM methods are presented in the appendix of this paper. However, these methods can not well utilize the contextual information and the potential relationship among different event classes.

The deep learning methods were also investigated for related tasks, like acoustic scene classification [13], acoustic event detection [14] and unsupervised feature learning [15] and better performance could be obtained in these tasks. For music tagging task, [16], [17] have also demonstrated the superiority of deep learning methods. Recently, the deep learning based methods

Manuscript received July 12, 2016; revised November 17, 2016; accepted February 24, 2017. Date of current version May 23, 2017. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of the U.K. under Grant EP/N014111/1. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Juan P. Bello. *(Corresponding author: Yong Xu.)*

Y. Xu, Q. Huang, W. Wang, P. J. B. Jackson, and M. D. Plumbley are with the Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: yx0001@surrey.ac.uk; q.huang@surrey.ac.uk; w.wang@surrey.ac.uk; p.jackson@surrey.ac.uk; m.plumbley@surrey.ac.uk).

P. Foster and S. Sigtia are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, U.K. (e-mail: p.a.foster@qmul.ac.uk; s.s.sigtia@qmul.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2017.2690563

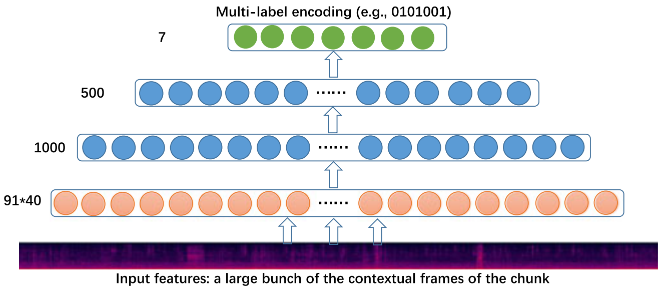
Multi-label encoding (e.g., 0101001)
7
500
1000
91*40
......
Input features: a large bunch of the contextual frames of the chunk

Fig. 1. DNN-based environmental audio tagging framework using the shrinking structure [23].

have also been widely used for environmental audio tagging [18], [19], a newly proposed task in DCASE 2016 challenge [11] based on the CHiME-home dataset [20]. However, it is still not clear what would be appropriate input features, objective functions and the model structures for deep learning based audio tagging. Furthermore, only the chunk-level instead of frame-level labels are available in the audio tagging task. Multiple acoustic events could occur simultaneously with interfering background noise, for example, *child speech* could exist with *TV sound* for several seconds. Hence, a more robust deep learning method is needed to improve the audio tagging performance.

Deep learning was also widely explored in feature learning [21], [22]. These works have demonstrated that data-driven learned features can get better performance than the expert-designed features. In [21], four unsupervised learning algorithms, K-means clustering, restricted Boltzmann machine (RBM), Gaussian mixtures and auto-encoder are explored in image classification. Compared with RBM, auto-encoder is a non-probabilistic feature learning paradigm [22]. For the audio tagging task, Mel-frequency Cepstral Coefficients (MFCCs) and Mel-Filter Banks (MFBs) are commonly adopted as the basic features. However it is not clear whether they are the best choice for audio tagging.

In this paper, we propose a robust deep learning framework for the audio tagging task, with focuses mainly on the following two parts, acoustic modeling and unsupervised feature learning, respectively. For the acoustic modeling, we investigate deep models with shrinking structure, which can be used to reduce the model size, accelerate the training and test process [23]. Dropout [24] and background noise aware training [25] are also adopted to further improve the tagging performance in the DNN-based framework. Different loss functions and different basic features will be also compared for the environmental audio tagging task. For the feature learning, we propose a symmetric or asymmetric deep de-noising auto-encoder (syDAE or asyDAE) based unsupervised method to generate a new feature from the basic features. There are two motivations here, the first is the background noise in the environmental audio recordings which will introduce some mismatch between the training set and the test set. However, the new feature learned by the DAE can mitigate the impact of background noise. The second motivation is that compact representation of the contextual frames is needed for the reason that only chunk-level labels are available. The proposed syDAE or asyDAE can encode the contextual frames into a compact code, which can be used to train a better classifier.

The rest of the paper is organized as follows. We present our robust DNN-based framework in Section II. The proposed deep DAE-based unsupervised feature learning will be presented in Section III. The data description and experimental setup will be given in Section IV. We will show the related results and discussions in Section V, and finally draw a conclusion in Section VI. Appendix will introduce the GMM and SVM based methods in detail, which will be used as baselines for performance comparison in our study.

II. Robust DNN-Based Audio Tagging

DNN is a non-linear multi-layer model for extracting robust features related to a specific classification [26] or regression [27] task. The objective of the audio tagging task is to perform multi-label classification on audio chunks (i.e. assign one or more labels to each audio chunk of a length e.g. four seconds in our experiments). The labels are only available for chunks, but not frames. Multiple events may happen at many particular frames.

A. DNN-Based Multi-Label Classification

Fig. 1 shows the proposed DNN-based audio tagging framework using the shrinking structure, i.e., the hidden layer size is gradually reduced through depth. In [23], it is shown that this structure can reduce the model size, training and test time without losing classification accuracy. Furthermore, this structure can serve as a deep PCA [28] to reduce the redundancy and background noise in the audio recordings. With the proposed framework, a large set of features of the chunk are encoded into a vector with values $\{0, 1\}$. Sigmoid was used as the activation function of the output layer to learn the presence probability of certain events. Rectified linear unit (ReLU) is the activation function for hidden units. Mean squared error (MSE) and binary cross-entropy were adopted and compared as the objective function. As the labels of the audio tagging are binary values, binary cross-entropy can get a faster training and better performance than MSE [29]. A stochastic gradient descent algorithm is performed in mini-batches with multiple epochs to improve learning convergence as follows,

$$E_{mse} = \frac{1}{N}\sum_{n=1}^{N} \|\hat{\mathbf{T}}_n(\mathbf{X}_{n-\tau}^{n+\tau}, \mathbf{W}, \mathbf{b}) - \mathbf{T}_n\|_2^2 \tag{1}$$

$$E_{bce} = -\sum_{n=1}^{N}(\mathbf{T}_n \log\hat{\mathbf{T}}_n + (1-\mathbf{T}_n)\log(1-\hat{\mathbf{T}}_n)) \tag{2}$$

$$\hat{\mathbf{T}}_n = (1+\exp(-\mathbf{O}_n))^{-1} \tag{3}$$

where E_{mse} and E_{bce} are the mean squared error and binary cross-entropy, $\hat{\mathbf{T}}_n(\mathbf{X}_{n-\tau}^{n+\tau}, \mathbf{W}, \mathbf{b})$ and $\mathbf{T}_n$ denote the estimated and reference tag vector at sample index n, respectively, with N representing the mini-batch size, $\mathbf{X}_{n-\tau}^{n+\tau}$ being the input audio feature vector where the window size of context is $2\tau + 1$. It should be noted that the input window size should cover a large set of contextual frames of the chunk considering the fact that the reference tags are in chunk-level rather than frame-level. The weight and bias parameters to be learned are denoted as $(\mathbf{W}, \mathbf{b})$. The DNN linear output is defined as $\mathbf{O}$ before the Sigmoid activation function is applied.

The updated estimate of $\mathbf{W}^\ell$ and $\mathbf{b}^\ell$ in the ℓ-th layer, with a learning rate λ, can be computed iteratively as follows:

$$(\mathbf{W}^\ell, \mathbf{b}^\ell) \leftarrow (\mathbf{W}^\ell, \mathbf{b}^\ell) - \lambda \frac{\partial E}{\partial (\mathbf{W}^\ell, \mathbf{b}^\ell)}, 1 \leq \ell \leq L + 1 \quad (4)$$

where L denotes the total number of hidden layers and the $(L + 1)$-th layer represents the output layer.

During the learning process, the DNN can be regarded as an encoding function, and the audio tags are automatically predicted. The background noise may exist in the audio recordings which may lead to mismatch between the training set and the test set. To address this issue, two additional methods are given below to improve the generalization capability of DNN-based audio tagging. Alternative input features, eg., MFCC and MFB features, are also compared.

B. Dropout for the Over-Fitting Problem

Deep learning architectures have a natural tendency towards over-fitting especially when there is little training data. This audio tagging task only has about four hours training data with imbalanced training data distribution for each type of tag, e.g., much fewer samples for event class ‘b’ compared with other event classes in the DCASE 2016 audio tagging task. Dropout is a simple but effective method to alleviate this problem [24]. In each training iteration, the feature value of every input unit and the activation of every hidden unit are randomly removed with a predefined probability (e.g., ρ). These random perturbations in the input or activations can effectively prevent the DNN from learning spurious feature dependencies. At the testing phase, the DNN scales all of the parameters tuned in the dropout training by $(1 - \rho)$, which is treated as a model averaging process [30].

C. Background Noise Aware Training

Different types of background noise in different recording environments could lead to the mismatch problem between the testing chunks and the training chunks. To alleviate this, we propose a simple background noise aware training (or adaptation) method. To enable this noise adaptation, the DNN is fed with the main audio features appended with an estimate of the background noise. In this way, the DNN can utilize extra on-line information of background noise to better predict the expected tags. The background noise is calculated as follows:

$$\mathbf{V}_n = [\mathbf{Y}_{n-\tau}, \ldots, \mathbf{Y}_{n-1}, \mathbf{Y}_n, \mathbf{Y}_{n+1}, \ldots, \mathbf{Y}_{n+\tau}, \hat{\mathbf{Z}}_n] \quad (5)$$

$$\hat{\mathbf{Z}}_n = \frac{1}{T} \sum_{t=1}^{T} \mathbf{Y}_t \quad (6)$$

where the background noise $\hat{\mathbf{Z}}_n$ is fixed over the chunk and estimated using the first T frames. Although this noise estimator is simple, a similar idea was shown to be effective in DNN-based speech enhancement [25], [27].

D. Alternative Input Features for Audio Tagging

Mel-frequency Cepstral Coefficients (MFCCs) have been used in environmental sound source classification [31], [32], however, some previous work [33], [34] showed that the use of MFCCs is not the best choice as they are sensitive to background noise. Mel-filter bank (MFB) features have already been demonstrated to be better than MFCCs in speech recognition in the DNN framework [35]. However it is not clear whether this is also the case for the audio tagging task using DNN models. Recent studies in audio classification have also shown that accuracy can be boosted by using features that are learned in an unsupervised manner, with examples in the areas of bioacoustics [36] and music [37]. We will study the potential of such methods for audio tagging and present a DAE-based feature learning method in following section.

III. Proposed Deep Asymmetric DAE

MFCCs and MFBs are used as the basic features for the training of DNN-based predictor in this work. MFCCs and MFBs are well-designed features derived by experts based on the human auditory perception mechanism [39]. Recently, more supervised or unsupervised feature learning works have demonstrated that data-driven learned features can offer better performance than the expert-designed features. Neural network based bottleneck feature [40] in speech recognition is one such type of feature, extracted from the middle layer of a DNN classifier. Significant improvement can be obtained after it is fed into a subsequent GMM-HMM (Hidden Markov Model) system and compared with the basic features. However, for the audio tagging task, the tags are weakly labeled and not accurate through the multiple voting scheme. Furthermore, there are lots of related audio files without labels on the web. Hence to use these unlabeled data, we proposed a DAE based unsupervised feature learning method.

Specifically, for environmental audio tagging task, disordered background noise exists in the recordings which may lead to the mismatch between the training set and the test set. DAE-based method can mitigate the effect of background noise and focus on more meaningful acoustic event patterns. Another motivation is that the compact representation of the contextual frames is needed since the labels are in chunk-level rather than frame-level.

An unsupervised feature learning algorithm is used to discover features from the unlabeled data. For this purpose, the unsupervised feature learning algorithm takes the dataset X as input and outputs a new feature vector. In [21], four unsupervised learning algorithms, K-means clustering, restricted Boltzmann machine (RBM), Gaussian mixtures and auto-encoder have been explored in image classification. Among them, RBM and auto-encoder are widely adopted to get new features or pretrain a deep model. Compared with RBM, auto-encoder is a non-probabilistic feature learning scheme [22]. The auto-encoder explicitly has a feature encoding module, called the *encoder*. It also defines another function, denoted as the *decoder*. The encoder and decoder function are represented as f_θ and g_θ, respectively. To force the hidden layers to discover more robust features, the de-noising auto-encoder [38] introduces a

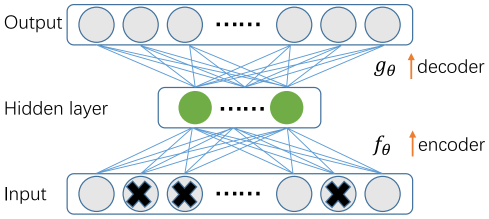

Fig. 2. A typical one hidden layer of de-noising auto-encoder [38] structure with an encoder and a decoder. Some input units are set to zero by the Dropout process (shown by a black cross "X") to train a more robust system.

stochastic corruption process applied to the input layer, which randomly selects some nodes to set their values to zero. Dropout [24] is used here to corrupt the input units. Compared with the auto-encoder, DAE can detect more robust features and prevent it from simply learning the identity function [38]. Fig. 2 shows a typical one hidden layer of DAE structure with an encoder and a decoder. The encoder produces a new feature vector $\mathbf{h}$ from an input $\mathbf{x} = x^{(1)}, \ldots, x^{(T)}$. It is defined as,

$$\mathbf{h} = f_\theta(\tilde{\mathbf{x}}) = s_f(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}) \tag{7}$$

where $\mathbf{h}$ is the new feature vector or new representation or code [22] of the input data $\mathbf{x}$ with the corrupted version $\tilde{\mathbf{x}}$. s_f is the non-linear activation function. $\mathbf{W}$ and $\mathbf{b}$ denote the weights and bias of the encoder, respectively. On the other hand, the decoder, g_θ can transfer the new feature representation back to the original feature space, namely producing a reconstruction $\hat{\mathbf{x}} = g_\theta(\mathbf{h})$.

$$\hat{\mathbf{x}} = g_\theta(\mathbf{h}) = s_g(\mathbf{W}'\mathbf{h} + \mathbf{b}') \tag{8}$$

where $\hat{\mathbf{x}}$ is the reconstructed feature which is the estimation of the input feature. s_g is the non-linear activation function of the decoder. $\mathbf{W}'$ and $\mathbf{b}'$ denote the weights and bias of the decoder. Here $\mathbf{W}$ and $\mathbf{W}'$ are not tied, namely $\mathbf{W}' \neq \mathbf{W}^T$. The set of parameters $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{W}', \mathbf{b}'\}$ of the auto-encoder are updated to get the lowest reconstruction error $L(\mathbf{x}, \hat{\mathbf{x}})$, which is a measure of the Euclidean distance between the input $\mathbf{x}$ and the output. $\hat{\mathbf{x}}$. The general loss function for the de-noising auto-encoder [38] training can be defined as,

$$\Gamma_{AE}(\theta) = \sum_t L(x^{(t)}, g_\theta(f_\theta(\tilde{x}^{(t)}))) \tag{9}$$

Furthermore, the DAE can be stacked to obtain a deep DAE. The DAE is actually an advanced PCA with the non-linear activation functions [28].

In Fig. 3, the framework of deep asymmetric DAE (asyDAE) based unsupervised feature learning for audio tagging is presented. It is a deep DAE stacked by simple DAE with random initialization. To utilize the contextual information, multiple frames MFB features are fed into the deep DAE. A typical DAE is a symmetric structure (syDAE) with the same size as the input. However here the deep DAE is only designed to predict the middle frame feature. This is because the more predictions in the output means the more memory needed in the bottleneck layer. In our practice, the deep DAE would generate a larger reconstruction error if multiple frames features were designed as the output with a narrow bottleneck layer. This leads to an inaccurate representation of the original feature in a new space. Nonetheless, with only the middle frame features in the output, the reconstruction error is smaller. Fig. 4 plots the reconstruction error between the asyDAE and syDAE for the example shown in Section IV. However, we will show the performance difference between deep asyDAE and deep syDAE later in Section V. The default size of the bottleneck code is 50 and 200 for asyDAE and syDAE, respectively. For syDAE, there is a trade-off when setting the bottleneck code size, to avoid the high input dimension for the back-end DNN classifier, as well as for reconstructing the multiple-frame output. Typically, the weights between the encoder and the decoder are tied. Here we set them to be untied to retain more contextual information in the bottleneck codes. More specifically, the input frame number in the DAE input layer is chosen as seven for the reason that 91-frame expansion will be used in the back-end DNN classifier. In addition, larger frame expansion in DAE is more difficult to encode into a fixed bottleneck code.

As the output of DAE is a real-valued feature, MSE was adopted as the objective function to fine-tune the whole deep DAE model. A stochastic gradient descent algorithm is performed in mini-batches with multiple epochs to improve learning convergence as follows,

$$Er = \frac{1}{N}\sum_{n=1}^{N} \|\hat{\mathbf{X}}_n(\mathbf{X}_{n-\tau}^{n+\tau}, \mathbf{W}, \mathbf{b}) - \mathbf{X}_n\|_2^2 \tag{10}$$

where Er is the mean squared error, $\hat{\mathbf{X}}_n(\mathbf{X}_{n-\tau}^{n+\tau}, \mathbf{W}, \mathbf{b})$ and $\mathbf{X}_n$ denote the reconstructed and input feature vector at sample index n, respectively, with N representing the mini-batch size, $\mathbf{X}_{n-\tau}^{n+\tau}$ being the input audio feature vector where the window size is $2\tau + 1$. $(\mathbf{W}, \mathbf{b})$ denote the weight and bias parameters to be learned.

The activation function of the bottleneck layer is another key point in the proposed deep DAE based unsupervised feature learning framework. Sigmoid is not suitable to be used as the activation function of the code layer, since it compresses the value of the new feature into a range [0, 1] which will reduce its representation capability. Hence, Linear or ReLU activation function is a more suitable choice. In [28], the activation function of the units of the bottleneck layer or the code layer of the deep DAE is linear. A perfect reconstruction of the image can be obtained. In this work, ReLU and Linear activation functions of the bottleneck layer are both verified to reconstruct the audio features in the deep auto-encoder framework. Note that all of the other layer units also adopt ReLU as the activation function.

In summary, the new feature derived from the bottleneck layer of the deep auto-encoder can be regarded as the optimized feature due to three factors. The first one is that the DAE learned feature is generated from contextual input frames with new compact representations. This kind of features are better for capturing the temporal structure information compared with the original feature. The second advantage is that the deep de-noising AE based unsupervised feature learning can smooth the disordered background noise in the audio recordings to alleviate the mismatch problem between the training set and test set.

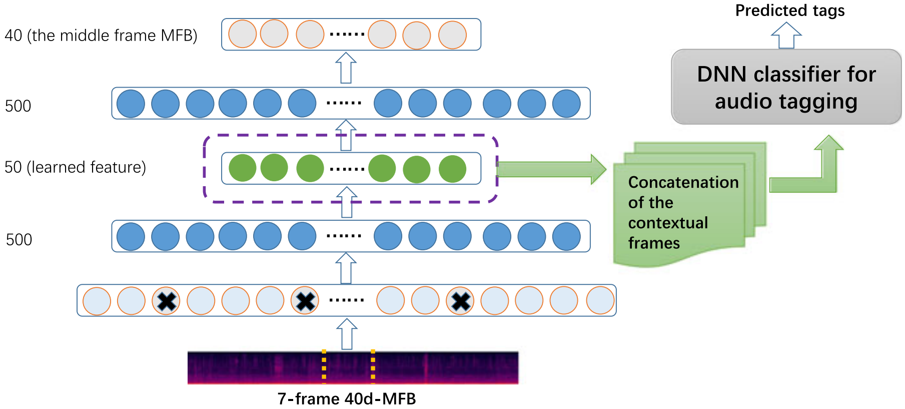
Predicted tags
DNN classifier for audio tagging
40 (the middle frame MFB)
500
50 (learned feature)
500
Concatenation of the contextual frames
7-frame 40d-MFB

Fig. 3. The framework of deep asymmetric DAE (asyDAE) based unsupervised feature learning for audio tagging. The weights between the encoder and the decoder are untied to retain more contextual information into the bottleneck layer (shown in the dashed rectangle).

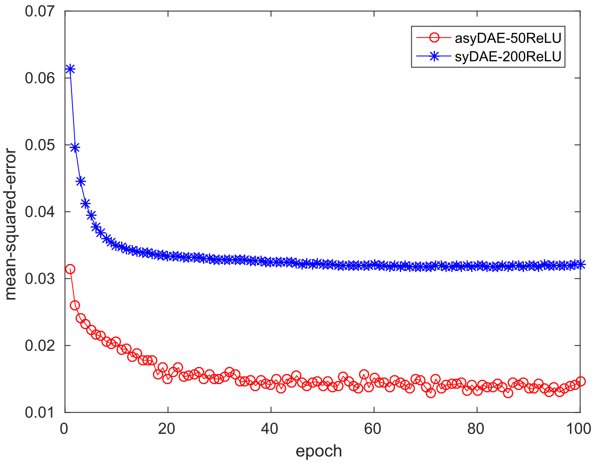

Fig. 4. The reconstruction error over the CV set of the asymmetric DAE with 50 ReLU units in the bottleneck layer (denoted as asyDAE-50ReLU) and the symmetric DAE with 200 ReLU units in the bottleneck layer (denoted as syDAE-200ReLU).

Finally, with this framework, the large amount of unlabeled data could be utilized and more statistical knowledge in the feature space can be learned.

IV. Data Description and Experimental Setup

A. DCASE2016 Data Set for Audio Tagging

The data that we used for evaluation is the dataset of Task 4 of the DCASE 2016 [11], which is built based on the CHiME-home dataset [20]. The audio recordings were made in a domestic environment [41]. Prominent sound sources in the acoustic environment are two adults and two children, television and electronic gadgets, kitchen appliances, footsteps and knocks produced by human activity, in addition to sound originating from outside the house [41]. The audio data are provided as 4-second chunks at two sampling rates (48 kHz and 16 kHz) with the 48 kHz data in stereo and the 16 kHz data in mono. The 16 kHz recordings were obtained by down-sampling the right channel of the 48 kHz recordings. Note that Task 4 of the DCASE 2016 challenge is based on using only 16 kHz recordings.

TABLE I
Labels Used in Annotations

Label	Description
b	Broadband noise
c	Child speech
f	Adult female speech
m	Adult male speech
o	Other identifiable sounds
p	Percussive sounds, e.g. crash, bang, knock, footsteps
v	Video game/TV

For each chunk, multi-label annotations were first obtained from each of 3 annotators. There are 4378 such chunks available, referred to as *CHiME-Home-raw* [20]; discrepancies between annotators are resolved by conducting a majority vote for each label. The annotations are based on a set of 7 label classes as shown in Table I. A detailed description of the annotation procedure is provided in [20]. To reduce uncertainty about annotations, evaluations are based on considering only those chunks where 2 or more annotators agreed about label presence across label classes. There are 1946 such chunks available, referred to as *CHiME-Home-refined* [20]. Another 816 refined chunks are kept for the final evaluation set of Task 4 of the DCASE 2016 challenge.

B. Experimental Setup

In our experiments, following the original specification of Task 4 of the DCASE 2016 [11], we use the same five folds from the given development dataset, and use the remaining audio recordings for training. Table II lists the number of chunks of training and test data used for each fold and also the final evaluation setup.

To keep the same feature configurations as in the DCASE 2016 baseline system, we pre-process each audio chunk by segmenting them using a 20ms sliding window with a 10 ms hop size, and converting each segment into 24-dimension MFCCs and 40-dimension logarithmic MFBs. For each 4-second chunk, 399 frames of MFCCs are obtained. A large set of frames

TABLE II
THE NUMBER OF AUDIO CHUNKS FOR TRAINING AND TEST FOR THE DEVELOPMENT SET AND THE FINAL EVALUATION SET

Fold index	#Training	#Test
0	4004	383
1	3945	442
2	3942	463
3	4116	271
4	4000	387
Evaluation set	4387	816

expansion is used as the input of the DNN. The impact of the number of frame expansion on the performance will be evaluated in the following experiments. Hence the input size of DNN was the number of expanded frames plus the appended background noise vector. All of the input features are normalized into zero-mean and unit-variance. The first hidden layer with 1000 units and the second with 500 units were used to construct a shrinking structure [23]. The 1000 or 500 hidden units are a common choice in DNNs [42]. Seven sigmoid outputs were adopted to predict the seven tags. The learning rate was 0.005. The momentum was set to be 0.9. The dropout rates for input layer and hidden layer were 0.1 and 0.2, respectively. The mini-batch size was 100. T in Equation (6) was 6. In addition to the *CHiME-Home-refined* set [20] with 1946 chunks, the remaining 2432 chunks in the *CHiME-Home-raw* set [20] without 'strong agreement' labels in the development dataset were also added into the DNN training considering that DNN has a better fault-tolerant capability. Meanwhile, these 2432 chunks without 'strong agreement' labels were also added into the training data for GMM and SVM training. The deep asyDAE or deep asyDAE has 5 layers with 3 hidden layers. For asyDAE, the input is 7-frame MFBs, and the output is the middle frame MFB. The first and third hidden layer both have 500 hidden units while the middle layer is the bottleneck layer with 50 units. For syDAE, the output is 7-frame MFBs, and the middle layer is the bottleneck layer with 200 units. The dropout level for the asyDAE or syDAE is set to be 0.1. The final DAE models are trained at epoch 100.

For performance evaluation, we use equal error rate (EER) as the main metric which is also suggested by the DCASE 2016 audio tagging challenge. EER is defined as the point of the graph of false negative (FN) rate versus false positive (FP) rate [43]. The number of true positives is denoted as TP. EERs are computed individually for each evaluation fold, and we then average the obtained EERs across the five folds to get the final performance. Precision, Recall and F-score are also adopted to evaluate the performance among different systems.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{11}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

$$\text{F-score} = \frac{2\text{Precision}\cdot\text{Recall}}{\text{Precision}} + \text{Recall} \tag{13}$$

All the source codes for this paper and pre-trained models can be downloaded at Github website.[1] The codes for the SVM and GMM baselines are also uploaded at the same website.

C. Compared Methods

For a comparison, we also ran two baselines using GMMs and the SVMs mentioned in the Appendix section. For the GMM-based method, the number of mixture components is 8 which is a default configuration of the DCASE 2016 challenge. The sliding window and hop size set for the two baselines and our proposed methods are all the same. Additionally, we also use chunk-level features to evaluate on SVM-based method according to [44]. The mean and covariance of the MFCCs over the duration of the chunk can describe the Gaussian with the maximum likelihood [44]. Hence those statistics can also be unwrapped into a vector as a chunk-level feature to train the SVM. To handle audio tagging with SVM, each audio recording will be viewed as a bag. To accelerate computation, we use linear kernel function in our experiments.

We also compared our methods with the state-of-the-art methods. Lidy-CQT-CNN [18], Cakir-MFCC-CNN [19] and Yun-MFCC-GMM [45] are the first, second and third prize of the audio tagging task of the DCASE2016 challenge [11]. The former two methods used convolutional neural networks (CNN) as the classifier. Yun-MFCC-GMM [45] adopted the discriminative training method on GMMs.

V. RESULTS AND DISCUSSIONS

In this section, the overall evaluations on the development set and the evaluation set of the DCASE 2016 audio tagging task will be firstly presented. Then several evaluations on the parameters of the models will be given.

A. Overall Evaluations

Table III shows the EER comparisons on seven labels among the proposed asyDAE-DNN, syDAE-DNN, DNN baseline trained on MFB, DNN baseline trained on MFCC methods, Yun-MFCC-GMM [45], Cakir-MFCC-CNN [19], Lidy-CQT-CNN [18], SVM trained on chunks, SVM trained on frames and GMM methods [11], which are evaluated on the development set and the evaluation set of the DCASE 2016 audio tagging challenge. On the development set, it is clear that the proposed DNN-based approaches outperform the SVM and GMM baselines across the five-fold evaluations. GMM is better than the SVM methods. SVM performs worse on the audio event 'b' where less training samples are included in the imbalanced development set compared with other audio events [11]. However, the GMM and DNN methods perform better on the audio event 'b' with lower EERs. The frame-level SVM is superior to the chunk-level SVM. This is because the audio tagging is a multi-label classification task rather than a single-label classification task while the statistical mean value in the chunk-SVM will make the feature indistinct among different labels in the same

[1] https://github.com/yongxuUSTC/aDAE_DNN_audio_tagging

TABLE III
EER Comparisons on Seven Labels Among the Proposed asyDAE-DNN, syDAE-DNN, DNN Baseline Trained on MFB, DNN Baseline Trained on MFCC Methods, Yun-MFCC-GMM [45], Cakir-MFCC-CNN [19], Lidy-CQT-CNN [18], SVM Trained on Chunks, SVM Trained on Frames and GMM Methods [11], Which are Evaluated on the Development Set and the Evaluation Set

Tags	b	c	f	m	o	p	v	Average
				Development Set				
GMM (DCASE baseline) [11]	0.074	0.225	0.289	0.269	0.290	0.248	0.050	0.206
Chunk-SVM	0.464	0.438	0.430	0.470	0.524	0.518	0.274	0.445
Frame-SVM	0.205	0.199	0.284	0.390	0.361	0.308	0.090	0.263
Yun-MFCC-GMM [45]	0.074	0.165	0.249	0.216	0.278	0.210	0.039	0.176
Cakir-MFCC-CNN [19]	0.070	0.210	0.250	0.150	0.260	0.210	0.050	0.171
Lidy-CQT-CNN * [18]	–	–	–	–	–	–	–	–
MFCC-DNN	0.078	0.145	0.230	0.126	0.268	0.183	0.029	0.151
MFB-DNN	**0.067**	0.142	0.206	0.102	0.256	0.148	0.025	0.135
Proposed syDAE-DNN	0.068	0.134	0.206	**0.087**	0.238	0.146	**0.023**	0.129
Proposed asyDAE-DNN	**0.067**	**0.124**	**0.202**	0.092	**0.231**	**0.143**	0.023	**0.126**
				Evaluation Set				
GMM (DCASE baseline) [11]	0.117	0.191	0.314	0.326	0.249	0.212	0.056	0.209
Chunk-SVM	0.032	0.385	0.407	0.472	0.536	0.506	0.473	0.402
Frame-SVM	0.129	0.166	0.241	0.353	0.336	0.268	0.093	0.227
Yun-MFCC-GMM [45]	0.032	**0.177**	**0.179**	0.253	0.266	0.207	0.102	0.174
Cakir-MFCC-CNN [19]	0.022	0.25	0.25	0.159	0.258	0.208	0.027	0.168
Lidy-CQT-CNN [18]	0.032	0.21	0.214	0.182	0.32	0.168	0.035	0.166
MFCC-DNN	0.032	0.204	0.21	0.209	0.288	0.194	0.039	0.168
MFB-DNN	0.032	0.184	0.204	0.172	0.272	0.179	0.053	0.157
Proposed syDAE-DNN	0.023	0.184	0.203	0.165	0.280	**0.174**	0.041	0.153
Proposed asyDAE-DNN	**0.014**	0.210	0.207	**0.149**	**0.256**	0.175	**0.022**	**0.148**

*Lidy-CQT-CNN [18] did not measure the EER results on the development set.

chunk. Compared with the DNN methods, SVM and GMM are less effective in utilizing the contextual information and the potential relationship among different tags. Note that the DNN models here are all trained using binary cross-entropy defined in Eq. (2) as the loss function. Binary cross-entropy is found better than the mean squared error for training the audio tagging models which will be shown in the following subsection. The DNN trained on MFCCs is worse than the DNN trained on MFBs with the reduced EER from 0.151 to 0.135, especially on the percussive sounds ('p'), e.g. crash, bang, knock and footsteps. This result is consistent with the observations in speech recognition using DNNs [35]. Compared with MFBs, MFCCs lost some information after the discrete cosine transformation (DCT) step. The bottleneck code size for asyDAE-DNN and syDAE-DNN here is 50 and 200, respectively. It is found that asyDAE-DNN can reduce the EER from 0.157 to 0.148 compared with MFB-DNN-baseline, especially on tag 'c' and tag 'o'. The asyDAE-DNN method is slightly better than the syDAE-DNN because the syDAE-DNN should have a larger bottleneck code to reconstruct the seven-frame output. However, the large size of the bottleneck code in syDAE-DNN will make the input dimension of the back-end DNN classifier very high. Lidy-CQT-CNN [18] did not measure the EER on the development set [18]. Our proposed DNN methods can get better performance than Cakir-MFCC-CNN [19] and Yun-MFCC-GMM [45].

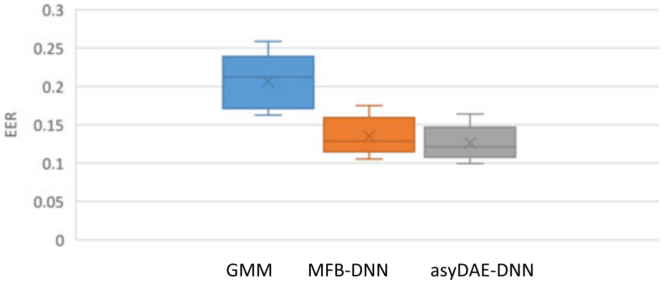

Fig. 5. The box-and-whisker plot of EERs, among the GMM baseline, Mel-Filter bank (MFB)-DNN baseline and asymmetric DAE (asyDAE)-DNN method, across five standard folds on the development set.

Fig. 5 shows the box-and-whisker plot of EERs, among the GMM baseline, MFB-DNN baseline and asyDAE-DNN method, across five standard folds on the development set of the DCASE 2016 audio tagging challenge. It can be found that the asyDAE-DNN is consistently better than the MFB-DNN baseline. To test the statistical significance between the asyDAE-DNN and MFB-DNN baseline, we use the paired-sample t-test [46] tool in MATLAB. The audio tagging task of the DCASE2016 challenge has five standard folds with seven tags. Hence, a 35-dimension vector can be obtained for each method, then the paired-sample t-test tool can be used to calculate the p-value. Its results indicate that t-test rejects the null hypothesis at the 1% significance level. It was found that the p-value is $\ll 0.01$ in this test which indicates that the improvement is statistically significant. Finally our proposed method can get a 38.9% relative EER reduction compared with the GMM baseline of the DCASE 2016 audio tagging challenge on the development set.

Table III also presents EER comparisons on the evaluation set. Note that the final evaluation set was not used for any training which means syDAE and asyDAE also did not use it in the training. It can be found that our proposed asyDAE-DNN can

TABLE IV
PRECISION, RECALL AND SCORE COMPARISONS BETWEEN THE MFB-DNN BASELINE AND THE ASYDAE-DNN METHOD, WHICH ARE EVALUATED FOR SEVEN TAGS ON THE FINAL EVALUATIONS SET OF THE DCASE2016 AUDIO TAGGING TASK

Evaluation set		b	c	f	m	o	p	v	Average
Precision	MFB-DNN	1.000	**0.684**	0.676	0.464	0.583	0.774	0.980	0.737
	asyDAE-DNN	1.000	0.654	**0.691**	**0.474**	**0.692**	**0.819**	**0.991**	**0.760**
Recall	MFB-DNN	0.968	0.905	**0.507**	0.405	0.224	**0.677**	**0.976**	0.666
	asyDAE-DNN	0.968	**0.912**	0.464	**0.456**	**0.288**	0.658	0.973	**0.674**
F-score	MFB-DNN	0.984	**0.780**	**0.580**	0.432	0.324	0.722	0.978	0.686
	asyDAE-DNN	0.984	0.762	0.556	**0.465**	**0.407**	**0.730**	**0.982**	**0.698**

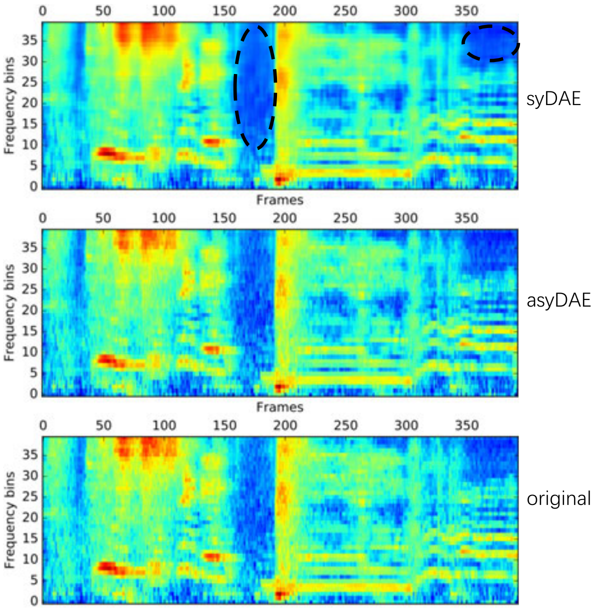

Fig. 6. Spectrograms of the reconstructed Mel-Filter Banks (MFBs) by the deep asymmetric DAE (asyDAE) and deep symmetric DAE (syDAE), and also the original MFBs. The dotted ovals indicate the smoothed parts on the reconstructed MFBs. The Y-axis is the frequency bin and the X-axis is the frame number.

get the state-of-the-art performance. Our MFB-DNN is a strong baseline through the use of several techniques, e.g., the dropout, background noise aware training, shrinking structure and also binary cross-entropy. The proposed asyDAE-DNN can get a 5.7% relative improvement compared with the MFB-DNN baseline. syDAE-DNN did not show improvement over the MFB-DNN because syDAE-DNN with the bottleneck code size 200 can not well reconstruct the unseen evaluation set. However the asyDAE-DNN with the bottleneck code size 50 can well reconstruct the unseen evaluation set. Finally, our proposed methods can get the state-of-the-art performance with 0.148 EER on the evaluation set of the DCASE 2016 audio tagging challenge. Another interesting result here is that Yun-MFCC-GMM [45] performs well on tag ‘c’ and tag ‘f’ where high pitch information exists. It would be interesting to fuse their prediction posteriors together in our future work.

To give a further comparison between the MFB-DNN baseline and the asyDAE-DNN method, Table IV shows precision, recall and score comparisons evaluated for seven tags on the final evaluation set of the DCASE2016 audio tagging task. As the DNN prediction belongs to [0, 1], a threshold 0.4 is set to judge whether it is a hit or not. Using asyDAE-DNN better performance than the MFB-DNN baseline can be obtained on most of the three measures. One interesting result is that DNN method can get a quite high score on tag ‘b’ although there are only few training samples in the training set [11].

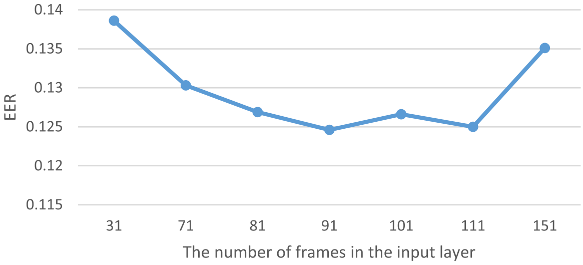

Fig. 7. EERs on Fold 1 of the development set evaluated using different number of frame expansions in the input layer of the MFB-DNN.

Fig. 6 shows the spectrograms of the original MFBs and the reconstructed MFBs by the deep syDAE and deep asyDAE. Both of them can reconstruct the original MFBs well while syDAE got a smoother reconstruction. There is background noise in original MFBs which will lead to the mismatch problem mentioned earlier. syDAE can well reduce the background noise shown in the dashed ellipses with the risk of losing the important spectral information. However, asyDAE can be a trade-off between background noise smoothing and signal reconstruction. On the other hand, the weights of the encoder and decoder in the deep asyDAE and the deep syDAE are not typically tied. In this way, more contextual information is encoded into the bottleneck layer to get a compact representation, which is helpful for the audio tagging task considering the fact that the reference labels are in chunk-level.

B. Evaluations for the Number of Contextual Frames in the Input of the DNN Classifier

The reference label information for this audio tagging task is on the utterance-level rather than the frame-level, and the occurring orders and frequencies of the tags are unknown. Hence, it is important to use a large set of the contextual frames in the input of the DNN classifier. However, the dimension of the input layer of the DNN classifier will be too high and the number of training samples would be reduced if the number of the frame

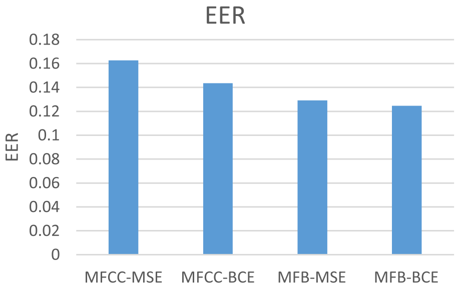

Fig. 8. EERs on Fold 1 of the development set evaluated using different features, namely MFCCs and Mel-Filter Banks (MFBs), different loss functions, namely mean squared error (MSE) and binary cross entropy (BCE).

expansion is too large. Larger input size will increase the complexity of the DNN model and as a result, some information could be lost during the feed-forward process considering that the hidden unit size is fixed to be 1000 or 500. Fewer training samples will make the training process of DNN unstable considering that the parameters are updated using a stochastic gradient descent algorithm performed in mini-batches.

Fig. 7 shows the EERs for Fold 1 evaluated by using different number of contextual frames in the input of the DNN classifier. Here the MFBs are used as the input features. It can be found that using the 91-frame MFBs as the input gives the lowest EER. As mentioned in the experimental setup, the window size of each frame is 20 ms with 50% hop size. 91-frame expansion means that the input length is about one second. However, the length of the whole chunk is 4 seconds, so it indicates that most of the tags occurred several times and overlap with each other heavily in the certain chunk. Meanwhile, 91-frame expansion in the input layer of the DNN is a good trade-off among the contextual information, input size, and total training samples.

C. Evaluations for Different Kinds of Input Features and Different Types of Loss Functions

Fig. 8 shows EERs on Fold 1 evaluated using different features, namely MFCCs and MFBs, different loss functions, namely mean squared error (MSE) and binary cross entropy (BCE). It can be found that MFBs perform better than MFCCs. MFBs contain more spectral information than the MFCCs. BCE is superior to MSE considering that the value of label is binary, either zero or one. MSE is more suitable to fit the real values.

D. Evaluations for Different Bottleneck Size of DAE and Comparison With the Common Auto-Encoder

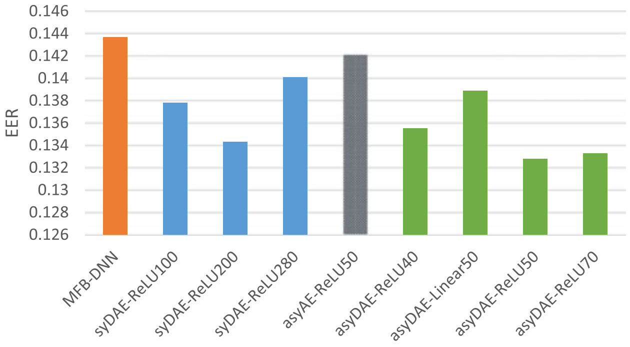

Fig. 9. EERs on Fold 0 of the development set evaluated using different de-noising auto-encoder configurations and compared with the MFB-DNN baseline. syDAE-ReLU200 means the symmetric DAE with 200 ReLU units in the bottleneck layer. asyDAE-Linear50 means the asymmetric DAE with 50 linear units in the bottleneck layer. aAE-ReLU50 denotes the asymmetric auto-encoder without de-noising (or dropout).

TABLE V
EERs for Fold 1 Across Seven Tags Using DNNs and GMMs Trained on the 'CHiME-Home-Raw' Set and 'CHiME-Home-Refined' Set

Dataset	b	c	f	m	o	p	v
DNN-Refine	0.009	0.168	0.223	0.158	**0.273**	0.118	0.050
DNN-Raw	**0.002**	**0.124**	**0.209**	**0.146**	0.277	**0.089**	**0.025**
GMM-Refine	**0.000**	**0.203**	0.343	0.303	**0.305**	**0.333**	0.154
GMM-Raw	0.013	0.283	**0.294**	**0.217**	0.326	0.347	**0.051**

Fig. 9 shows the EERs on Fold 0 evaluated using different de-noising auto-encoder configurations and compared with the MFB-DNN baseline. For the deep syDAE, the bottleneck layer size needs to be properly set. If it is too small, the 7-frame MFBs can not be well reconstructed. While it will increase the input size of the DNN classifier if the bottleneck code is too large. For the deep asyDAE, the bottleneck layer with 50 ReLU units is found empirically to be a good choice. The linear unit (denoted as asyDAE-Linear50) is worse than the ReLU unit for the new feature representation. Another interesting result is that the performance was almost the same if there is no de-noising (or dropout) operation (denoted as aAE-ReLU50) in the ordinary auto-encoder. The reason is that the baseline DNN is well trained on MFBs with the binary cross-entropy as the loss function.

E. Evaluations for the Size of the Training Dataset

In the preceding experiments, 'CHiME-Home-raw' dataset was used to train the DNN, GMM and SVM models. Here, to evaluate the performance using different training data sizes, DNNs were trained based on 'CHiME-Home-raw' or 'CHiME-Home-refined' alternatively while keeping the same testing set. MFBs were used as the input features for the DNN classifier.

Table V shows the EERs for Fold 1 across seven tags with the DNNs trained on the 'CHiME-Home-raw' set and 'CHiME-Home-refined' set. It can be clearly found that the DNN trained on the 'CHiME-Home-raw' set is better than the DNN trained on the 'CHiME-Home-refined' set, although part of the labels of the 'CHiME-Home-raw' set are not accurate. This indicates that DNN has fault-tolerant capability which suggests that the labels for the tags can not be refined with much annotators' effort. The size of the training set is crucial for the DNN training. Nonetheless the GMM method is sensitive to the inaccurate labels. The increased training data with inaccurate tag labels does not help to improve the performance of GMMs.

F. Further Discussions on the Deep Auto-Encoder Features

Fig. 10 presents the audio spectrogram of the deep asyDAE features, which can be regarded as the new non-negative

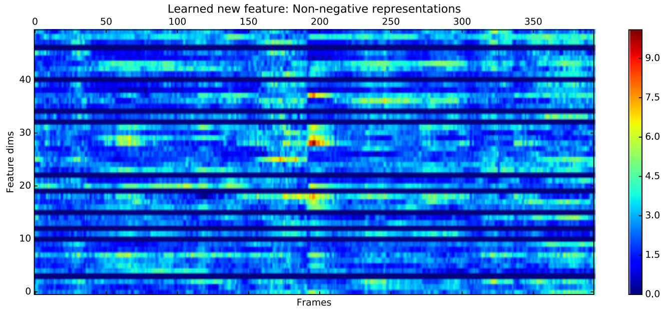

Fig. 10. The audio spectrogram of the deep asymmetric DAE (asyDAE) features with the non-negative representation.

representation or optimized feature of the original MFBs. The units of the bottleneck layer in the deep asyDAE are all activated by the ReLU functions as mentioned in Sec. III. Hence, the values of the learned feature are all non-negative, leading to a non-negative representation of the original MFBs. Such a non-negative representation can then be multiplied with the weights in the decoding part of the DAE to obtain the reconstructed MFBs. It is also adopted to replace the MFBs as the input to the DNN classifier to make a better prediction for the tags. The pure blue area at some dimensions in Fig. 10 indicates the zero values in the ReLU activation function.

VI. Conclusion

In this paper we have studied the acoustic modeling and feature learning issues in audio tagging. We have proposed a DNN incorporating unsupervised feature learning based approach to handle audio tagging with weak labels, in the sense that only the chunk-level instead of the frame-level labels are available. A deep asymmetric DAE with untied weights based unsupervised feature learning was also proposed to generate a new feature with non-negative representations. The DAE can generate smoothed feature against the disordered background noise and also give a compact representation of the contextual frames. We tested our approach on the dataset of the Task 4 of the DCASE 2016 challenge, and obtained significant improvements over the two baselines, namely GMM and SVM. The proposed unsupervised feature learning method can get a relative 6.7% EER reduction compared with the strong DNN baseline on the development set. We also get the state-of-the-art performance with 0.148 EER on the evaluation set compared with the latest results [45], [19], [18] from the DCASE 2016 challenge. For the future work, we will use convolutional neural network (CNN) to extract more robust high-level features for the audio tagging task. Larger dataset, such as Yahoo Flickr Creative Commons 100 Million (YFCC100m) dataset [47] and YouTube-8M dataset [48] will be considered to further evaluate the proposed algorithms.

Appendix

Two baseline methods compared in our work are briefly summarized below.

A. Audio Tagging Using Gaussian Mixture Models

GMMs are a commonly used generative classifier. To implement multi-label classification with simple event tags, a binary classifier is built associating with each audio event class in the training step. For a specific event class, all audio frames in an audio chunk labeled with this event are categorized into a positive class, whereas the remaining features are categorized into a negative class. On the classification stage, given an audio chunk C_i, the likelihoods of each audio frame $x_{ij}, (j \in \{1 \cdots L_{C_i}\})$ are calculated for the two class models, respectively. Given audio event class k and chunk C_i, the classification score $S_{C_{ik}}$ is obtained as log-likelihood ratio:

$$S_{C_{ik}} = \sum_j \log(f(x_{ij}, \Theta_{\text{pos}})) - \sum_j \log(f(x_{ij}, \Theta_{\text{neg}})) \quad (14)$$

B. Audio Tagging Using Multiple Instance SVM

Multiple instance learning is described in terms of bags $\mathbf{B}$. The jth instance in the ith bag, B_i, is defined as x_{ij} where $j \in I = \{1 \cdots l_i\}$, and l_i is the number of instances in B_i. B_i's label is $Y_i \in \{-1, 1\}$. If $Y_i = -1$, then $x_{ij} = -1$ for all j. If $Y_i = 1$, then at least one instance $x_{ij} \in B_i$ is a positive example of the underlying concept [49].

As MI-SVM is the bag-level MIL support vector machine to maximize the bag margin, we define the functional margin of a bag with respect to a hyper-plane as:

$$\gamma_i = Y_i \max_{j \in I}(\langle \mathbf{w}, \mathbf{x}_{ij} \rangle + b) \quad (15)$$

Using the above notion, MI-SVM can be defined as:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2}\|\mathbf{w}^2\| + A \sum_i \xi_i$$

$$\text{subject to}: \quad \forall_{\mathrm{i}} : \gamma_{\mathrm{i}} \geq 1 - \xi_{\mathrm{i}}, \ \ \xi_{\mathrm{i}} \geq 0 \quad (16)$$

where $\mathbf{w}$ is the weight vector, b is bias, ξ is margin violation, and A is a regularization parameter.

Classification with MI-SVM proceeds in two steps. In the first step, $\mathbf{x}_i$ is initialized as the centroid for every positive bag B_i as follows

$$\overline{\mathbf{x}}_i = \sum_{j \in I} \mathbf{x}_{ij} / l_i \quad (17)$$

The second step is an iterative procedure in order to optimize the parameters.

Firstly, $\mathbf{w}$ and b are computed for the data set with positive samples $\{x_I : Y_i = 1\}$.

Secondly, we compute

$$f_{ij} = \langle \mathbf{w}, \mathbf{x}_{ij} \rangle + b, \quad \mathbf{x}_{ij} \in \mathbf{B}_i$$

Thirdly, we change $\overline{\mathbf{x}}_i$ by

$$\overline{\mathbf{x}}_i = \mathbf{x}_j$$

$$j = \arg\max_{j \in I} f_{ij}, \forall I, Y_I = 1$$

The iteration in this step will stop when there is no change of $\overline{\mathbf{x}}_i$. The optimized parameters will be used for testing.

References

[1] C. Alberti *et al.*, “An audio indexing system for election video material,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2009, pp. 4873–4876.

[2] J. Foote, “Automatic audio segmentation using a measure of audio novelty,” in *Proc. IEEE Int. Conf. Multimedia Expo*, 2000, pp. 452–455.

[3] S. Allegro, M. Bchler, and S. Launer, “Automatic sound classification inspired by auditory scene analysis,” in *Proc. Workshop Consistent Reliable Acoust. Cues Sound Anal.*, 2001, pp. 45–48.

[4] B. Picart, S. Brognaux, and S. Dupont, “Analysis and automatic recognition of human beatbox sounds: A comparative study,” in *Proc. IEEE Int. Conf. Acoust. Speech, Signal Process.*, 2015, pp. 4225–4229.

[5] G. Chen and B. Han, “Improve K-means clustering for audio data by exploring a reasonable sampling rate,” in *Proc. 7th Int. Conf. Fuzzy Syst. Knowl. Discovery*, 2010, pp. 1639–1642.

[6] M. Riley, E. Heinen, and J. Ghosh, “A text retrieval approach to contnent-based audio retrieval,” in *Proc. Int. Soc. Music Inf. Retrieval*, 2008, pp. 1639–1642.

[7] X. Shao, C. Xu, and M. Kankanhalli, “Unsupervised classification of music genre using hidden markov model,” in *Proc. IEEE Int. Conf. Multimedia Expo*, 2004, pp. 2023–2026.

[8] R. Cai, L. Lu, and A. Hanjalic, “Unsupervised content discovery in composite audio,” in *Proc. Int. Conf. Multimeida*, 2005, pp. 628–637.

[9] T. Sainath, D. Kanevsky, and G. Ivengar, “Unsupervised audio segmentation using extended baum-welch transformations,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2007, pp. I-209–I-212.

[10] G. Xia, D. Liang, R. Dannemberg, and M. Harvilla, “Segmentation, clustering, and displaying in a personal audio database for musicians,” in *Proc. Int. Soc. Music Inf. Retrieval*, 2011, pp. 139–144.

[11] [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2016/task-audio-tagging

[12] A. Kumar and B. Raj, “Audio event detection using weakly labeled data,” in *Proc. 2016 ACM Multimedia Conf.*, 2016, pp. 1038–1047.

[13] Y. Petetin, C. Laroche, and A. Mayoue, “Deep neural networks for audio scene recognition,” in *Proc. 23rd Eur. Signal Process. Conf.*, pp. 125–129.

[14] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, “Polyphonic sound event detection using multi label deep neural networks,” in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–7.

[15] P. Hamel, S. Wood, and D. Eck, “Automatic identification of instrument classes in polyphonic and poly instrument audio,” in *Proc. Int. Soc. Music Inf. Retrieval*, 2009, pp. 399–404.

[16] S. Dieleman and B. Schrauwen, “End-to-end learning for music audio,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 6964–6968.

[17] K. Choi, G. Fazekas, and M. Sandler, “Automatic tagging using deep convolutional neural networks,” in *Proc. Int. Soc. Music Inf. Retrieval*, 2016, pp. 805–811.

[18] T. Lidy and A. Schindler, “CQT-based convolutional neural networks for audio scene classification and domestic audio tagging,” *IEEE AASP Challenge Detect. Classif. Acoust. Scenes Events*, Budapest, Hungary, Tech. Rep 2016.

[19] E. Cakır, T. Heittola, and T. Virtanen, “Domestic audio tagging with convolutional neural networks,” *IEEE AASP Challenge Detect. Classif. Acoust. Scenes Events*, 2016. [Online]. Available: https://www.cs.tut.fi/sgn/arg/dcase2016/documents/challengetechnicalreports/Task4/Cakir

[20] P. Foster, S. Sigtia, S. Krstulovic, J. Barker, and M. Plumbley, “CHiME-home: A dataset for sound source recognition in a domestic environment,” in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2015, pp. 1–5.

[21] A. Coates, H. Lee, and A. Y. Ng, “An analysis of single-layer networks in unsupervised feature learning,” *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.

[22] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[23] S. Zhang, Y. Bao, P. Zhou, H. Jiang, and L. Dai, “Improving deep neural networks for LVCSR using dropout and shrinking structure,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 6849–6853.

[24] G. E. Dahl, T. N. Sainath, and G. E. Hinton, “Improving deep neural networks for LVCSR using rectified linear units and dropout,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 8609–8613.

[25] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, “Dynamic noise aware training for speech enhancement based on deep neural networks,” in *Proc. INTERSPEECH*, 2014, pp. 2670–2674.

[26] G. Hinton *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[27] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, “A regression approach to speech enhancement based on deep neural networks,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 1, pp. 7–19, Jan. 2015.

[28] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[29] P. Zhou and J. Austin, “Learning criteria for training neural network classifiers,” *Neural Comput. Appl.*, vol. 7, no. 4, pp. 334–342, 1998.

[30] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” 2012, arXiv preprint arXiv:1207.0580.

[31] R. Radhakrishnan, A. Divakaran, and A. Smaragdis, “Audio analysis for surveillance applications,” in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2005, pp. 158–161.

[32] L.-H. Cai, L. Lu, A. Hanjalic, and L.-H. Zhang, “A flexible framework for key audio effects detection and auditory context inference,” *IEEE Trans. Audio Speech, Signal Process.*, vol. 14, no. 3, pp. 1026–1039, May 2006.

[33] P. Hamel and D. Eck, “Learning features from music audio with deep belief networks,” in *Proc. 11th Int. Soc. Music Inf. Retrieval Conf.*, 2010, pp. 339–344.

[34] C. V. Cotton and D. P. W. Ellis, “Spectral vs. spectro-temporal features for acoustic event detection,” in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2011, pp. 69–72.

[35] M. L. Seltzer, D. Yu, and Y. Wang, “An investigation of deep neural networks for noise robust speech recognition,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 7398–7402.

[36] D. Stowell and M. D. Plumbley, “Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning,” *Peer J*, vol. 2, 2014, Art. no. e488.

[37] Y. Vaizman, B. Mcfee, and G. Lanckriet, “Codebook-based audio feature representation for music information retrieval,” *IEEE Trans. Acoust., Speech, Lang. Process.*, vol. 22, no. 10, pp. 1483–1493, Oct. 2014.

[38] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.

[39] S. Molau, M. Pitz, R. Schluter, and H. Ney, “Computing mel-frequency cepstral coefficients on the power spectrum,” in *Proc. Int. Conf. Acoust. Speech, Signal, Process.*, 2001, vol. 1, pp. 73–76.

[40] F. Grézl, M. Karafiát, S. Kontár, and J. Cernocky, “Probabilistic and bottleneck features for LVCSR of meetings,” in *Proc. IEEE Int. Conf. Acoust. Speech, Signal, Process.*, 2007, vol. 4, pp. 757–760.

[41] H. Christensen, J. Barker, N. Ma, and P. Green, “The CHiME corpus: A resource and a challenge for computational hearing in multisource environments,” in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1918–1921.

[42] A.-R. Mohamed, G. E. Dahl, and G. Hinton, “Acoustic modeling using deep belief networks,” *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 14–22, Jan. 2012.

[43] K. P. Murohy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[44] M. I. Mandel and D. Ellis, “Song-level features and support vector machines for music classification,” in *Proc. 6th Int. Conf. Music Inf. Retrieval*, 2005, pp. 594–599.

[45] S. Yun, S. Kim, S. Moon, J. Cho, and T. Kim, “Discriminative training of GMM parameters for audio scene classification and audio tagging,” *IEEE AASP Challenge Detect. Classification Acoust. Scenes Events*, 2016. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2016/documents/challengetechnicalreports/Task4/Yun

[46] M. E. Schuckers, “Receiver operating characteristic curve and equal error rate,” in *Computational Methods in Biometric Authentication*. New York, NY, USA: Springer, 2010, pp. 155–204.

[47] B. Thomee *et al.*, “YFCC100M: The new data in multimedia research,” *Commun. ACM*, vol. 59, no. 2, pp. 64–73, 2016.

[48] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, “YouTube-8M: A large-scale video classification benchmark,” 2016, arXiv:1609.08675.

[49] S. Andrew, I. Tsochantaridis, and T. Hofmann, “Support vector machines for multiple-instance learning,” in *Proc. 15th Int. Conf. Neural Inf. Process. Syst.*, 2003, pp. 557–584.

Yong Xu (M'17) was born in 1988. He received the Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, China, in 2015, on the topic of DNN-based speech enhancement and recognition. He currently works at the University of Surrey, Guildford, U.K. as a Research Fellow. He once visited Prof. Chin-Hui Lee's lab in Georgia Institute of Technology, USA from September 2014 to May 2015. Prior to his current work, he once also worked in IFLYTEK company from April 2015 to April 2016 to develop far-field ASR technologies. His research interests include deep learning, speech enhancement and recognition, audio and scene classification, etc.

Qiang Huang (M'05) received the Ph.D. degree in computer science from the University of East Anglia (UEA), Norwich, U.K., in 2005. His early research was on the development wireless local area network, and then on speech and natural language processing. From 2006 to 2008, he was with the Knowledge Media Institute, Milton Keynes, U.K., where his research was on information retrieval. In 2009, he returned to UEA and worked as a Senior Research Associate on adaptive cognition by audio and visual analysis. Before moving to the University of Surrey, Guildford, U.K., in 2016, he worked in the School of Informatics, University of Edinburgh, and was in charge of a multimodal dialogue system and joined the project of Natural Speech Technology. His current research interests include multimodal signal processing, speech and natural language processing, and information retrieval. He is the author and co-author of about 40 publications in these fields.

Wenwu Wang (M'02–SM'11) received the B.Sc. degree in automatic control, the M.E. degree in control science and control engineering, and the Ph.D. degree in navigation guidance and control, all from Harbin Engineering University, Harbin, China, in 1997, 2000, and 2002, respectively. He then joined Kings College, London, U.K., in May 2002, as a Postdoctoral Research Associate and transferred to Cardiff University, U.K., in January 2004, where he worked in the area of blind signal processing. In May 2005, he joined the Tao Group Ltd. (now Antix Labs Ltd.), Reading, U.K., as a DSP Engineer. In September 2006, he joined Creative Labs, Ltd., Egham, U.K., as an R&D Engineer. Since May 2007, he has been with the Centre for Vision Speech and Signal Processing, University of Surrey, Guildford, U.K., where he is currently a Reader in Signal Processing, and a Co-Director of the Machine Audition Lab. During spring 2008, he was a Visiting Scholar at the Perception and Neurodynamics Lab and the Center for Cognitive Science, The Ohio State University. His current research interests include blind signal processing, sparse signal processing, audio–visual signal processing, machine learning and perception, machine audition (listening), and statistical anomaly detection. He has (co)-authored over 150 publications in these areas, including two books *Machine Audition: Principles, Algorithms and Systems* (IGI Global, 2010) and *Blind Source Separation: Advances in Theory, Algorithms and Applications* (Springer, 2014). He is currently an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING. He is also Publication Co-Chair of ICASSP 2019 (to be held in Brighton, U.K.).

Peter Foster received the M.Sc. and B.Sc. degrees in computer science from the University of Edinburgh, Edinburgh, U.K., and from the University of East Anglia, Norwich, U.K., respectively, and the Ph.D. degree from Queen Mary University of London, London, U.K., undertaken at the Centre for Digital Music. He is currently pursuing a career in industry with a focus on time-series analysis. He was subsequently employed as a Postdoctoral Research Assistant at the Centre for Digital Music, Queen Mary University of London. His interests include time-series similarity and classification.

Siddharth Sigtia received the B.E. degree in electrical and electronics engineering, the M.Sc. degree in physics from the Birla Institute of Technology ad Science, Pilani, India, and the Ph.D. degree in electronics engineering from the Centre for Digital Music, Queen Mary University of London, London, U.K. He is currently a Researcher at the Siri Speech team at Apple. His research interests include machine learning and neural networks for audio analysis, specifically deep neural networks for acoustic modeling for music, speech, and environmental audio and recurrent neural networks music language modeling.

Philip J. B. Jackson received the M.A. degree in engineering from Cambridge University, Cambridge, U.K., and the Ph.D. degree in electronic engineering from the University of Southampton, Southampton, U.K. He is a Senior Lecturer in machine audition at the Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, U.K.. His broad interests in acoustical signal processing provide research contributions in speech production, auditory processing and recognition, audio–visual machine learning, blind source separation, articulatory modeling, visual speech synthesis, spatial audio recording, reproduction and quality evaluation, and sound field control. He also leads research on object-based production in the S3A project on future spatial audio, and enjoys listening to sound.

Mark D. Plumbley (S'88–M'90–SM'12–F'15) received the B.A.(Hons.) degree in electrical sciences and the Ph.D. degree in neural networks from the University of Cambridge, Cambridge, U.K., in 1984 and 1991, respectively. From 1991 to 2001, he was a Lecturer with Kings College London, London, U.K., before moving to Queen Mary University of London, London, in 2002, later becoming the Director of the Centre for Digital Music. In 2015, he joined the Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, U.K., as a Professor of Signal Processing. His research interests include automatic analysis of music and other sounds, including automatic music transcription, beat tracking and acoustic scene analysis, using methods such as source separation, sparse representations and deep learning. He is a Member of the IEEE Signal Processing Society Technical Committee on Signal Processing Theory and Methods.