Environmental Audio Tagging: Trends and Techniques

Dr. Jayasudha.J.S [1]
Mrs. Sangeetha.M.S [2]
Department of Computer Science and Engineering, Sree Chitra Thirunal College of Engineering, Trivandrum, India

***Abstract*—Real life environment consists of various kinds of sounds other than speech and music. All these sounds carry information about our everyday environment and have its own features. In order to categorize different kinds of sounds and to study them separately, tagging is introduced into the area of sound analysis. Environmental audio tagging predicts the presence or absence of certain acoustic events in the interested acoustic scene. Audio tagging forms the backbone of sound recognition and classification work. This work on audio tagging consists of extracting relevant features from input audio and of using those features to identify a set of classes into which the sound is most likely to fit. Existing works for this task largely uses conventional classifiers which do not have the feature abstraction found in deeper models. A deep learning framework is used here for unsupervised feature learning and classification.**

Keywords*—deep learning, environmental audio tagging, unsupervised feature learning, multilabel classification.

I. Introduction

The main challenge limiting widespread acceptance of large, freely available internet audio archives is the difficulty in organizing and accessing them. Traditionally, text-based retrieval approaches are often employed. But the disadvantage is that it is impossible to search for unlabeled sound files. To overcome this, there has been much recent interest in retrieving unlabelled audio from text queries and the related problem of auto-tagging, i.e., the ability to automatically describe and label a sound clip based on its audio content.

There has been a considerable amount of attention paid to the automatic prediction of tags for music and audio. Social tags are user-generated keywords associated with some resource on the Web. There have been many attempts at automatically applying tags to audio for different purposes: database management, music recommendation, improved human computer interfaces, estimating similarity among songs, and so on.

Audio tagging aims at putting one or several tags on a sound clip. The tags are the sound events that occur in the audio clip, for example, speech, television, percussion, bird singing, and so on. Audio tagging has many applications in areas such as information retrieval, sound classification and recommendation system. For environmental audio tagging, there is a large amount of audio data on-line. How to utilize them, predict them and further add some new tags on the related audio is a challenge. The audio recordings from surrounding environment 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.

II. Literature Review

The sound events in real life have no fixed pattern. Different contexts, such as forest, city and home contain different kinds of sounds. They can be of different sparsity based on the context. They can occur in isolation or be completely overlapped with other sound events. Identifying isolated sounds have been done with considerable accuracy. But identifying a mixture of labels in an overlapped sound environment is a challenging task, where still considerable amount of improvements can be made.

Recently, the deep learning based methods have also been widely used for environmental audio tagging, a newly proposed task in DCASE 2016 challenge based on the CHiME-home dataset [18]. 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 occur with TV sound for several seconds. Hence, a more robust deep learning method is needed to improve the audio tagging performance.

A. Mood Detection System

Garima Vyas et al. used K-means clustering algorithm and decision rule to identify the mood underlying a piece of music by extracting suitable and robust features from music clip [1]. The objective here is to accurately attach mood tag to a musical audio. Mel frequency cepstral coefficients, frame energy and peak difference are the three features that were extracted to decide the mood tag of the musical piece. The Audio Preprocessor performs the task of pre-processing the audio clips provided by the user to the system. The pre-processing job involves the following steps:

1) Audio File Splitting: An audio with duration of 45 sec is clipped. Such a clip has proved to be acceptable from experimentation point of view as it is not very short to lose any important information and not very long to increase the processing time.
2) Audio Format Conversion: Each music clip of 45 second is transformed to a standard format namely WAV (type as stereo and 16 bits PCM) with a rate of sampling 44.1 kHz. Thus, this component makes sure that the input music clips given by the user are transformed so that they can be developed for processing and analyzing.

The Audio Feature Extractor revolves around the features of audio signal associated with the music clips obtained as a result from the Audio Pre-processor. The audio clip of 45sec is segmented into frames of 55 60ms with an overlap of 35 40ms. From each frame the MFCCs (Mel frequency cepstral coefficients) and energy is computed. Then the maximum and minimum peak is calculated which in turns gives the peak difference.

The Mood Identification System is the major processing unit of the whole system and is accountable for mining the mood from the music dataset acquired as input from the audio feature extractor module. The module has two important functions to perform as mentioned below:-

1) Mood Learner: The input is received in the form of a training dataset of music features with the manually updated mood attribute by the domain experts, from the training point of view. In this case clustering is used to train the system.
2) Mood Detector: The Mood detector is used to evaluate the dataset under consideration against the mood classifier model that is finalized. The evaluation uses the threshold value to classify the mood of an audio. In case a full song is fed instead by the user, the system reciprocates the maximum voted mood from the moods predicted for the clips derived from that song. The final output of this module is generally utilized by the end-user application like any Music information retrieval application or a mood-annotator. Decision Rule is used for tagging a mood to the musical song on the basis of peak difference and frame energy. According to a comprehensive study, it is observed that sad mood music clips have peak difference less than 0.6000 and frame energy less than 57.0000. On the other side, the happy mood songs have peak difference greater than 0.8000 and energy greater than 57.0000.
The architecture of mood detection system is shown in figure 1. The audio pre-processor, mood identification system and audio feature extractor are the various modules in the mood detection system.

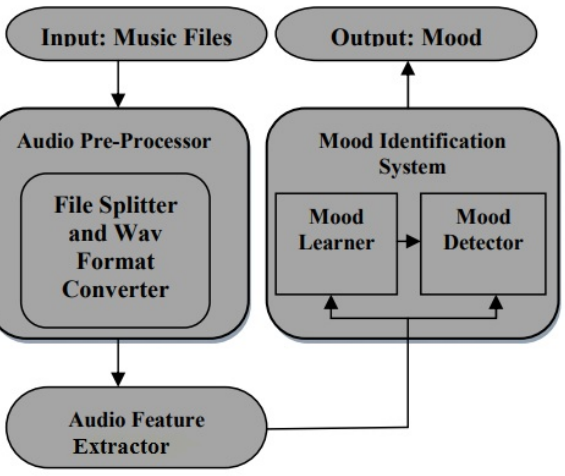


Fig. 1. Mood Detection System

B. Support Vector Machine

Jrgen T. Geigeret al. use SVM for acoustic scene classification [2]. The system classifies 30 second long recordings of 10 different acoustic scenes. From the extremely uneven recordings, a large number of spectral, cepstral, energy and voicing-related audio features are extracted. Using a sliding window approach, classification is performed on short windows. SVM are used to classify these short segments, and a majority voting scheme is employed to get a decision for longer recordings. To better capture the non-stationary nature of the scenes, classification is performed on smaller windows. Each recording is split into (overlapping) windows with a length of several seconds, and the statistical functionals are computed for all low level descriptors in those segments. Longer windows lead to better results. This windowing is done on the training data and on the test data. Thus, models are trained with a larger number of training instances per class. Classification is performed on the windowed test data. Each of the windows is separately fed to the classifier to recognize one part of the scene. In order to get one decision for the whole instance, a majority voting scheme is employed.

For evaluation of the system, the official dataset of the IEEE AASP Challenge on Detection and Classication of Acoustic Scenes and Events were employed. This dataset contains 30 seconds recordings of various acoustic scenes, categorised into ten dierent classes. Using large-scale audio feature extraction and SVM, an accuracy of 73% is obtained on the development set of the D-CASE challenge.

C. The Random Forests

Li Yang and Feng Su, employed a random forest based ensemble learning and classication model for auditory contexts, in which individual segments of audio stream are classied and aggregated by Hough voting or bagging to form the nal context category [3]. Auditory contexts refer to the acoustic modeling of a specic location or site such as restaurant or bus station, while audio events are short audio segments with distinct acoustic patterns corresponding to specic object or event in an auditory context, such as laughing or gunre.

The block diagram of the auditory context classication is shown in figure 2. A composite audio feature representation that characterizes dierent aspects of the signal stream at audio event scale is employed. Three elementary features: the local discriminant bases (LDB) feature, the pseudo-semantic (PSEM) feature, and the bag-of-audio-words (BOAW) feature, are concatenated to form the feature vector for an audio signal segment. Random forest based method for auditory context classication is applied. Correspondingly, an ensemble bagging/voting scheme for auditory context classication, which aggregates the class votes casted by individual segments of the input audio stream for the potential context category is employed.

To evaluate the auditory context classification method, the test data is collected for 10 auditory contexts from Internet and some movie/TV clips, including 6 outdoor and 4 indoor contexts: auditorium, war field, forest, beach, train station, street, inside vehicle, playground, restaurant and raining. Total 21 audio event categories are considered, including engine, car-braking, siren, horn, music, gun-shot, explosion, running water, bird twittering, thundering, talking, applause, laughter, cheer, etc. The training and testing set contain around 100 and 150 mono channel audio samples, respectively, for each audio event and context category. The typical sample length is 1-3 seconds for audio events and 15 seconds 2 minutes for contexts with 44.1kHz sampling rate. The experiment results show the effectiveness of the random forest based framework, and combination of several heterogeneous features incrementally enhances the average performance. The flexibility in tuning feature compositions according to the complexity and efficiency requirements makes the framework potentially applicable to a wide range of circumstances.

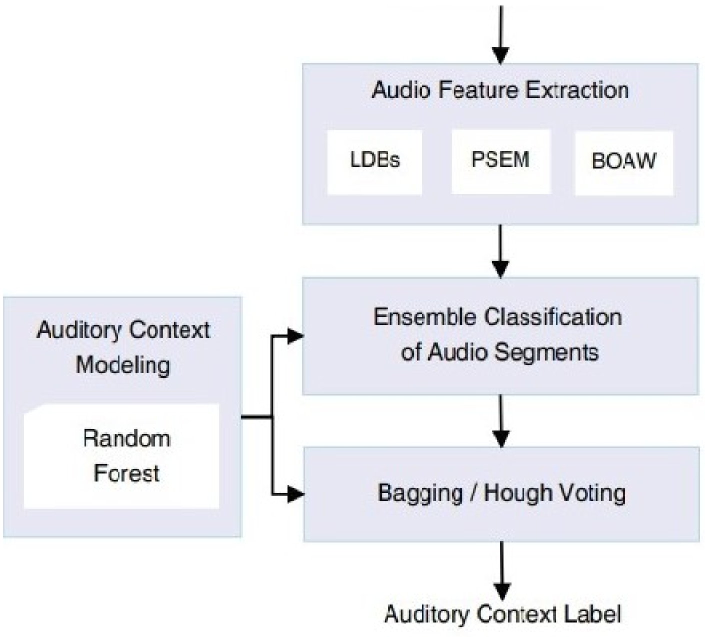


Fig. 2. Block diagram of the auditory context classification algorithm

D. Know Thy Neighbor

The Know-Thy-Neighbor(KTN) algorithm is used by Alexandros Nanopoulos et al. to measure the similarity of 2 musical pieces [4]. The k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input includes k nearest training examples in the feature space. The output relies on whether k-NN is used for classification or regression.

The Know-Thy-Neighbor (KTN) method, commences with a training phase in which the tracks of collection, D act as training data for which weights are learned. The weight of each such track is defined as its number of occurrences, i.e., the number of times it appears among the n nearest neighbors of the rest tracks in D. The n-occurrences are initialized to zero and computed as follows. For each track in D, find a list: L1 of its n nearest neighbors based on the audio feature space and L2 of its n nearest neighbors based on the tag feature space. Then, increase the number of n-occurrences for each track in L1 L2, treat the two feature spaces uniformly.

The intuition behind KTN is as follows: since adequate tags does not exist for the query track, KTN identifies a representative neighbor for the query track in the audio feature space. The representatives of tracks in the audio feature space is determined by their learned weights, i.e., their occurrences, which promotes tracks that have the property of being popular nearest neighbors. This property is known as hubness. Since the selected neighboring track is considered as representative, its k nearest neighbors based on the tag feature space are returned as an accurate estimation of the actual nearest neighbors of the query song. Therefore, KTN combines effectively both feature spaces to address the cold start problem.

Audio data were harvested using the iTunes API. Track selection was based on the cumulative highest popularity tags offered for the track in Last.fm by selecting the 50 top rank tracks for each top rank tag. The data gathered contain 5,459 discrete tracks and each track is a 30 second clip of the original audio.

The KTN method is suitable in the case of the cold-start problem that results from the lack of adequate tags. This method avoids the forced use of solely audio-based similarity measures when measuring music similarity and utilizes available contextual knowledge in the form of social tags. The KTN is also shown to be effective in comparison to the audio-based similarity computation with respect to precision of the resulting similarity measures. This is verified through experimental results with real data, which illustrate the suitability of this method.

E. Hidden Markov Model

Xi Shao et al. suggested Hidden Markov Model as an unsupervised approach for automatic music genre classification. The observed low-level audio features are classified into different music categories. A hidden Markov model is a doubly embedded stochastic process, where the actual states producing the output are hidden. In simpler Markov models (like a Markov chain), the state is directly observable, and therefore the state transition probabilities are the sole parameters, while in the hidden Markov model, the state is not directly visible, but the output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the series of tokens created by an HMM provides some information about the series of states.

This approach contains two steps:

- Every individual music piece is segmented into clips, and each clip is further segmented according to its intrinsic rhythmic structure. Features are extracted based on these segments. Then train a Hidden Markov Model (HMM) for this music piece based on these features.
- Embed the distance between every pair of music pieces into a distance matrix and perform clustering to generate desired clusters.

Clustering by Hidden Markov Models: Initially, a HMM model is build for each music piece. Every music piece is split into clips. These clips belonging to one music piece are used to train a HMM model as shown in figure 3. Assume there are N pieces of music in the database, then the distance between two music pieces will be calculated and the distance matrix D is N x N dimension. Given a distance matrix D, many clustering methods can be used. Here, Agglomerative Hierarchical Clustering is used to generate clusters. This method does bottom-up clustering. It starts with N singleton clusters and forms a sequence of clusters by successive merging. For the scenario where the number of desired clusters is known (denoted as c), the merging process will come to an end when the C clusters are produced. While for the number of desired clusters is unknown, the merging process will come to an end when the distance between two nearest clusters is above a threshold.

The music dataset for each genre contains 50 music pieces. The genres are Pop, Country, Jazz and Classic. They are obtained from music CDs and internet. All data are 44.1 kHz sample rate, mono channels and 16 bits per sample. The average classification accuracy is 89% using this dataset and HMM topology.

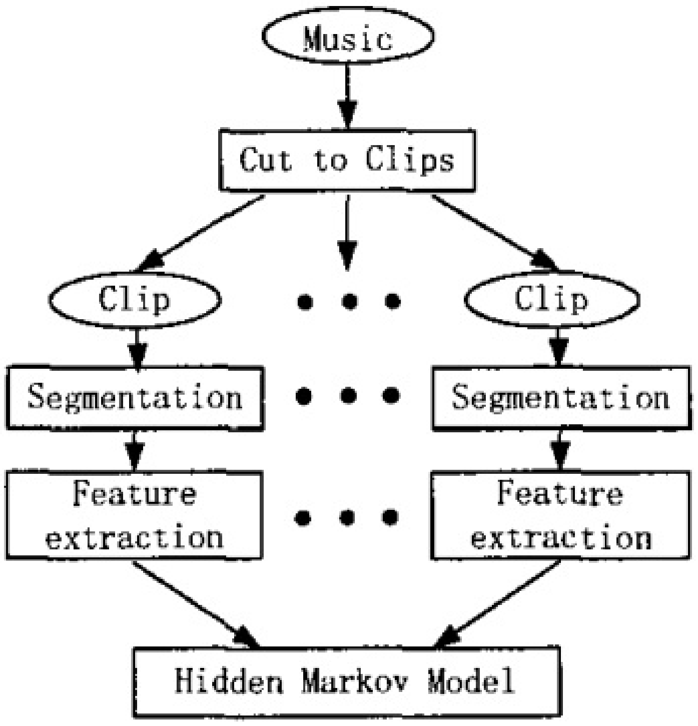


Fig. 3. HMM training for individual piece music

F. Soft Annotator Fusion

Remi Foucard *et al.* described a method of fusing annotations that preserves information about the uncertainty of the tag/song association [6]. This fusion provides with continuous scores, that are used for training a regressive boosting algorithm. Boosting is a learning technique, training iteratively several complementary versions of a weak (performing badly) classifier.

The soft scores are constructed based on the annotators individual responses. Firstly, every possible response is converted to a value v ∈ [0, 1]. Consecutive values are equally spaced, moreover v = 0 and v = 1 must always be possible answers. For instance, there are four possible responses for the tag Instrument-Trumpet: None, Uncertain, Present and Prominent. They are respectively mapped to: 0, 0.33, 0.67 and 1. Because majority voting and threshold mean calculation do not reflect uncertainty, the individual scores are averaged as given in equation 1.

$$V_s = \frac{1}{K}\sum_{k=1}^{K} v_k \tag{1}$$

where v_k is the value corresponding to the choice made by annotator k. Alternatively, for the negative tags (e.g. Emotion NOT happy), the value is simply $V = 1 - P$, where P is the value associated with the corresponding positive tag.

Adaboost: The soft scores obtained by annotator fusion will be used to train a regressive boosting system. AdaBoost, which is an abridged form of "Adaptive Boosting", is a machine learning meta-algorithm. It can be used along with many other types of learning algorithms to improve their performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that depicts the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are adjusted in favor of those instances misclassified by preceeding classifiers.

The experiment is done on the CAL500 database. This database contains 500 pop songs, with tags describing mood, instrumentation, genre, etc. The tests are conducted with 10-fold cross-validation, keeping 450 songs for training, and 50 for testing. For complexity reduction, Remi Foucard *et al.* only use 30 seconds of each song: between instants 30 seconds and 60 seconds.

Two different ranking metrics are used to evaluate this output. Ranking metrics evaluate the list of examples ranked by predicted score S_n. This list is compared against the binary ground truth. A perfect ranking would put all positive songs at the top.

- The first measure is the Mean Average Precision (MAP). It can be obtained by moving down the ranked list, and averaging the precision obtained at every truly positive example.
- The second is the Receiver Operating Characteristic (ROC) curve. This curve represents the correct detection rate with respect to the false alarm rate, computed at each element in the ranking.

Remi Foucard *et al.* also employed regressive boosting for learning the scores obtained by this fusion.The results show that the soft scores, combined with regressive learning, lead to a better learning of the tags.

G. Convolutive Non-negative Matrix Factorization

Courtenay V. Cotton and Daniel P. W. Ellis employs convolutive non-negative matrix factorization (NMF) as an approach for detecting and modeling acoustic events that directly describes temporal context [7]. NMF is useful for finding parts-based decomposition of data; It is used to discover a set of spectro-temporal patch bases which aptly describe the data, with the patches analogous to event-like structures. Then features are derived from the activations of these patch bases, and perform event detection on a database made up of 16 classes of meeting-room acoustic events. This NMF algorithm allows both to locate transients in time and to build a dictionary of event-patch codewords, within a single optimization framework, avoiding the separate transient detection and patch clustering. The metric used for evaluation is the acoustic event error rate (AEER) that was used in CHIL evaluations for the event detection task and is defined in equation 2.

$$AEER = 100(D + I + S)/N \qquad (2)$$

where D is the number of deletions, I is the number of insertions, S is the number of substitutions, and N the total number of events that occur in the ground truth labels.

H. Joint Detection - Classification Model

Qiuqiang Kong*et al.* used a joint detection-classification (JDC) model to detect and classify the audio clip simultaneously [8]. The JDC model has the capability to handle informative and disregard uninformative sounds. Then only informative regions are used for classification. The joint detection classification (JDC) model is inspired by humans perception. Humans use two steps to label an audio, a detection step and a classification step. In the detection step, humans listen when to attend to sounds and when to ignore sounds. They attend to informative audio events and ignore uninformative sounds or silences. In a classification task, humans only tag informative events. The JDC model is trained on weakly labelled data, without requiring event level labelled data. The JDC model has the following features:

- Weakly labelled data can be fed into the model directly without the event level label.
- The detectors attend and ignore mechanism simulates the humans perception.
- The detectors attend and ignore mechanism facilitates the recognition of short audio events.
- Audio event detector is trained without the event level label.

For the audio tagging task, assume there are K different audio tags. The labelled target of an audio clip can be denoted as $t \in (0, 1)^k$ where $t_k \in 0, 1$ represents the existence of the k^{th} tag. The JDC model consists of a detector and a classifier acting on each tag and each block with output value between 0 and 1. The output of the classifier on the k^{th} tag and the m^{th} block indicates how possible the m^{th} block has tag k. The output of the detector w^{km} on the k^{th} tag and the m^{th} block indicates how informative the m^{th} block is when classifying the k^{th} tag. If $w^{km}$is close to 1 it means the m^{th} block is informative and should be attended when classifying the k^{th} tag. If w^{km} is close to 0 it means the m^{th} block is uninformative and should be ignored when classifying the k^{th} tag. The loss function of the JDC model is defined in equation 3:

$$loss = \sum_{k=1}^{K} d(p_k, t_k) \qquad (3)$$

where $d(p_k, t_k)$ is the binary cross-entropy error.

The experimental results on the CHiME Home dataset show that the JDC model reduces the equal error rate (EER) from 19.0 % to16.9%. The audio event detector is trained successfully without requiring the event level label.

I. Automatic Music Annotation with Acoustically-Objective Tags

Derek Tingle et al. developed an autotagging system to create a large-scale semantic music discovery engine[9]. The acoustic tags are considered acoustically objective because they can be frequently implemented to songs by expert musicologists. The audio features used for comparison are Echo Nest Timbre (ENT), Echo Nest Song (ENS) and Mel frequency cepstral coecients (MFCC).

Autotagging can be considered a multiclass, multilabel classification problem in which each song can be labeled with multiple tags. For each tag in the vocabulary, Derek Tingleet al. train a classifier using the audio features that are extracted from annotated training songs. To annotate a new song, each tag-level classifier is used to predict a relevance score (e.g., probability) that is proportional to the strength of association between the song and the tag. Then during evaluation, songs are rank ordered based on their predicted relevance to a given tag.

J. Bag Of Features Approach

Axel Plinge et al.,proposed a Bag-of-Features approach for classifying acoustic events [10].Mel and gammatone frequency cepstral coefficients that evolve from psychoacoustic models are employed as input features for the Bag-of representation. Rather than using a prior classification or segmentation step to remove silence and background noise, Bag of Features representations are learned for a background class. Supervised learning of codebooks and temporal coding are used to improve the recognition rates.

A Bag-of-Features approach is used for building a codebook of acoustic words from the training set. Most Bag of Features approaches use clustering algorithms, e.g. the Lloyd algorithm, on the complete training set to derive a codebook and later assign each feature to a centroid by hard quantization. The expectation-maximization (EM) algorithm is applied to all feature vectors y_k for each class Ω_c in order to estimate means and deviations μ_{ic}, σ_{ic} for all C classes. A soft quantization of a feature vector y_k can be computed as given in equation 4:

$$q_{(k,l)}(y_k, v_l) = N(y_k|(\mu_l, \sigma_l)) \quad (4)$$

Then, a histogram b can be computed over all K frames of the input window by using equation 5:

$$b_{(l)}(y_n, v_l) = \frac{1}{K} \sum K q_{(k,l)}(y_k, v_l) \quad (5)$$

This method of Bag-of-Super-Features is analogy to the super-vector construct used in speaker identification.

The Bag of Features method was evaluated on the recent IEEE AASP Challenge Detection and Classification of Acoustic Scenes and Events Event Detection development set. Since the training data consists of event recordings only, non labeled portions of the scripts not used in the test were used for training in order to have training data. For the non-event class 88% precision and 90% recall were obtained.

K. Deep Neural Networks

Robust sound event classification, the ability to recognize sounds under real-world noisy conditions, is an especially challenging task. A sound event classification framework compares auditory image front end features with spectrogram image-based front end features using deep neural network classifiers [11]. The requirement is that a trained system, when presented with an unknown sound, is capable of correctly identifying the class of that sound. Furthermore, these techniques should be robust to interfering acoustic noise.

Both Stabilised Auditory Image (SAI) and time-frequency domain Spectrogram Image Features (SIF), will be evaluated for standard robust sound event classification tasks [11]. The sound event detectors or classifiers have been used here. They have been evaluated under real world conditions including severe levels of degrading acoustic background noise. In each case, the front end analysis and feature extraction operations are followed by back-end machine learning methods.

Ian McLoughlin *et al.*, first investigated the use of Googlestyle SAI features with a back end SVM classifier and used this baseline to evaluate the effect of several modifications to the feature extraction and representation process [11]. Next, the SVM classifier in the best performing system is replaced with a DNN back-end. Finally, the DNN classification performance is evaluated with a number of different feature representations that are derived from an SIF. A block diagram of the SAI extraction and feature vector formation can be seen in Fig. 4. Subsequently, a novel low-resolution overlapped spectrogram image feature was developed and evaluated with the DNN classifier. Several variants of the system were then proposed and evaluated, including a simple de-noising method as well as post-processing of context-by-context classification outputs across a single sound.

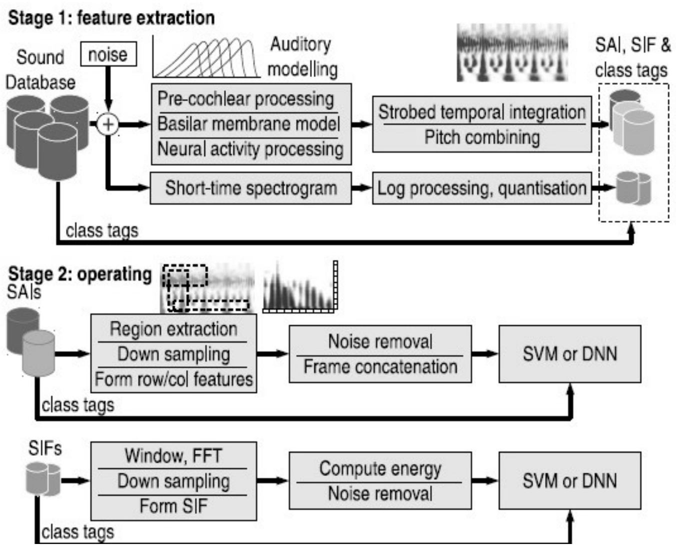


Fig. 4. Block diagram of audio event classification system using front-end SAI and SIF analysis

A total of 50 sound classes are chosen from the Real World Computing Partnership (RWCP) Sound Scene Database in Real Acoustic Environments as the dataset. In the RWCP database, every class contains 80 recordings and contains a single example sound per recording. The sounds were captured with high SNR and have both lead-in and leadout silence sections. The training data set comprises 50 randomly-selected files from each class. The remaining 30 files from each class are set aside for evaluation. Therefore, a total of 2500 files are available for training and 1500 per testing run. All evaluations apart from the multi-condition tests use classifiers that are trained with exclusively clean sounds, with no pre-processing or noise removal applied. In all cases, evaluation is performed separately for both clean sounds, as well as sounds corrupted by additive noise. The noise-corrupted tests use four background noise environments selected from the NOISEX-92 database.

L. Convolutional Gated Recurrent Neural Network

Yong Xu *et al.* proposed a convolutional neural network (CNN) to extract robust features from mel-filter banks (MFBs), spectrograms or even raw waveforms for audio tagging [12]. Gated recurrent unit (GRU) based recurrent neural networks (RNNs) are then grouped to model the long-term temporal structure of the audio signal. To accompany the input information, an auxiliary CNN is proposed to learn on the spatial features of stereo recordings.

Fig. 5 shows the framework of a convolutional gated recurrent neural network for audio tagging. The CNN is regarded as the feature extractor along the short window (e.g., 32ms) from the basic features, e.g., MFBs, spectrograms or raw waveforms. Then the robust features extracted are fed into the GRU-RNN to learn the long-term audio patterns. The CNN also assists to extract robust features opposed to the background noise by the max-pooling operation, especially for the raw waveforms. Since the label of the audio tagging task is at the chunk-level rather than the frame-level, a large number of frames of the context were fed into the whole framework. The GRU-RNN can select related information from the long term context for each audio event. A bi-directional GRU-RNN is designed to utilize the future information. Finally the output of GRU-RNN is mapped to the posterior of the target audio events through one feed-forward neural layer, with sigmoid output activation function. This framework is flexible enough to be applied to any kinds of features, especially for raw waveforms. Raw waveforms have lots of values, which leads to a high dimension problem. However the proposed CNN can learn on short windows like the short-time Fourier transform (STFT) process, and the FFTlike basis sets or even mel-like filters can be learned for raw waveforms. Finally, one-layer feed-forward DNN gets the final sequence of GRUs to predict the posterior of tags.

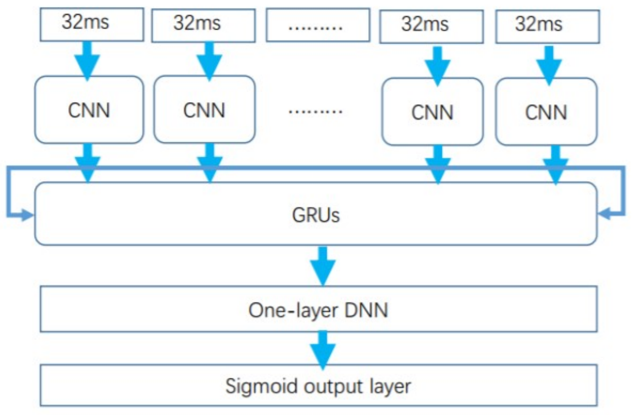


Fig. 5. The framework of convolutional gated recurrent neural network for audio tagging

M. Semi-Supervised Canonical Density Estimation

Jun Takagi *et al.* proposed a novel semi-supervised method for building a statistical model that represents the relationship between sounds and text labels (tags)[13]. This method, of semi-supervised canonical density estimation (SSCDE), makes use of unlabeled sound data in two ways:

- A low-dimensional latent space representing several topics of audio is extracted by a semi-supervised form of canonical correlation analysis.
- Topic models are learned by multi-class extension of semi-supervised kernel density estimation in the topic space.

SSCDE fully makes use of unannotated sounds for both feature extraction and model estimation. More specically, a low dimensional latent space representing topics of music is estimated by applying a semi-supervised variant of canonical correlation analysis called SemiCCA, which extends the ordinary CCA to be able to utilize both paired and unpaired samples. Then, in the estimated latent space, topic models are learned by multi-class extension of semi-supervised nonparametric density estimation called semi-supervised kernel density estimation (SSKDE).

N. Automatic Audio Tagging Using Covariate Shift Adaptation

Gordon Wichern *et al.* demonstrated a semi-supervised approach to automatic tagging of general audio files under a covariate shift assumption[14]. They assumed the audio files follow a covariate shift model in the acoustic feature space, i.e., the feature distributions are different in the training and test phases, but the conditional distribution of labels given features remains unchanged. This method uses a specially designed audio similarity measure as input to a set of weighted logistic regressors, which attempt to alleviate the influence of covariate shift. The trends in low-level audio feature trajectories (does each feature stay constant, go up, down, or vary in more complex ways) are approximated and use the distance between these low-level feature trends as a kernel function in a kernel logistic regression classification scheme. Importance weights [16] are then applied to overcome possible shifts in the audio feature space between the training and test sets, where weights are estimated using the Kullback-Leibler importance estimation procedure [17]. The performance is tested using data from the Freesound project, a database of freely available sound recordings uploaded by users of the site. The classifiers are trained using only uncompressed high bit rate audio files, while testing data contains only low bit-rate compressed files.

O. Attention and Localization based on a Deep Convolutional Recurrent Model for Weakly Supervised Audio Tagging

Audio tagging aims to perform multi-label classification on audio chunks. The problem with audio tagging is that it has only a chunk-level label lacking a frame-level label. Yong Xu *et al.* presents a weakly supervised method to not only predict the tags but also specify the temporal locations of the occurred acoustic events [15]. There are 2 modules in this deep convolutional recurrent model:

- Attention module
- Localization module

The term attention is used to focus on specific parts of the input. For the audio tagging task, the attention method can automatically select and attend on the important frames for the targets while ignoring the unrelated parts (e.g., the background noise segments). It can also be regarded as learning a weighting factor on each frame.The attention scheme is conducted based on the convolutional gated recurrent neural network. The localization module is defined to find the temporal locations when the specific event happens. Attention is used for global event-independent frame-level feature selection, while the event-dependent localization is used to find the locations of each event.

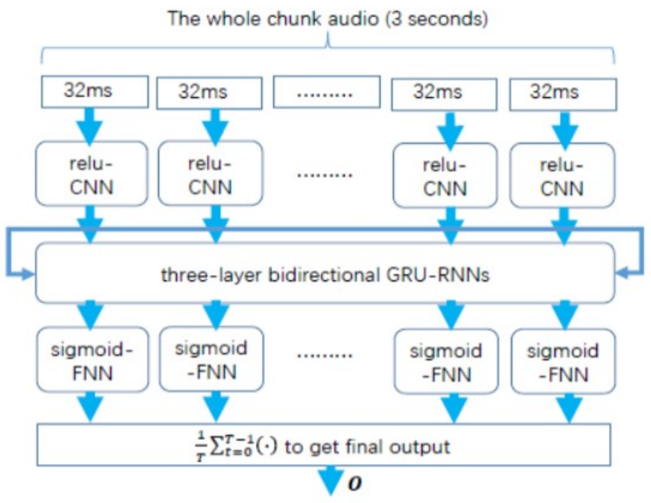

The whole chunk audio (3 seconds)
32ms
32ms
....
32ms
32ms
relu-CNN
relu-CNN
....
relu-CNN
relu-CNN
three-layer bidirectional GRU-RNNs
sigmoid-FNN
sigmoid-FNN
....
sigmoid-FNN
sigmoid-FNN
$\frac{1}{T}\sum_{t=0}^{T-1}(\cdot)$ to get final output
O

Fig. 6. The framework of the chunk level convolutional gated recurrent neural network for audio tagging

The framework of the chunk-level convolutional gated recurrent neural network for audio tagging is shown in Fig. 6. The whole audio chunk is chopped into frames with half overlap. Each frame is fed into a convolutional neural network (CNN) with a large receptive field considering that only one CNN layer is used. The CNN can help to extract more robust features through the max-pooling operations. Rectified Linear Unit (ReLU) is the activation function of CNNs. The output activations of each frame from the CNNs are fed into the following gated recurrent unit (GRU) based recurrent neural network (RNN). Then three-layer GRU-RNNs are followed by one-layer feed-forward neural network (FNN) and the activation function is Sigmoid. The audio tagging is a multi-label task which means several acoustic events could happen simultaneously. Hence the output activation function should be sigmoid. Ultimately, each frame can create one prediction for the audio tags. Their results should be averaged together to obtain the final predictions. The binary cross-entropy and tag vector are calculated using equations 6 and 7.

$$E = -\sum_{n=1}^{N}(P_n log O_n + (1-P_n) log(1-O_n)) \qquad (6)$$

$$O = \frac{1}{T}\sum_{t=0}^{T-1}(1+exp(-S_t))^{-1} \qquad (7)$$

where E is the binary cross-entropy, O_n and P_n denote the estimated and reference tag vector at sample index n, respectively. The bunch size is represented by N. The feed forward neural network (FNN) linear output is defined as S_t at t^{th} frame before the sigmoid activation function is applied. T denotes the total number of frames in the whole audio chunk.

III. Deep Neural Network and Auto Encoder for Audio Tagging

Deep neural networks (DNNs) are receiving attention as a technology that has shown remarkable performance enhancement in current machine learning fields. A DNN is a deep artificial neural network composed of many hierarchies and can achieve better performance in classification problems because complex nonlinear learning boundaries can be separated better than with conventional artificial neural networks. DNN has shown great potential when applied to speech recognition and image classification, but the cases in which it has been applied to audio event classification is much less. In this work, an audio event classifier using DNN was implemented.

A. *Fundamentals of Deep Neural Network and Auto Encoder*

A DNN is a family of ANNs that consists of many hidden layers. The deep architecture of DNN consists of several hidden layers, and every hidden layer carries a nonlinear transformation from the preceeding layer to next one. In the context of Deep Learning, the idea of pre-training was introduced by Hinton and Salakhutdinov in 2006. The pre-training algorithm make sure of generalization and fine-tuning lets deep networks to train well. Due to multiple hidden layers in DNNs, they possess considerable ability to capture powerful abstracted features from training data. Furthermore, an Auto Encoder (AE) is an unsupervised learning algorithm that utilizes back propagation to set the target output to be its input as shown in Figure 7. The specialty of AE is that it discover patterns in a dataset by identifying the key features. As a result, they are used extensively in feature extraction applications, compression, learning generative models and dimensionality reduction. Figure shows an AE that contains three layers, namely, input layer that represents the input, hidden layer that depicts the learned features and output layer that indicate the reconstruction.

The input and hidden layers of Auto Encoder part are in charge for mapping the input data into hidden representations. However, the hidden and output layers from decoder part are accountable for rebuilding the actual input data from the learned hidden representations. A dataset with the input vector x, the representation vector a_d and reconstruction vector $\hat{x}$ can be represented by Equations (8) and (9), where w and w' signifies the weights of encoder and decoder respectively. The mapping functions are shown with f() and g(). Furthermore, the reconstruction error between x and $\hat{x}$ is given in Equation (10). The overall cost function in the case of m training samples

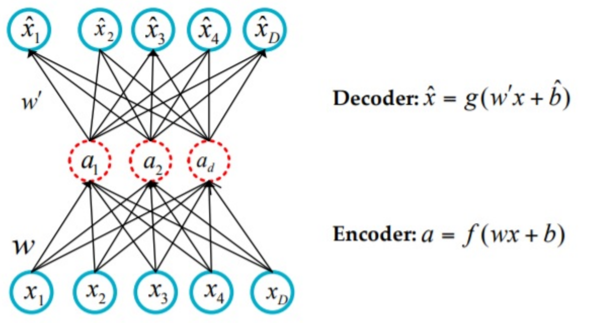


Fig. 7. Structure of a typical Auto Encoder Model

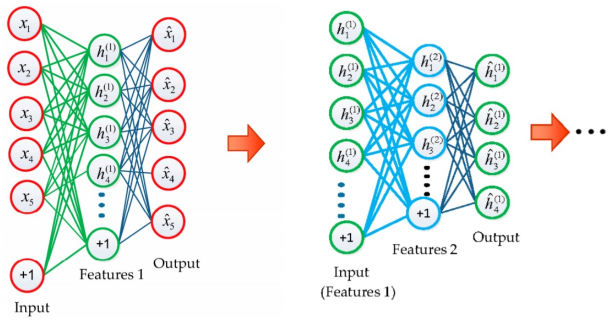


Fig. 8. Layer wise unsupervised pretraining

is defined in Equation (11). In Equation (11), λ is a weight decay term, n_l shows the number of layers in the network and $W_{ji}^{(l)}$ specifies the weight of i^{th} neuron of layer l. The expression b represents a bias term. The first term in Equation (11) represents the reconstruction error, and the second term is a weight decay term, which is a regularization term. The goal of the regularization term is to reduce the magnitude of the weights to keep away overfitting.

$$a = f(wx + b) \tag{8}$$

$$\hat{x} = g(w^{'}x + \hat{b}) \tag{9}$$

$$J(W, b; x, \hat{x}) = \frac{1}{2}\left\|h_{w,b}(x) - \hat{x}\right\|^2 \tag{10}$$

$$J(W, b) = [\frac{1}{m}\sum_{i=1}^{m} J(W, b; x, \hat{x})] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2 \tag{11}$$

B. *Unsupervised feature learning by Stacked Auto Encoders*

A stacked AE (SAE) consists of several layers of Auto Encoders, where the output of every layer is used as the input to the next layer, as shown in Figure 8. By stacking an autoencoder, the deep architectures of input data can be learned effectively. In this work, a greedy layer-wise unsupervised pre-training technique is used to extract features layer by layer in an unsupervised setting. The benefit of this approach is that it not only yields better local minima by proper initialization of the weights in the DNN and training each autoencoder in sequence but also reduces the generalization restrictions of traditional machine learning algorithms. Thus a greedy layer-wise unsupervised pre-training is implemented on SAE on the raw inputs $x^{(k)}$ to extract primary features $h_k^{(1)}$ in the hidden layer, as shown in Figure 8.

After training the first AE, second AE is then trained by using the primary features $h_k^{(1)}$ of the first AE as the raw input to next SAE to extract secondary features $h_k^{(2)}$ from the dataset, as shown in Figure 9. The distinction between two AEs was that, the features that were generated from the first AE is used as the training input to the second. This greedy layer-wise process is known as pre-training. The extracted features $h_k^{(2)}$ from the second autoencoder were used as raw input to a softmax layer. Finally, a deep network is obtined by training the softmax layer and stacking the AE layers. Consequently, the softmax classifier is capable of classifying the different audio events into seven classes. The features acquired by the stacked autoencoders are high-level patterns. The use of these features significantly improves the classification accuracy. In addition to unsupervised pre-training, a fine-tuning strategy is also adopted to further improve the classification results. The whole network is retrained in a supervised manner using a back propagation algorithm as shown in Figure 9. This supervised optimization step is called ”Fine-tuning”.

Fine tuning approach is normally used in deep learning applications, and it greatly improves the performance of a stacked Auto Encoder. It regards every layers of stacked AEs as a single model. Thus, in each iteration, all weights of stacked autoencoder is refined. Stacked Auto Encoders ensure the deep representation of input features with robust feature extraction ability.

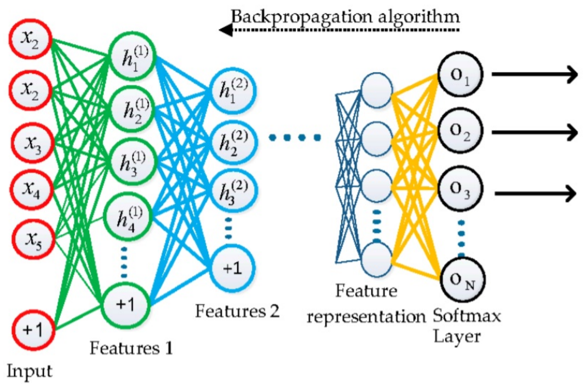


Fig. 9. Supervised Fine-tuning

C. *Softmax Classifier for Classification*

After fine-tuning, the data is classified by feeding the results from the stacked AE to a softmax classifier. The softmax classifier can handle multi label classification problems. Audio tagging deals with classifying audio into seven classes. For a given test input x, softmax classifier estimates the probability p(y = j|x) of each class j = 1, . . . , k, where k = 7, represents seven different classes. The probabilities of each class are calculated using equation 12 as follows:

$$p(y^{(i)} = j|x^{(i)}; \theta) = \frac{e^{\theta_j^T x(i)}}{e^{\theta_l^T x(i)}} \tag{12}$$

where $y^{(i)}$ is the output class corresponding to input vector $x^{(i)}$ and θ_j represents the parameter vector of softmax regression model for the class j, j = 1, 2, 3 . . . , k. The maximum probability of each class is determined by the following equation 13:

$$Class(x^{(i)}) = arg_{j=1,...,k} max p(y^{(i)} = j|x^{(i)}; \theta)$$

(13)

where $x^{(i)}$ represents class with the highest probability. The softmax layer outputs probabilities in the range of [01].

D. Methodology

The DNN method consists of three main parts. In the first part, the audio recordings were downloaded. The dominant sound sources in the audio 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. The second part is the feature extraction process, where Mel Frequency Cepstral Coefficients (MFCC) are extracted from audio. MFCC is used as the basic feature for DNN training. Two Auto Encoders are trained in unsupervised fashion by using greedy layer by layer pre training strategy. During training, the first autoencoder gets trained, and features are extracted in the hidden layer. Likewise, the second AE is trained in similar fashion, except that the features of the first AE is used as inputs of the second AE. The extracted features are then fed into softmax layer, which is trained in a supervised manner. As the AE learns compact data aptly, deep networks can be build by stacking it. Two Auto Encoders are stacked with softmax layer to form a deep model. After constructing a deep model, results are ultimately optimized by retraining the complete model in supervised manner. This procedure is often known as fine-tuning. The classification results reveal that fine-tuning approach produce notable improvement in network performance.

IV. Support Vector Machine for Audio Tagging

Support Vector Machine (SVM) has been successfully used in pattern recognition such as speaker identification, face detection, and text recognition. Compared to other classifiers that separate the data in its original space, such as k Nearest Neighbor(k-NN), Neural Network (NN), and Naive Bayes (NB), SVM maps non-linear separable data to higher dimensional space and performs separation in that space. This characteristic is exploited here and propose a SVM based audio classifier to classify mixed type audio data.

A. Fundamentals of SVM

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for classification or regression tasks. It is seen that, SVM is mostly used in classification tasks. This method plot each data item as a point in n-dimensional space (where n is number of features) with the value of each feature being the value of a specific coordinate. Then, classification is perfrmed by finding the hyper-plane that differentiate the two classes very well. A better separation is obtained by the hyperplane that has the maximum distance to the nearest training-data point of any class.

B. SVM for multilabel classification

In this work, multi-label classification problem is transformed into 7 independent binary classification problem via the one-versus-all scheme. This is a conceptually simple and computationally efficient solution for multi-label classification. A one-versus-all binary SVM classifier is trained for each class by taking all the training data of the class as positive examples and all the rest of the data as negative examples. Given a labeled multi-label training set $D = (x_i, y_i)$ where x_i is the input feature vector for the i^{th} instance, and its label vector y_i is a $0, 1$ valued vector with length K such as K = $|Y|$. If $y_{ik} = 1$, it indicates that the instance x_i is assigned into the k^{th} class; otherwise, the instance does not belong to the k^{th} class. For the k^{th} class (k = 1,....,K), the binary SVM training is a standard quadratic optimization problem as expression 14 below:

$$min_{w_k, b_k, \varepsilon_{ik}} \tfrac{1}{2} ||W_k||^2 + C \sum_{i=1}^{N} \varepsilon_{ik} \tag{14}$$

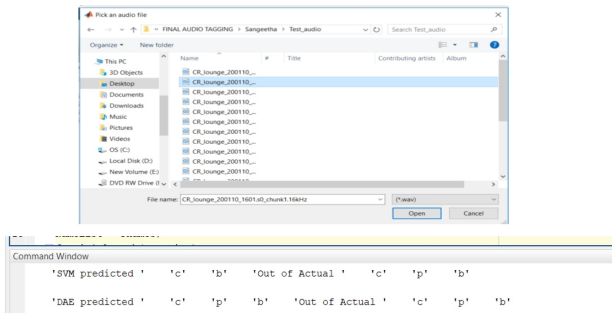
subject to equation 15

$$y_i k(W_k^T x_i + b_k) \geq 1 - \epsilon_{ik}, \epsilon_{ik} \geq 0 \tag{15}$$

where ϵ_{ik} are the slack variables and C is the trade-off parameter. It augments the soft class separation margin. The model parameters w_k and b_k returned by this binary learning problem define a binary classifier associated with the k^{th} class: $f_k(x_i) = w_k^T x_i + b_k$. The set of binary classifiers from all classes can be used independently to predict the label vector $\hat{y}$ for an unlabeled instance $\hat{x}$. The k^{th} component of the label vector $\hat{y_k}$ has value 1 if $f_k(\hat{x}) > 0$, and has value -1 otherwise. The absolute value $|f_k(\hat{x})|$ can be viewed as a confidence value for its prediction $\hat{y_k}$ on instance $\hat{x_k}$.

V. Comparing Deep Auto Encoder with Support Vector Machine

The audio tagging performance of the DNN is compared to SVM, in this work. Figure 10. shows the classification results of the DNN classifier and SVM classifier. The eventwise performance of each of the class labels is shown in figure 12, figure 13, figure 14 and figure 15. Figure 16 shows the Precision Recall (PR) graph and Receiver Operating Characteristic (ROC) curve. ROC and PR curves are typically generated to evaluate the performance of a machine learning algorithm on a given dataset. In ROC space, one plots the False Positive Rate (FPR) on the x-axis and the True Positive Rate (TPR) on the y-axis. The FPR measures the fraction of negative examples that are misclassified as positive. The TPR measures the fraction of positive examples that are correctly labeled. In PR space, one plots Recall on the x-axis and Precision on the y-axis. Recall is identical to TPR, for precision measures that fraction of examples classified as positive that are truly positive. From figures and graphs below, it can be seen that the DNN classifier achieves the best overall performance in most of the per-class results. The overall accuracy of DNN classifier is 97% compared to SVM classifier with an accuracy of 93%. The DNN classifier is useful for audio tagging task and performs better than SVM.

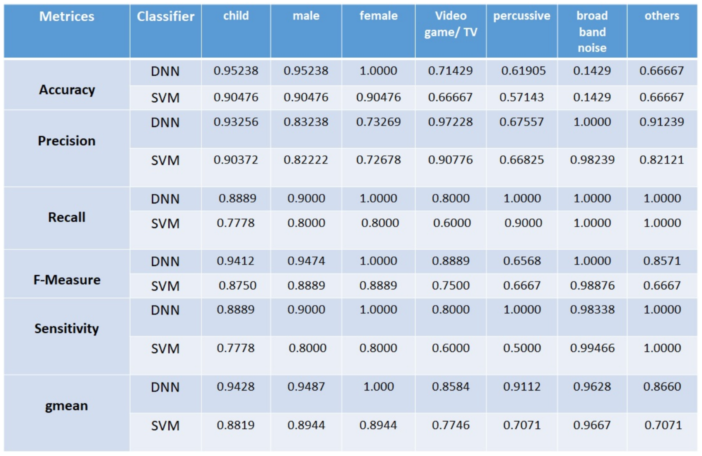


```
'SVM predicted '    'c'    'b'    'Out of Actual '    'c'    'p'    'b'

'DAE predicted '    'c'    'p'    'b'    'Out of Actual '    'c'    'p'    'b'
```

Fig. 10. Audio Tagging Output

TABLE I. Comparison between DNN and SVM methods



Metrices	Classifier	child	male	female	Video game/ TV	percussive	broad band noise	others
Accuracy	DNN	0.95238	0.95238	1.0000	0.71429	0.61905	0.1429	0.66667
	SVM	0.90476	0.90476	0.90476	0.66667	0.57143	0.1429	0.66667
Precision	DNN	0.93256	0.83238	0.73269	0.97228	0.67557	1.0000	0.91239
	SVM	0.90372	0.82222	0.72678	0.90776	0.66825	0.98239	0.82121
Recall	DNN	0.8889	0.9000	1.0000	0.8000	1.0000	1.0000	1.0000
	SVM	0.7778	0.8000	0.8000	0.6000	0.9000	1.0000	1.0000
F-Measure	DNN	0.9412	0.9474	1.0000	0.8889	0.6568	1.0000	0.8571
	SVM	0.8750	0.8889	0.8889	0.7500	0.6667	0.98876	0.6667
Sensitivity	DNN	0.8889	0.9000	1.0000	0.8000	1.0000	0.98338	1.0000
	SVM	0.7778	0.8000	0.8000	0.6000	0.5000	0.99466	1.0000
gmean	DNN	0.9428	0.9487	1.000	0.8584	0.9112	0.9628	0.8660
	SVM	0.8819	0.8944	0.8944	0.7746	0.7071	0.9667	0.7071

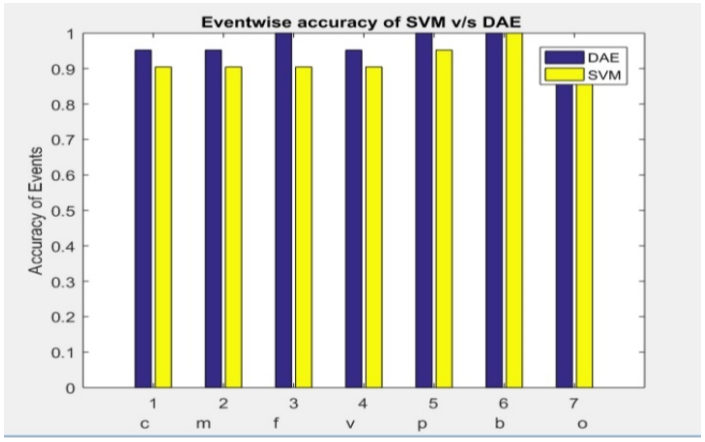


Fig. 12. Eventwise Accuracy of DNN and SVM

VI. Conclusion

Although several techniques are used for audio tagging, research works are still going on for optimization of multi-label acoustic classication. A Deep Neural Network (DNN) framework incorporating unsupervised feature learning and supervised fine tuning is used for handling the multilabel classification task. The DNN classifier achieves the best overall performance in most of the per-class results. The overall accuracy of DNN classifier is 97% compared to SVM classifier with an accuracy of 93%. Hence, the DNN classifier is useful for audio tagging task and performs better than SVM. Therefore it is strongly recommended to use DNN based audio tagging for applications such as, keyword based audio retrieval, multimedia database search, intelligent monitoring systems to recognize activities in the environments and so on. Here, MFCC is used as the basic feature for DNN and SVM training. In future, more audio features can be employed to

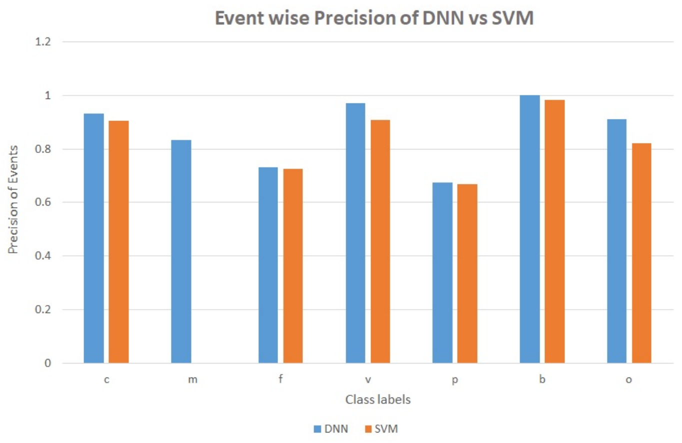


Fig. 13. Eventwise Precision of DNN and SVM

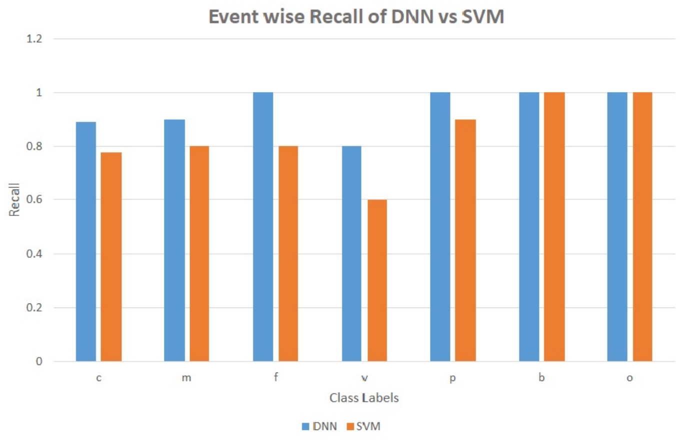


Fig. 14. Eventwise Recall of DNN and SVM

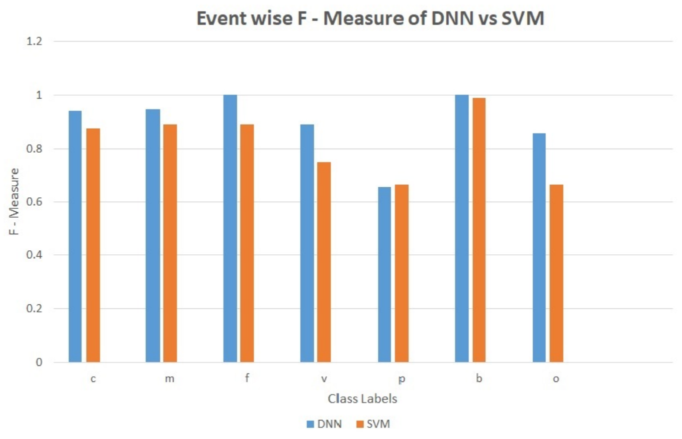


Fig. 15. Eventwise F-Measure of DNN and SVM

classify various sound sources. More audio classes can also be incorporated by extending the work in future for increasing the accuracy of multi label classification.

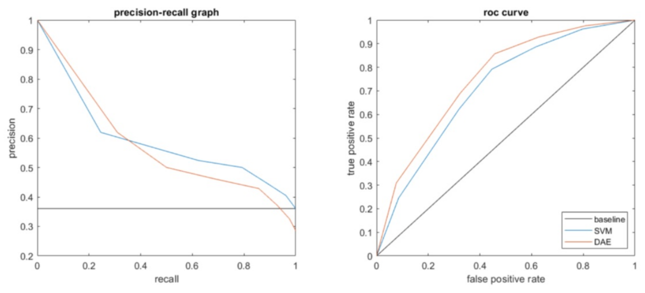


Fig. 16. Precision Recall graph and ROC curve

REFERENCES

[1] Garima Vyas, Malay Kishore Dutta “Automatic Mood Detection of Indian Music Using MFCCs and K-means Algorithm ”, Contemporary Computing (IC3), IEEE Seventh International Conference, 7-9 Aug 2014.

[2] Jurgen T. Geiger, Bjorn Schuller, Gerhard Rigoll “Large Scale Audio Feature Extraction And SVM For Acoustic Scene Classification ”, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2013.

[3] Li Yang and Feng Su, “Auditory Context Classification Using Random Forests ”, Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on 25-30 March 2012.

[4] Alexandros Nanopoulos, Ioannis Karydis, “Know Thy Neighbor: Combining Audio Features And Social Tags For Effective Music Similarity ”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011.

[5] Xi Shao and Changsheng Xu and Mohan S Kankanhalli, “Unsupervised classification of music genre using hidden markov model ”, IEEE International Conference on Multimedia and Expo, 2004.

[6] Remi Foucard, Slim Essid, Mathieu Lagrang and Ga el Richard, “A regressive boosting approach to automatic audio tagging based on soft annotator fusion ”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 25-30 March 2012.

[7] Courtenay V. Cotton and Daniel P. W. Ellis, “Spectral VS. Spectro - Temporal Features For Acoustic Event Detection ”, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) 2011.

[8] Qiuqiang Kong, Yong Xu, Wenwu Wang, Mark D. Plumbley “A Joint Detection -Classification Model For Audio Tagging Of Weakly Labelled Data ”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2017.

[9] Derek Tingle, Youngmoo E. Kim, Douglas Turnbull “Exploring Automatic Music Annotation with ’Acoustically-Objective’ Tags ”, MIR,Philadelphia,Pennsylvania, USA. Copyright 2010 ACM.

[10] Axel Plinge, Rene Grzeszick, and Gernot A. Fink, “A BAG-OF-FEATURES Approach To Acoustic Event Detection ”, IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP), 2014.

[11] Ian McLoughlin, Haomin Zhang, Zhipeng Xie, and Wei Xiao “Robust Sound Event Classification using Deep Neural Networks ”, IEEE/ACM Transactions on Audio, Speech, and Language Processing, Volume: 23, Issue: 3, March 2015.

[12] Yong Xu Qiuqiang Kong Qiang Huang Wenwu Wang Mark D. Plumbley “Convolutional Gated Recurrent Neural Network Incorporating Spatial Features for Audio Tagging ”, Neural Networks (IJCNN), International Joint Conference on 14-19 May 2017.

[13] Jun Takagi, Yasunori Ohishi, Akisato Kimura, Masashi Sugiyama, Makoto Yamada, Hirokazu Kameoka "Automatic Audio Tag Classification Via Semi-Supervised Canonical Density Estimation ", IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP), 2011.

[14] Gordon Wichern, Makoto Yamada, 1Harvey Thornburg, 2Masashi Sugiyama, and Andreas Spanias "Automatic Audio Tagging Using Covariate Shift Adaptation ", IEEE International Conference on Acoustics, Speech and Signal Processing, 2010.

[15] Yong Xu, Qiuqiang Kong, Qiang Huang, Wenwu Wang, Mark D. Plumbley "Attention and Localization based on a Deep Convolutional Recurrent Model for Weakly Supervised Audio Tagging ", http://dx.doi.org/10.21437/Interspeech.2017.

[16] H. Shimodaira "Improving predictive inference under covariate shift by weighting the log-likelihood function, "Journal of Statistical Planning and Inference, vol. 90, 2000.

[17] M. Sugiyama, S. Nakajima, H. Kashima, P. von Bunau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation ", Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2008.

[18] Yong Xu, Qiang Huang, Wenwu Wang, Peter Foster, Siddharth Sigtia, Philip J. B. Jackson, and Mark D. Plumbley, "Unsupervised Feature Learning Based on Deep Models for Environmental Audio Tagging ", IEEE/ACM Transactions on Audio, Speech and Language Processing, VOL. 25, NO. 6, June 2017.