Single Event Audio Detection

Arth Banka
Indian Institute of Technology Kanpur
`arth20@iitk.ac.in`

Abstract

Single event audio detection is performed to classify audio sequences into an established class. In this report, I present the different computational methods that can be used to perform the task of classifying audio sequences into one of ten possible classes. The report contains introduction, literature survey, my methodology, a tabular representation of the precision, recall, F1 score, confusion matrix obtained for the various methods, observations and discussion.

1 Introduction

Single-event audio detection is the recognition of the type of audio amongst a set of established classes. For specific environments and applications, the types of events to categorise can vary. For my particular problem statement, I study event detection over the classes of ten sounds that are commonly observed in urban residential areas. The classes into which audio signals have been classified into are- 'Bark', 'Meow', 'Siren', 'Shatter', 'Knock', 'Crying _and _sobbing, 'Microwave _oven', 'Vehicle _horn _and _car _horn and _honking', 'Doorbell', 'Walk _and _footsteps'. The data-set provided in the problem statement to train the computational learning based systems consisted of a collection of a thousand spectrograms and their corresponding labels. The data-set consisted of hundred data-sets corresponding to each of the ten classes. The spectrograms were mel-spectrograms that had been calculated by using "torchaudio.transforms.MelSpectrogram". The parameters used for spectrograms were: Nfft = 2048 , nmels = 128 , hop length = 512 , windowing function = hann. The data was then processed and was used to train different learning based systems.

2 Literature Survey

The current state-of-art in pattern recognition problems applied to audio signals does not allow to draw an ultimate conclusion on the single best feature or the best feature set to be used in detection and classification task, irrespective of the kind of audio sources involved Regardless, some of the most common domains in audio analysis are the time, the frequency, the time-frequency and the cepstrum domains. Feature extraction is a very important part in analyzing and finding relations between different things. The data provided by audio files cannot be understood by the models directly. To convert them into a format that is both understandable and with low complexity to feed them to machine learning models, it is required to extract the features that represent the data in a compact and useful way. Machine listening can be described as the auditory equivalent to computer vision, in that it combines techniques from signal processing and machine learning to develop systems that are able to extract meaningful information from sounds. Since the used dataset will be a labeled one, supervised machine learning algorithms will be used to do the audio detection project.

3 Methodology

3.1 Data loading and processing

A function "data _loader" was defined to import the relevant data into the variables X and Y. The mel spectrograms were modified in the original data-set .The Y axis of the spectrograms was transformed into log scale or decibels by using the function "librosa.power _to _db" from the librosa library. In some cases, the spectrograms were further transformed into mel frequency cepstral coefficients (MFCC). MFCC is obtained after performing a discrete cosine transform on the log scaled mel spectrogram. Since the MFCC is a further compression of the mel spectrogram performance is often worse on it compared to a mel spectrogram, however MFCCs are compressed forms of mel spectrograms so they can be used in networks to prevent resource exhaustion errors. Although not relevant in this instance, MFCCs perform better in speech recognition tasks since they are closer to the natural form of human auditory perception. Depending upon the model either MFCCs or spectrograms were used. The spectrograms/MFCCs were zero-padded before loading into X, by using another self defined function "padder" which was designed to ensure that the data was of uniform dimensions. The labels were encoded into one-hot vectors for training the learning based systems. Then the data was divided into test sets and validation data sets using the function "test _train _split" from "sklearn.model _selction" module. The random state variable in the function was chosen such that the validation set fairly represented each of the ten classes. The minimum count of a class was found to be 16 and the maximum was found to be 23. The data being skewed in this instance can result in poor training, therefore it is important to ensure an equitable distribution.

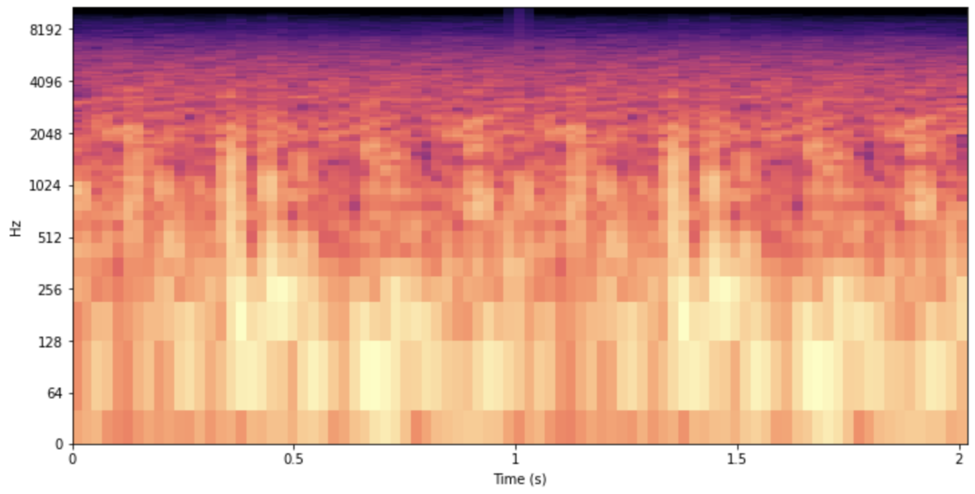


Figure 1: A sample image of a mel spectrogram after being modified to log-scale.

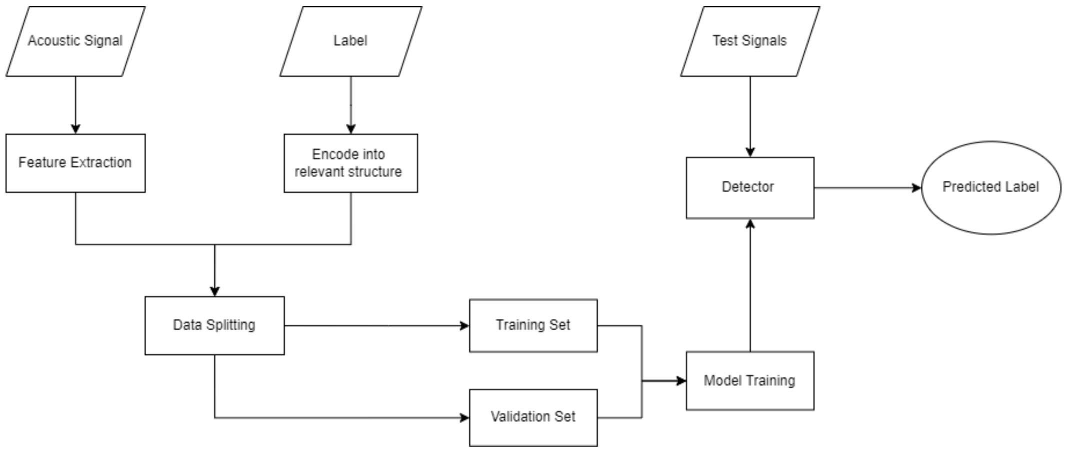


Figure 2: Standard pipeline of a single event audio detector.

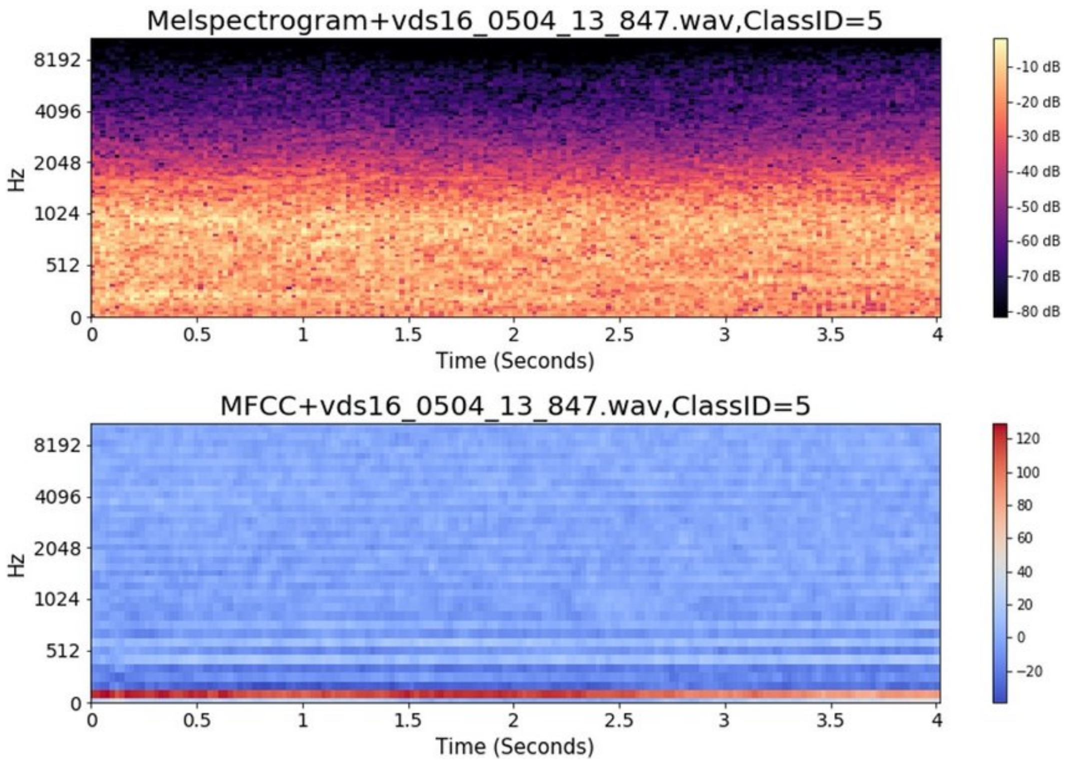
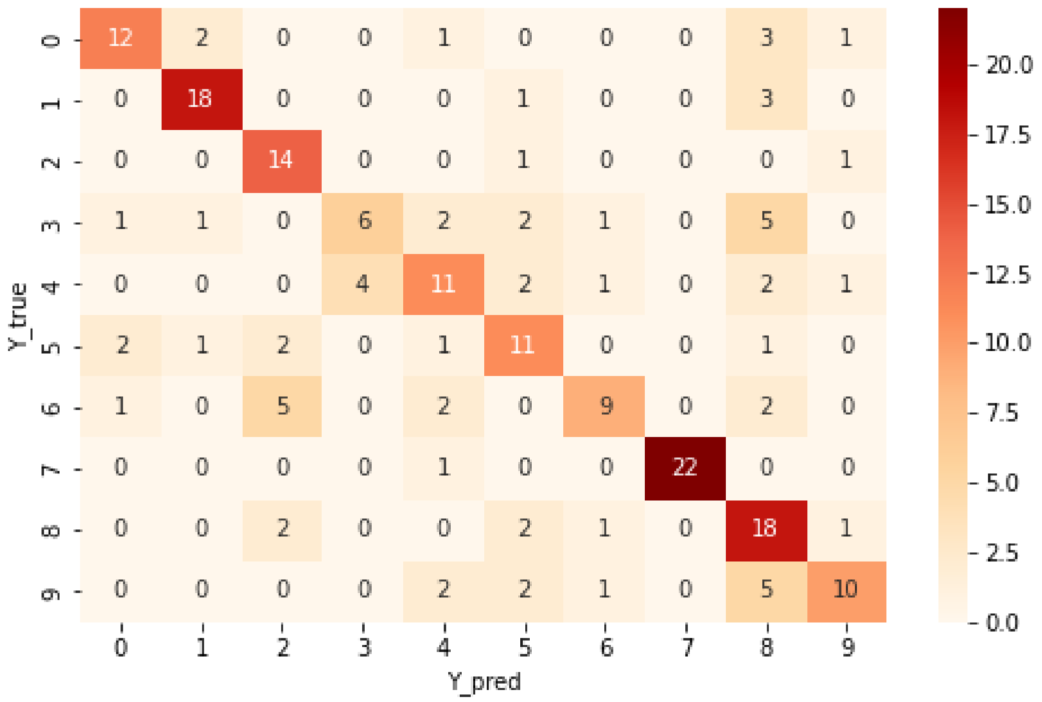


Figure 3: A sample image comparing the image representations of mel spectrograms and MFCCs.

3.2 K-Nearest Neighbors

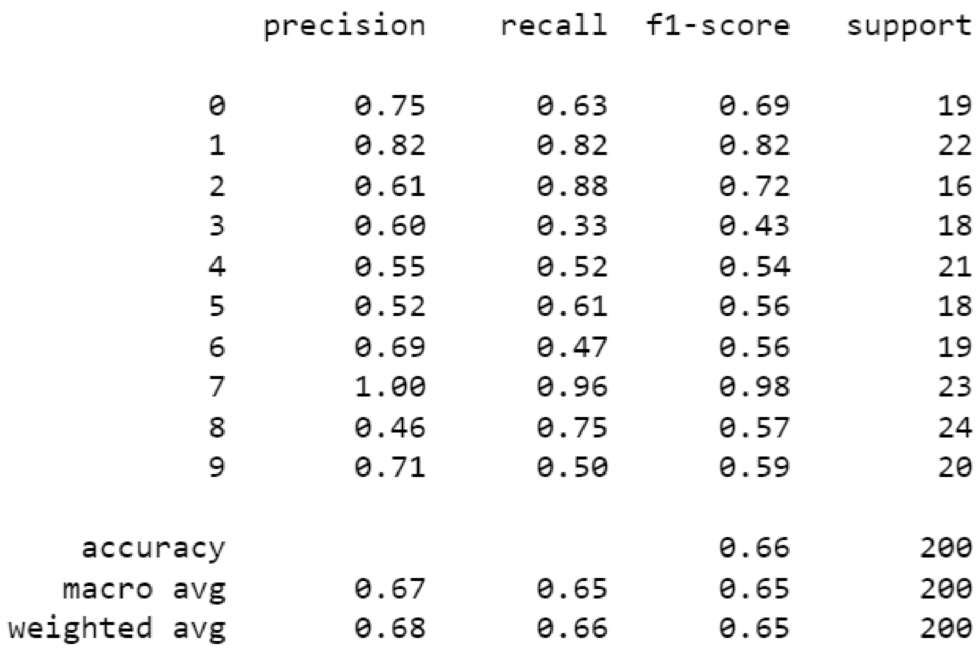
The K-Nearest Neighbors (KNN) algorithm utilises feature similarity to classify data. The model outputs a prediction depending on how closely a given input's features resemble the data in the training set. The output of a KNN model used for multi-class classification will be the number corresponding to the class to which a majority vote of its neighbours attributed the object. KNN models are not as complex as alternate learning-based algorithms and have relatively high accuracy compared to other simple algorithms. However, it is computationally expensive because it stores all of the training data. This makes the prediction stage slow, especially with a large number of samples.

Audio templates are transformed to rich feature representations for the model to work. I have used MFCC spectrograms thus yielding a vector space in which these templates lie and the compressed nature of MFCC prevents resource exhaustion errors when using KNN. 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.



	0	1	2	3	4	5	6	7	8	9
0	12	2	0	0	1	0	0	0	3	1
1	0	18	0	0	0	1	0	0	3	0
2	0	0	14	0	0	1	0	0	0	1
3	1	1	0	6	2	2	1	0	5	0
4	0	0	0	4	11	2	1	0	2	1
5	2	1	2	0	1	11	0	0	1	0
6	1	0	5	0	2	0	9	0	2	0
7	0	0	0	0	1	0	0	22	0	0
8	0	0	2	0	0	2	1	0	18	1
9	0	0	0	0	2	2	1	0	5	10

Figure 4: The confusion matrix heat map for the KNN model for validation set during training.



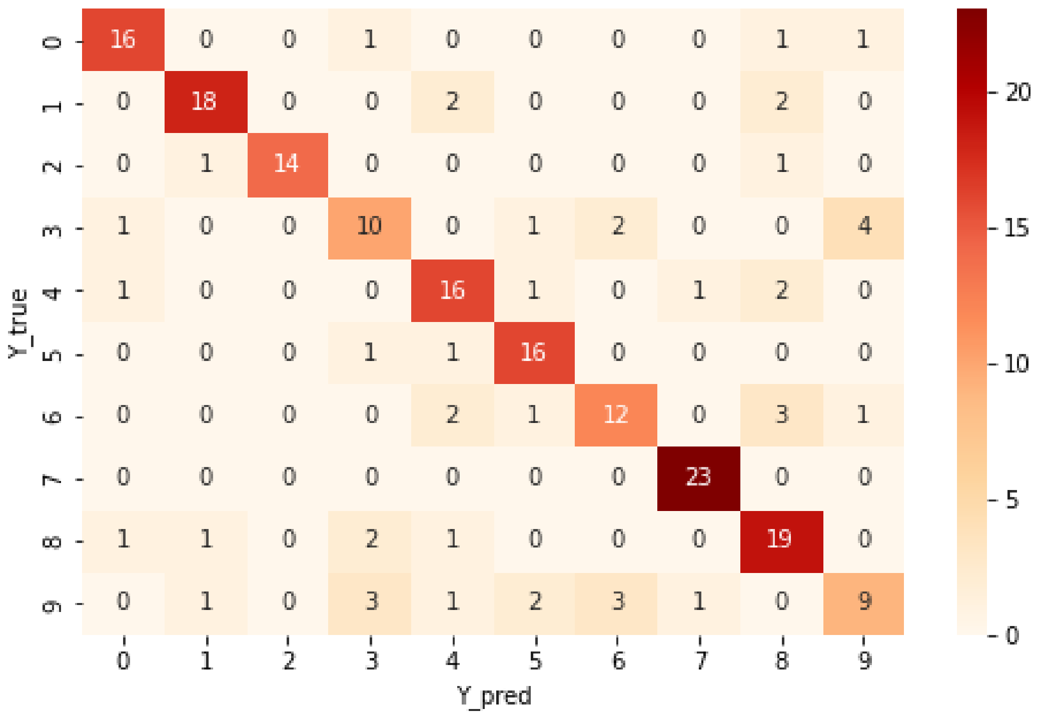
```
              precision    recall  f1-score   support

           0       0.75      0.63      0.69        19
           1       0.82      0.82      0.82        22
           2       0.61      0.88      0.72        16
           3       0.60      0.33      0.43        18
           4       0.55      0.52      0.54        21
           5       0.52      0.61      0.56        18
           6       0.69      0.47      0.56        19
           7       1.00      0.96      0.98        23
           8       0.46      0.75      0.57        24
           9       0.71      0.50      0.59        20

    accuracy                           0.66       200
   macro avg       0.67      0.65      0.65       200
weighted avg       0.68      0.66      0.65       200
```

Figure 5: The classification report obtained after running KNN on validation set.

3.3 ANN Model

Artificial neural networks are capable of learning complex non-linear functions from the input audio features to their classes, improving performance. However, due to limited computational resources, I have used MFCC features. A feed-forward artificial neural network consists of an input layer of the dimension of features, an output layer of the 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. Back-propagation 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 back-propagation leads to learning optimal classification weights.

The network architecture for the neural network was obtained after hyper-parameter variation. I used a layer to flatten the input MFCC data, followed by four dense layers and one dense output layer in my model. The first four layers had 1024, 512, 256, and 128 neurons, respectively, each having ReLU activation. The output layer had ten neurons to represent the ten different classes.



	0	1	2	3	4	5	6	7	8	9
0	16	0	0	1	0	0	0	0	1	1
1	0	18	0	0	2	0	0	0	2	0
2	0	1	14	0	0	0	0	0	1	0
3	1	0	0	10	0	1	2	0	0	4
4	1	0	0	0	16	1	0	1	2	0
5	0	0	0	1	1	16	0	0	0	0
6	0	0	0	0	2	1	12	0	3	1
7	0	0	0	0	0	0	0	23	0	0
8	1	1	0	2	1	0	0	0	19	0
9	0	1	0	3	1	2	3	1	0	9

Figure 6: The confusion matrix heat map for the ANN model for validation set during training.

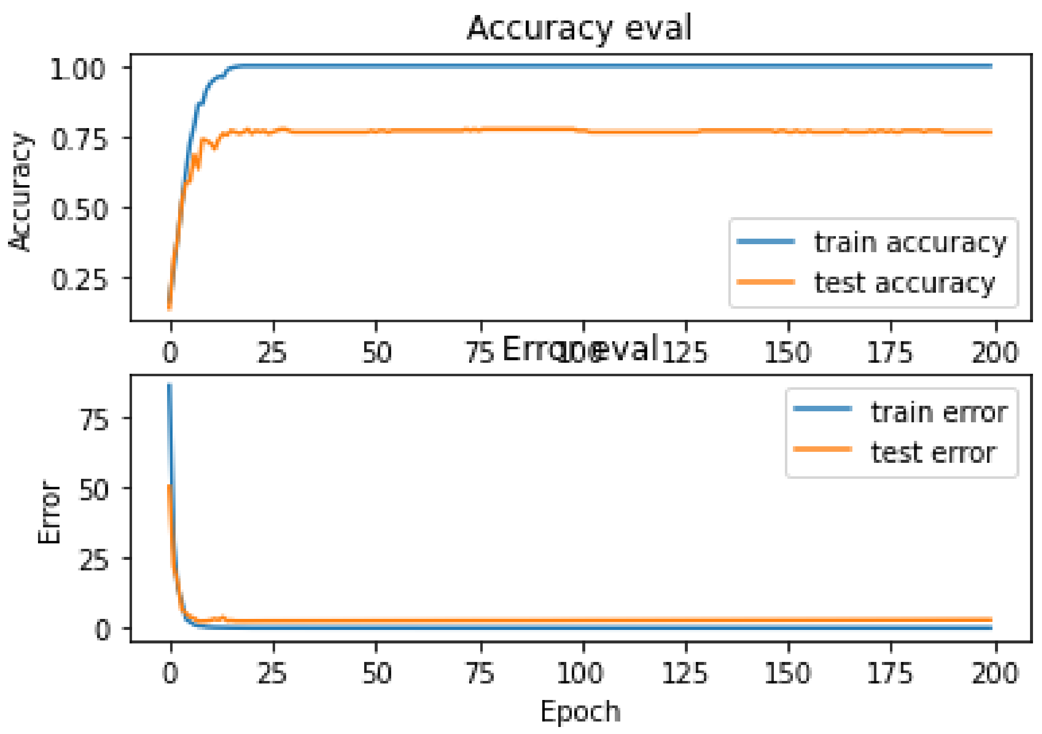
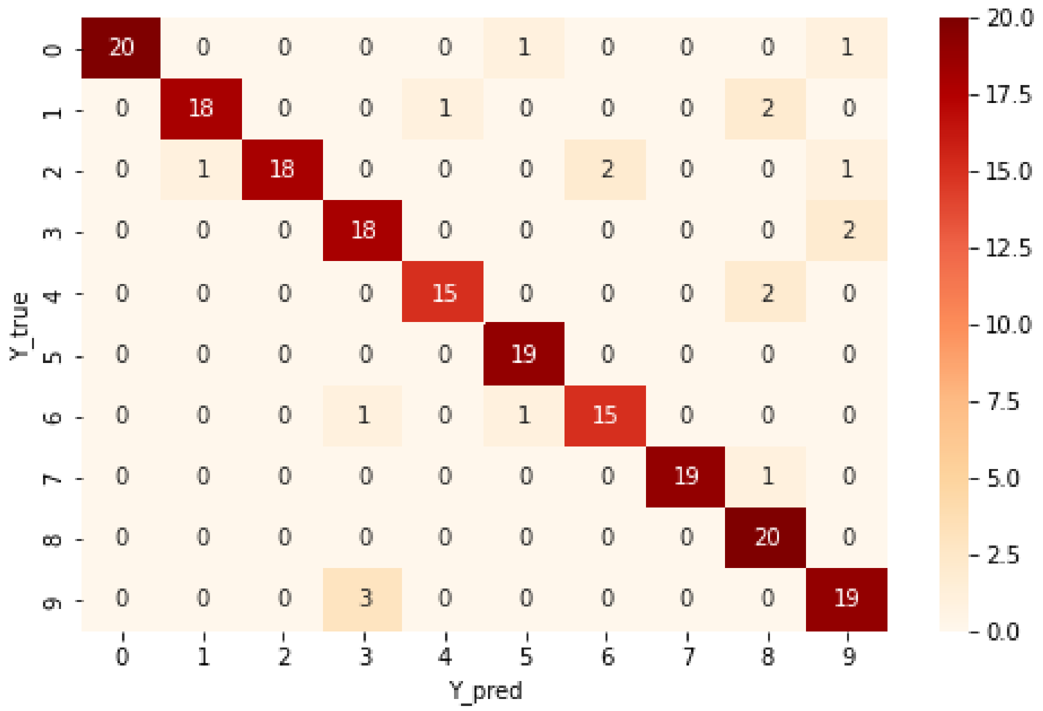


Figure 7: The error/loss and accuracy as a function of epoch number during ANN training.

3.4 CNN Model

Convolutional Neural networks (CNN) are widely used in tasks such as object detection, which is also a version of the classification problem. In sound event detection problems, the kernels are used to learn patterns in a sequence of spectrogram frames characteristic to a particular class. In CNN, we input an image of height × width × dimension. Therefore we modify the input data by adding an extra dimension using the reshape function. The input data used in this case were log scaled mel spectrograms since MFCCs are more decorrelated which does not mesh well with strong classifiers such as CNN.
The model architecture was arrived upon after hyper-parameter variation and trial and error. The final network architecture that gave the best result on the data contains consisted of five convolution layers of 32, 64, 64, 128, 256 kernels respectively, each having ReLU (Rectified Linear Unit) activation linearity. Each layer has strides of (1,1) and kernel size of (3,3) except the second convolution layer which has utilises a kernel of size (2,2). Max pooling has been used to since our data size is quite large and kernel size of (2,2) prevents loss of information. The increasing number of kernels ensure that the network learns sufficient specific features from the basic features for classification. Dropout layers have been added to prevent over-fitting during training. The flattened output is then passed through a dense layer of 512 neurons with the ReLU activation and finally into a dense layer of 10 neurons with the softmax activation. Batch size of 32 was used for training and batch normalization was used in the layers to ensure that inputs were standardised for each batch. The loss function used in the model was categorical cross entropy and optimizer was 'adam'.



	0	1	2	3	4	5	6	7	8	9
0	20	0	0	0	0	1	0	0	0	1
1	0	18	0	0	1	0	0	0	2	0
2	0	1	18	0	0	0	2	0	0	1
3	0	0	0	18	0	0	0	0	0	2
4	0	0	0	0	15	0	0	0	2	0
5	0	0	0	0	0	19	0	0	0	0
6	0	0	0	1	0	1	15	0	0	0
7	0	0	0	0	0	0	0	19	1	0
8	0	0	0	0	0	0	0	0	20	0
9	0	0	0	3	0	0	0	0	0	19

Figure 8: The confusion matrix heat map for the CNN model for validation set during training.

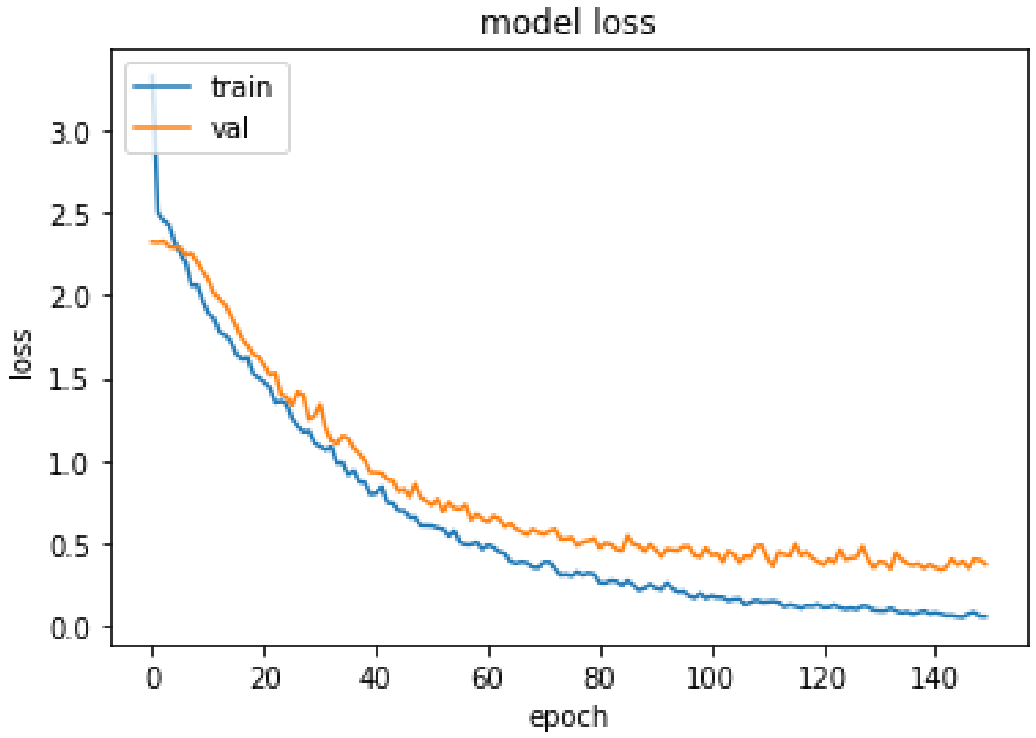


Figure 9: The error/loss as a function of epoch number during CNN training.

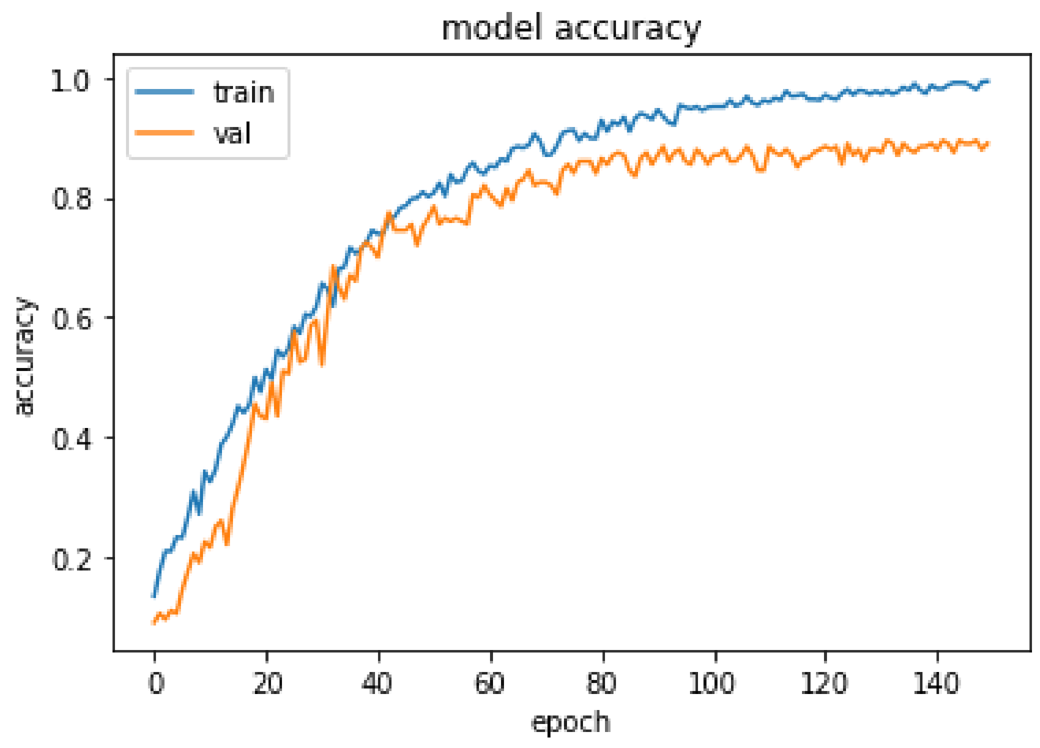


Figure 10: The accuracy as a function of epoch number during CNN training.

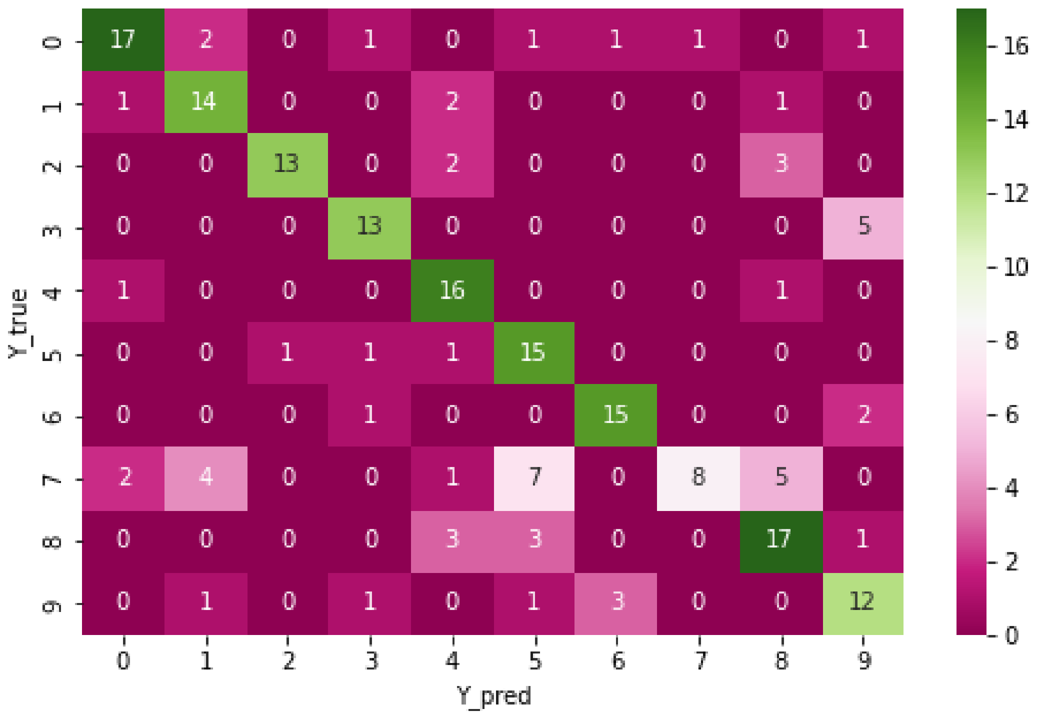
4 Results and Confusion Matrix

A test set consisting of 201 samples was released to verify the performance of the learning based systems. The results obtained have been shown in tabular form below.

CNN Evaluation Metrics		
Metrics	Score(Validation Set)	Score(Test Samples)
Accuracy	0.9100	0.6965
Precision	0.9403	0.7753
Recall	0.9152	0.6959
F1 Score	0.9273	0.7329

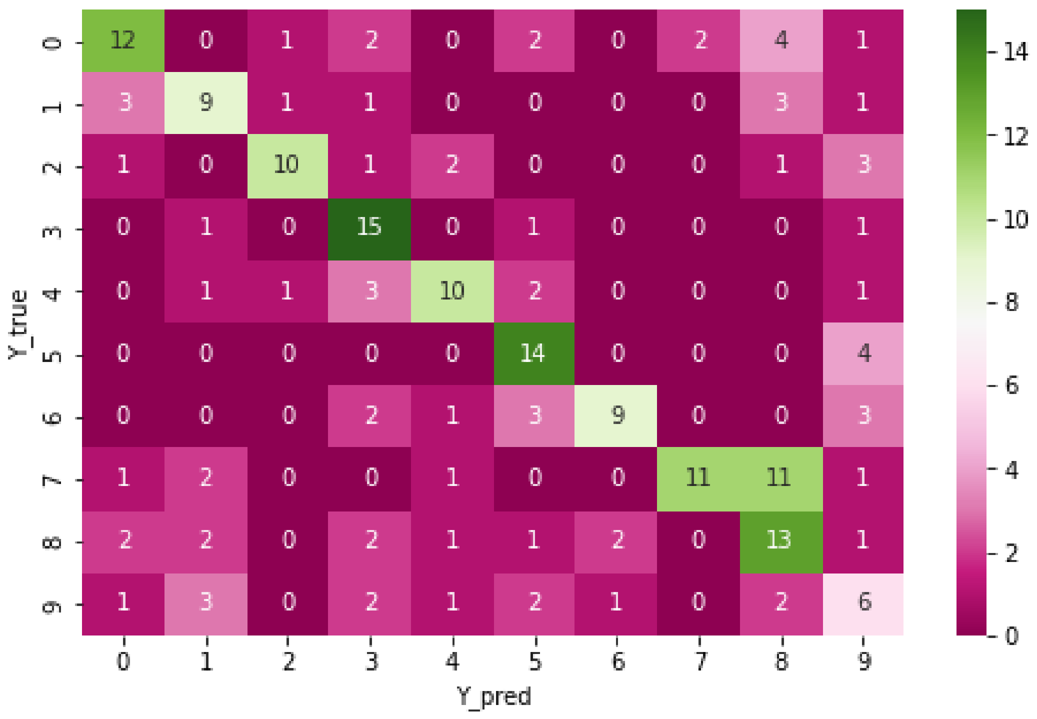
ANN Evaluation Metrics		
Metrics	Score(Validation Set)	Score(Test Samples)
Accuracy	0.7750	0.5423
Precision	0.7009	0.5477
Recall	0.6758	0.5322
F1 Score	0.6879	0.5398

KNN Evaluation Metrics		
Metrics	Score(Validation Set)	Score(Test Samples)
Accuracy	0.66	0.38
Precision	0.68	0.39
Recall	0.48	0.31
F1 Score	0.57	0.32



Y_true \ Y_pred	0	1	2	3	4	5	6	7	8	9
0	17	2	0	1	0	1	1	1	0	1
1	1	14	0	0	2	0	0	0	1	0
2	0	0	13	0	2	0	0	0	3	0
3	0	0	0	13	0	0	0	0	0	5
4	1	0	0	0	16	0	0	0	1	0
5	0	0	1	1	1	15	0	0	0	0
6	0	0	0	1	0	0	15	0	0	2
7	2	4	0	0	1	7	0	8	5	0
8	0	0	0	0	3	3	0	0	17	1
9	0	1	0	1	0	1	3	0	0	12

Figure 11: The confusion matrix heat map for the CNN model for ground truth/test set.



Y_true \ Y_pred	0	1	2	3	4	5	6	7	8	9
0	12	0	1	2	0	2	0	2	4	1
1	3	9	1	1	0	0	0	0	3	1
2	1	0	10	1	2	0	0	0	1	3
3	0	1	0	15	0	1	0	0	0	1
4	0	1	1	3	10	2	0	0	0	1
5	0	0	0	0	0	14	0	0	0	4
6	0	0	0	2	1	3	9	0	0	3
7	1	2	0	0	1	0	0	11	11	1
8	2	2	0	2	1	1	2	0	13	1
9	1	3	0	2	1	2	1	0	2	6

Figure 12: The confusion matrix heat map for the ANN model for ground truth/test set.

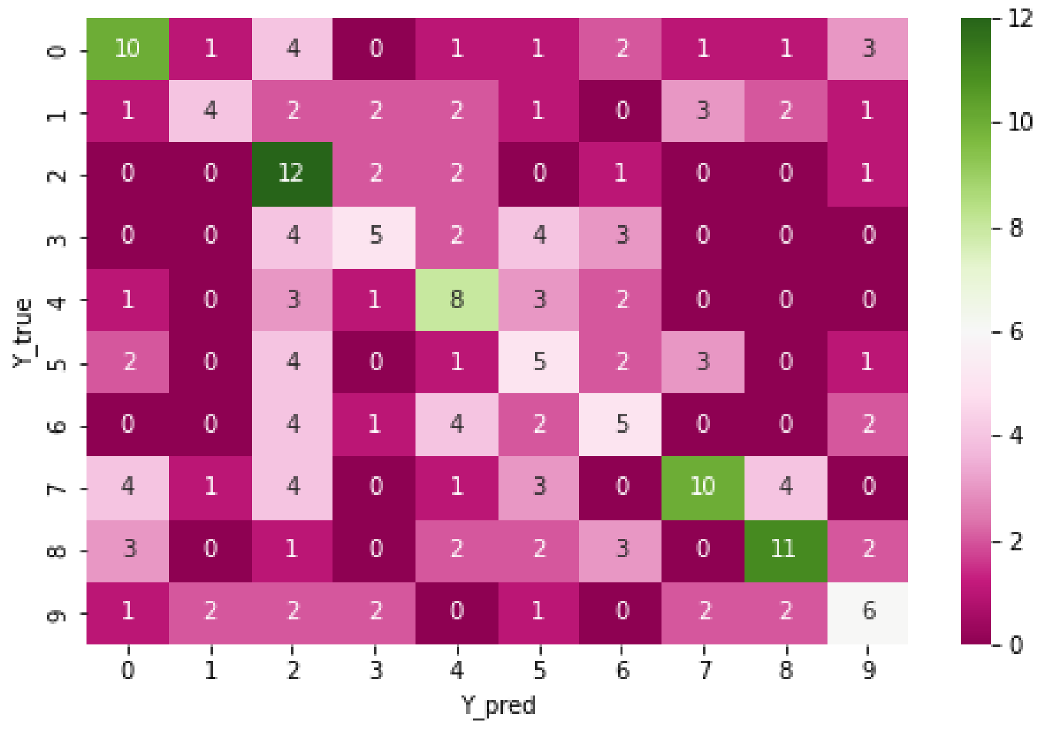


Figure 13: The confusion matrix heat map for the ANN model for ground truth/test set.

5 Discussion and Conclusion

Several different methods have been implemented for single event audio detection. Deep learning systems demonstrate descent performance at classification of audio inputs into their respective classes. Significant variations in recording style and even modelling noise can only be captured by huge data-sets, and without that, severe performance deterioration is seen.

We had the best results with our CNN as can be seen in the figures and tables above. The ANN proved to be the second best metric, however the CNN consistently beat it by a few percentage points across the data-sets. Part of this success stems from the CNN sharing its weights across multiple features. Therefore it was not as susceptible to overfitting as the other models I tried. Due to high number of parameters which need to be trained for ANNs we find that adequate performance can only be attained for sufficiently large datasets.

While the CNN does have better performance and is more robust, we should also note that it can be as much as three times slower than the ANN. While KNN offered 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.

A very interesting application of the data-set given to us was its use in security systems in households and to detect suspicious activity in public places. 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.

6 References

1. Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, Quoc V. Le, “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition”, Proc. Interspeech 2019, 2613-2617

2. Naoya Takahashi, Michael Gygli, Beat Pfister, Luc Van Gool, "Deep Convolutional Neural Networks and Data Augmentation for Acoustic Event Detection", Interspeech 2016

3. Haytham Fayek. 2016. Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between. In http://haythamfayek.com/blog/ Online. URL=http://haythamfayek.com/2016/04/21/speechprocessing-for-machine-learning.html

4. Aaqid Sayeed. 2016 Urban Sound Classification, Part I, Part II. https://aqibsaeed.github.io/2016-09-03-urban-soundclassification-part-1/

5. Urban Sound Event Classification for Audio-Based Surveillance Systems, João Pedro Duarte Galileu