Machine Learning for Sound Event Detection

Shiven Tripathi (190816)
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Shubham Korde (190834)
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

***Abstract*—Audio event detection is performed to tag audio sequences with class labels and event boundaries. In this work we study this task under the bag of frames or sliding window approach which relies on classification models to predict labels for individual frames. Discriminative models like regression classifiers and deep neural networks were tested to yield superior performance on held-out validation sets. Further experiments were also carried out with generative models like GMMs which model inherent distributions in data. Apart from solving for the classification task, we also contribute a novel and diverse audio event detection dataset with the classes for speech and music suitable for Indian populations across age groups and sound categories, on which we train and validate our models.**

I. Acoustic Event Detection

Audio event detection is the recognition of the type or categorical labels of audio, and demarcating the instances of its occurrence. For specific environments and applications, types of events to categorise can vary - for our specific problem statement, we study event detection over the classes of speech, music and silence. For learning based systems, data for building audio event detection models can be labelled as hard labels or soft labels. While hard labels provide stronger supervision with precise annotations for event boundaries, soft labels provide weak supervision of only the class information of the event present in any long audio sequence.

A. Introduction

Audio event detection is the recognition of the type or categorical labels of audio, and demarcating the instances of its occurrence. For specific environments and applications, types of events to categorise can vary - for our specific problem statement, we study event detection over the classes of speech, music and silence. For learning based systems, data for building audio event detection models can be labelled as hard labels or soft labels. While hard labels provide stronger supervision with precise annotations for event boundaries, soft labels provide weak supervision of only the class information of the event present in any long audio sequence.

B. Bag of frames approach

Any arbitrary audio recording can be broken to a sequence of frames, denoting a chunk of signal information across time. Due to the inherent simplicity of the classification task over the audio event detection task, which also requires a regression over boundaries of events, we reduce the problem of event detection to a problem of classification over chunks or frames of uniform size. This approach, which uses a bag of frames to classify the type of a chunk aims to capture short term temporal information stored in bags of frames to assist in audio tagging for a larger sequence. Aggregating the class information for these individual chunks results in the event boundaries for the original sequence, resulting in our solution for the task of audio event detection.

Since this method only relies on short term dependencies, it may fail for complex instances involving class mixing (Eg: certain music might have silence for a chunk length). While more advanced end to end methods would naturally yield superior performance, we have experimented with post processing steps like erosion and dilation to mitigate some of these issues.

II. Dataset

A. Raw Data Collection

To prepare the dataset we looked for music and speech audio on the internet. We tried to ensure our training data is diverse in terms of the intra-class variation. Audio samples from different types of musical instruments, string (guitar and sitar), percussion (tabla and drums) and wind (flute) were used in our training dataset in approximately equal proportions. We also included some audio samples from different kinds of music like rock, jazz and classical. In order to prevent confusion between the music and speech classes, only instrumental music without any vocals was used to prepare dataset. To generate the audio samples for the speech class it was ensured that the training data included voice of people with different ages and genders.

Audio	Details	Duration	Samples
S. Trivedi Hindi speech	Male, 20-30 years	10 min	60
A. Hathway speech	Female, 20-30 years	10 min	60
Audiobook	Male, 30-40 years	10 min	60
R Madhavan speech	Male, 40-50 years	10 min	60
Atif Aslam instrumental	Guitar	10 min	60
Hanuman Chalisa	Sitar, Santoor, Flute	20 min	120
C. Berry Rock music	Drums, Electric Guitar	10 min	60

TABLE I
Details of the dataset used for training

B. Data Augmentation

1) Noise Addition: The raw training data collected for training had little or no noise. However, the validation set had significant noise. Hence, random noise was added to each of

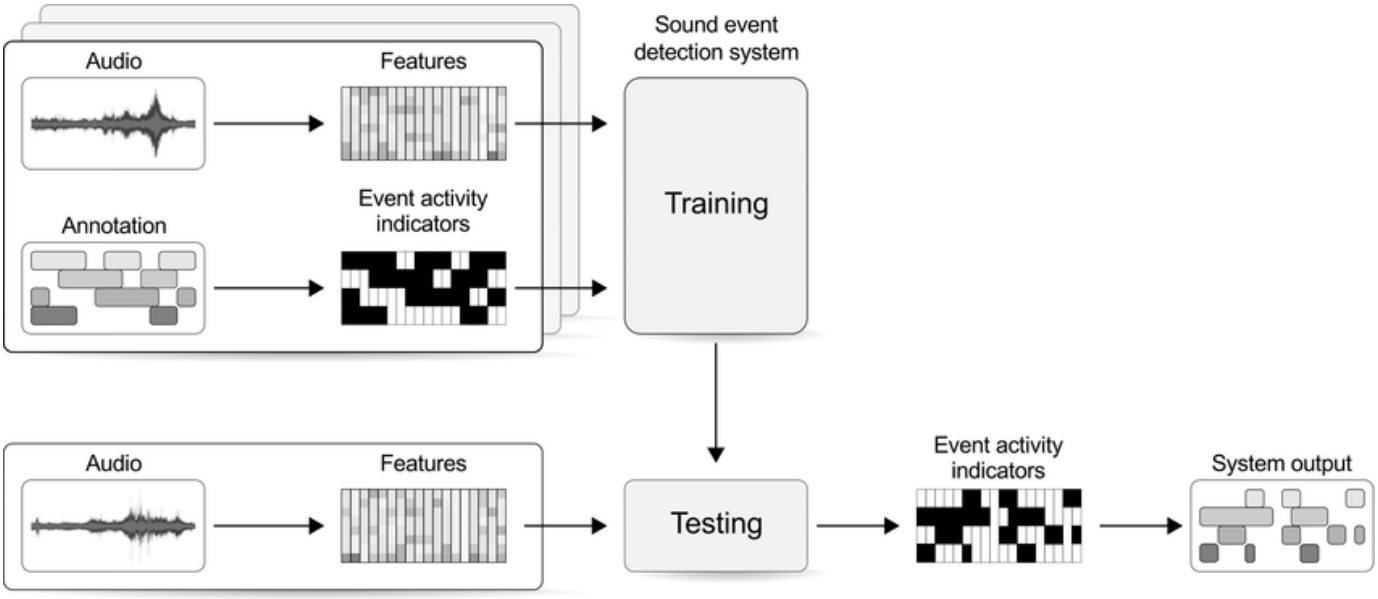

Fig. 1. Standard pipeline for Sound Event Detection systems

the audio sample. The intensity of noise was capped at 5% of the maximum sound intensity of the audio sample.

2) Audio Stretching: The speed of speech and music can also vary significantly within a class. We performed audio stretching, i.e. increasing audio speed and audio compression i.e. decreasing audio speed to generate variations of same data. The audio samples were randomly selected and were stretched by a factor of 0.5 or 1.5. The *effects* package of *librosa* library was used for this purpose.

3) Pitch Shifting: The pitch of the audio samples was shifted by ± 5 hertz to bring about frequency variations in the audio dataset. The *effects* package of *librosa* library was used for this purpose.

C. Spectrogram and Labelling

The final training dataset was generated using the script provided. As we planned to use the "bag of frames" approach to tackle the problem, we need to train our models to perform multi-class classification accurately. Therefore, we used the entry for each window in the spectrogram as the smallest frame or time duration which could be classified. The label for all the frames was assigned to be the same as the audio from which it was extracted.

D. Signal Processing and Feature Extraction

Feature extraction is essential for dimensionality reduction without losing relevant information present in the audio. The raw audio is converted to spectrogram using the short time fourier transform and then converted to decibel scale. Various audio features can be used for classification like Amplitude Envelopes, Zero Crossing Rates, Root Mean Squared Energy, Spectral Bandwidth etc. However, as the audio is provided as a spectrogram, only MFCC features were used which were extracted by first converting the spectrogram from decibel to mel scale and then using the mel spectrogram for feature extraction. All the above functions were performed using in-built librosa functions and utilities. Inorder to have uniformity across data, we performed CMVN (Cepstral Mean Variance Normalization) on the generated MFCC features. Hence, our final training dataset consisted for MFCC features or spectrograms (depending on the model) stored in a sequential data structure index by time, where each feature/spectrogram, corresponds to a time window starting at a time determined by its index and duration determined by the window length and sampling frequency.

$$start\_time = \frac{sample\_index}{total\_samples} \times audio\_duration \quad (1)$$

$$sample\_duration = \frac{window\_length}{sampling\_rate} sec \quad (2)$$

III. ML for Acoustic Event Detection

Following from our initial description of using a bag of frames approach for classifying individual chunks in audio, and aggregating them to generate event boundaries, we now describe the classification models studied and implemented for the classification task.

A. Template Matching

One of the simplest classification methods, Template Matching works on the hypothesis that samples of similar class must satisfy some similarity constraint. From the training data we have, we extract 100 samples for each class, which would serve as templates. These audio templates are transformed to rich feature representations like MFCC spectrograms yielding a vector space in which these templates lie. Then the task of classifying a new sample reduces to finding the class with the closest vector space, which can be identified by computing a

distance metric with the template features, like the Euclidean distance.

While this method offers great simplicity with no learning step required, its accuracy is lacking due to inherent variability in the total distribution of audio samples, which a small template space cannot fully represent.

B. Linear Models

Linear models aim to learn weights, a linear combination of which with the input features results in a score which can be used for decision boundaries for classes. For a categorical classification problem, we use the categorical cross entropy loss to train our linear models, with the weights being updated by the gradient ascent rule on mini-batch inputs. Inference is done on audio, by classifying each bag of frames using the learnt weights of the linear model.

1) Logistic Regression [1]: Recognising that simple linear relations are not adequate to represent the complexities of the data input we have, we extend the linear classifier by applying *SoftMax* non linearity, which is an extension of the *Sigmoid* function for the multi class setting. This allows non linear mappings to be learnt, while also providing a probabilistic interpretation to the model's output as the activation outputs can be interpreted as confidence scores.

C. Deep Neural Networks

To get excellent results with simple linear models, heavy feature engineering is required. In recent years deep neural networks have shown that this step can be avoided by simply increasing the number of parameters in models, as is done in deep neural networks with large number of hidden layers. This makes deep neural networks capable of learning complex non linear functions from the input audio features to their classes, improving performance. However, due to limited computational resources we used MFCC features.

1) Vanilla Deep Neural Network: A feedforward vanilla deep neural network [2] consists of an input layer of dimension of features, an output layer of dimension of classes, and multiple hidden layers consisting of neurons. The weights of these layers transform the inputs to a linear combination upon which an activation function like *tanh* or *ReLU* can be applied. Backpropagation is used to learn the weights and biases for these layers, with the gradient being propagated through the categorical cross entropy loss. Gradient descent on the derivatives obtained by backpropagation leads to learning of optimal weights for classification. In our model, we used two hidden layers with 128 neurons in each layer. The bag of frame comprising of 7 frames (3 before and 3 after) is flattened and passed as an input to the DNN. The bag of frames approach allows the DNN to learn temporal dependencies useful for classification.

2) Convolutional Neural Networks: Convolutional Neural networks (CNN) [3] are widely used in tasks such as object detection, which is also a version of classification problem. In sound event detection problem, the kernels are used to learn patterns in a sequence of spectrogram frames which is characteristic to particular class. The input to the Convolutional Neural Network is a sequence of MFCC features from 7 consecutive frames (two before and two after) of the spectrogram. The kernel size is kept small 3×3 and stride 1×1. The final model which gave the best result on the experimental data, consists of 4 convolutional layers with 16, 32, 64 and 128 kernels respectively, each having ReLU non-linearity. The increasing number of kernels ensure that the network learns sufficient specific features from the basic features for classification. As the size of our input is relatively small dimensional 20×5,max pooling is performed only once with kernel of size 2×2 preventing loss of information. A dropout layer is added, to prevent the model from overfitting. The flattened output is then passed through a series of dense layers to perform the classification. The architecture of dense layers is similar to the previously described DNN architecture.

3) Recurrent Neural Networks: Recurrent Neural Networks [4] have been proven effective in predicting involving time series data as it allows for long term temporal dependence. An approach similar to CNN was used wherein 7 consecutive frames were used to classify a given frame. The training data was reshaped to 7×20 before feeding to the RNN, to preserve time indexed sequences. The model architecture consisted of 2 LSTM layers with 128 neurons in each layer followed by a series of time distributed layers and a dense layer. The LSTMs return sequences for temporal data. The time distributed layers establish the dependence between the current and the previous outputs of the neurons. Dropout layer between the LSTM layers and time distributed layer prevent overfitting.

D. Gaussian Mixture Models

Gaussian Mixture Models [5] learn a cluster representation for a data in an unsupervised fashion. Using maximum likelihood estimates, and the Expectation Maximisation training procedure, we learn parameters for a bunch of gaussians which can properly represent our data. Since our data consists of three classes, we fit the GMM model to three class clusters. From the assignment parameters to the gaussian for a data sample, we get arbitrary class info, which can further be decoded using mode maximisation to find the optimal class label. This allows us to compare performances of our GMM models, by comparison with the actual class labels. Also unlike deep neural networks, GMMs are generative models, and can learn latent probabilistic representations for the data, which can be conditioned to generate new *i.i.d* samples.

IV. Training and Implementation

All models were trained on a train-test split of 80-20 % (except for Template Matching in which training data comprised of only 100 samples). We utilised Google Colab GPU instances supported by VMs with K80 GPU for training models. *Keras* framework was used for creating and training DNN models, while the rest were coded using standard Python libraries like *numpy*. For signal processing and feature extraction, *Librosa* library was used. Adam optimiser was

used for training in DNNs, while SGD was used for Logistic Regression.

V. Inference

A. Decoding

For inference, we applied the same feature transformations as training, and reduced each audio feature sample to bags of frames of features for classification. The bag of frames are used by the model to predict the event corresponding to each time frame. The prediction given by the model is a sequence of events encoded as one hot vectors with dimensions 313×3.

B. Postprocessing

1) Dilation: As the resolution of classification is small $0.03s$ some of the events in a sequence might get classified incorrectly. For Ex: there can be $0.03s$ pauses in running speech or in music. Hence, before the final time boundaries of the events are calculated, the dilation operation is performed on the sequence of output. The event corresponding to 2 frames before and after a given frame are used as a window. If the percentage of the majority event in the window exceeds the 80% threshold, the event for the given time frame is changed to the majority class.

2) Erosion: After performing Dilation, the boundaries of the events are calculated. There can still be some errors in parts of the audio where events are not incorrectly classified. This is due to inherent complexity of the task arising from the similarity in the sound of speech, silence and music in border cases. In such cases, the sequences are highly fragmented, Hence, to improve the precision of our model we discard all the events for which the duration is lesser than $0.5s$.

VI. Evaluation

A. Audio Tagging

For the audio tagging task, the output for each audio sample is a vector with each entry representing whether or not a particular event was observed in the audio. The evaluation metric used for this task is accuracy and F1 score.

B. Event Detection

In addition to their above frame level counterparts, event detection evaluation metrics like substitutions rate, insertion rate and deletion rate are used to compare performance. The evaluation is performed on a per-segment basis for the audio using the script provided.

VII. Results

Classification metrics on individual frames for sequences on the held out validation set have been reported in Table II.

Model	Accuracy	F1 Score
Template Matching	70.36	0.65
Logistic Regression	54.31	0.31
Gaussian Mixture Model	68.04	0.32
Deep Neural Network	99.13	0.97
Convolutional Neural Network	99.45	0.98
Recurrent Neural Network	99.84	0.99

TABLE II
Classification Accuracy and F1 Score on validation set

VIII. Discussion and Conclusion

Deep learning systems demonstrate remarkable performance at classification of audio inputs into their labels. We leverage such an approach to solve the event detection task under the assumption that if a system can generate correct labels for short chunks of inputs then it can aggregate this class information to generate boundaries. Due to high number of parameters which need to be trained for DNNs we find that adequate performance can only be attained for sufficiently large datasets, tagging and labelling of which remains a severe challenge. In particular we note that sound data is sufficiently complex and fails to be modelled appropriately by the most simple linear models and generative models like GMMs.

Investigation with separate validation sets collected by the instructor reveal that even deep models suffer from the problem of domain shift. Significant variations in recording style and even modelling noise can only be captured by huge datasets, and without that, severe performance deterioration is seen. Future work could focus on newer directions of research which seek to learn better representations for data like contrastive learning which also leverages vast unlabelled datasets by learning under self supervision.

References

[1] McCullagh, Peter, and John A. Nelder. Generalized linear models. Vol. 37. CRC press, 1989.

[2] Rumelhart, David E., Geoffrey E. Hinton and Ronald J. Williams. Learning internal representations by error propagation. (1986).

[3] LeCun Y., Haffner P., Bottou L., Bengio Y. (1999) Object Recognition with Gradient-Based Learning. In: Shape, Contour and Grouping in Computer Vision. Lecture Notes in Computer Science, vol 1681. Springer, Berlin, Heidelberg.

[4] Haşim Sak and Andrew Senior and Françoise Beaufays (2014). Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition arxiv 1402.1128

[5] Reynolds D. (2009) Gaussian Mixture Models. In: Li S.Z., Jain A. (eds) Encyclopedia of Biometrics. Springer, Boston, MA.

APPENDIX

Fig. 2. Loss Curve for Logistic Regression

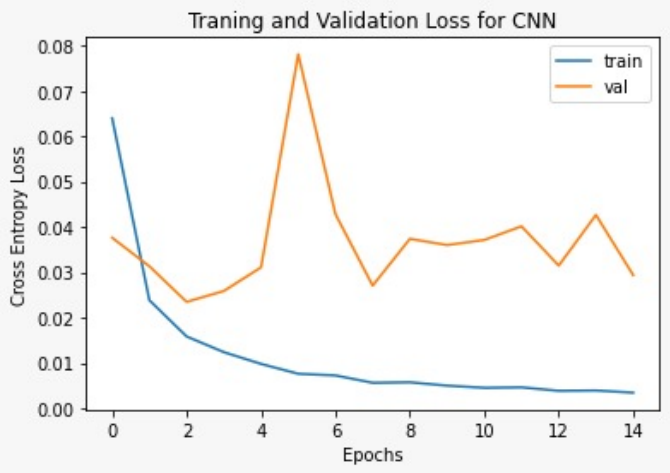

Fig. 3. Loss Curve for Deep Neural Network

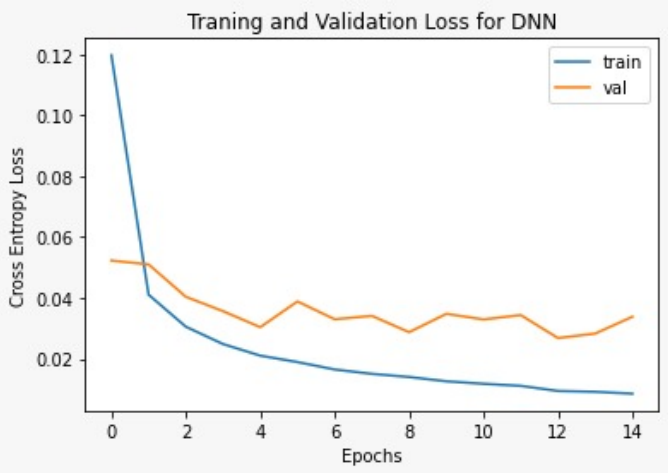

Fig. 4. Loss Curve for Convolutional Neural Network

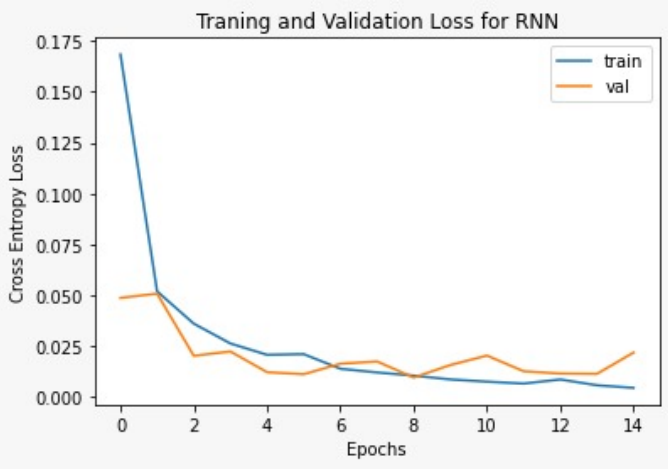

Fig. 5. Loss Curve for Recurrent Neural Network