Machine Learning Engineer Nanodegree

Capstone Report

Kannappan Natarasan
May 12th 2018

I . Definition

Project Overview

This project is to solve a challenge hosted by Kaggle challenges, to build an audio tagging system using a dataset of audio files covering a wide range of real-world environments. Sounds in the dataset include things like musical instruments, human sounds, domestic sounds, and animals from Freesound's library, annotated using a vocabulary of more than 40 labels from Google's AudioSet ontology. To predict sounds effectively the model should recognize an increased number of sound events of very diverse nature, and to leverage subsets of training data featuring annotations of varying reliability (see Data section for more information). https://www.kaggle.com/c/freesound-audio-tagging

Problem statement

From set of human labeled sound files a model has to be built by training with train dataset in .wav format. Train dataset has sound files which can be converted into 1D array and labels are provided from csv file Train.csv. CNN techniques can be used to train the model. Human labeled data set can be split into train validation and test, so that prediction accuracy can be measured.

Evaluation metrics

This prediction model built in this project would be evaluated by Mean Average Precision@3 , thus the predication accuracy can be calculated based on first three predicted files.

$$MAP@3 = \frac{1}{U} \sum_{u=1}^{U} \sum_{k=1}^{min(n,3)} P(k)$$

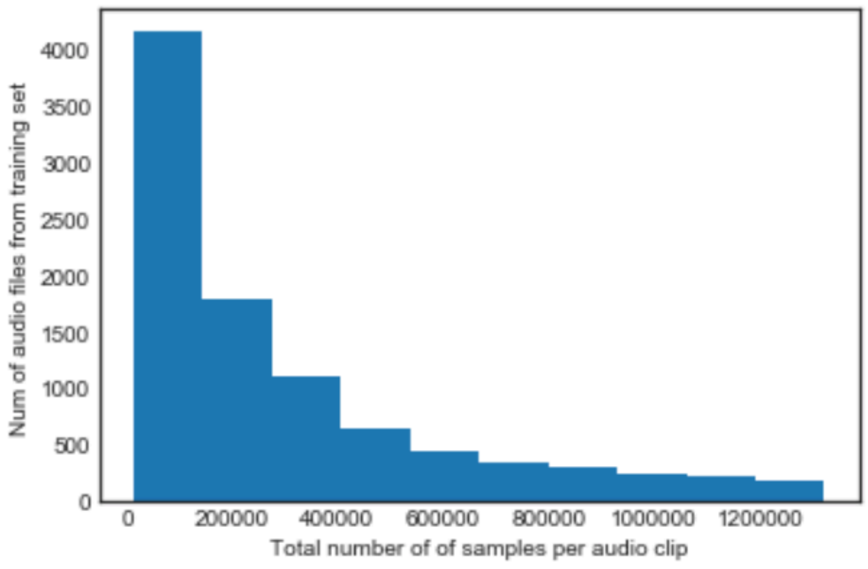
U	:	the number of scored audio files in the test data,
P(k)	:	the precision at cutoff k
n	:	the number predictions per audio file.

II . Analysis

Data Exploration

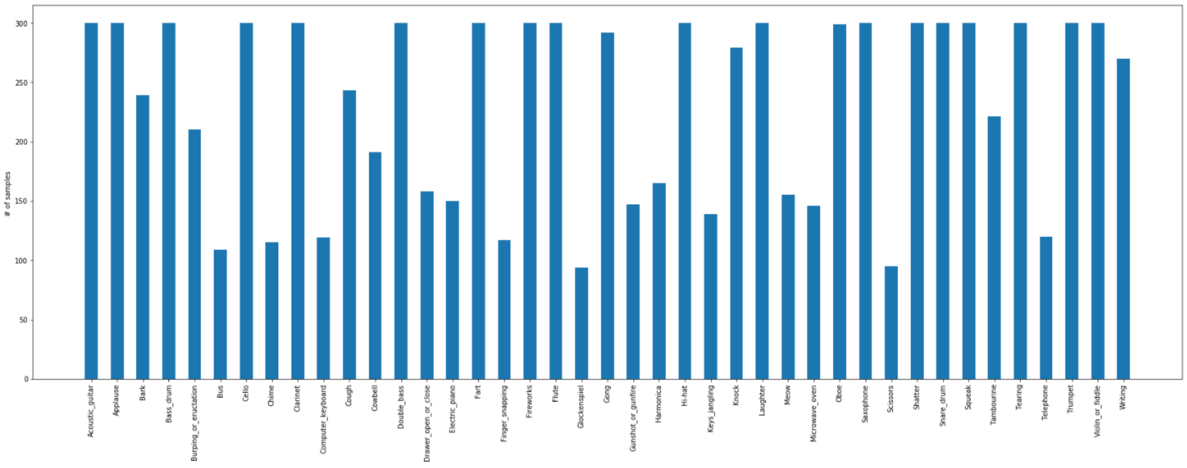
This challenge provides following dataset,

Traing set : There are audio files consisting of 41 labels and total 9473 audio files . Length of audio is decided by number of audio samples per file. Following histogram provides number of number of audio clips with a bucket of number of audio samples in each file.



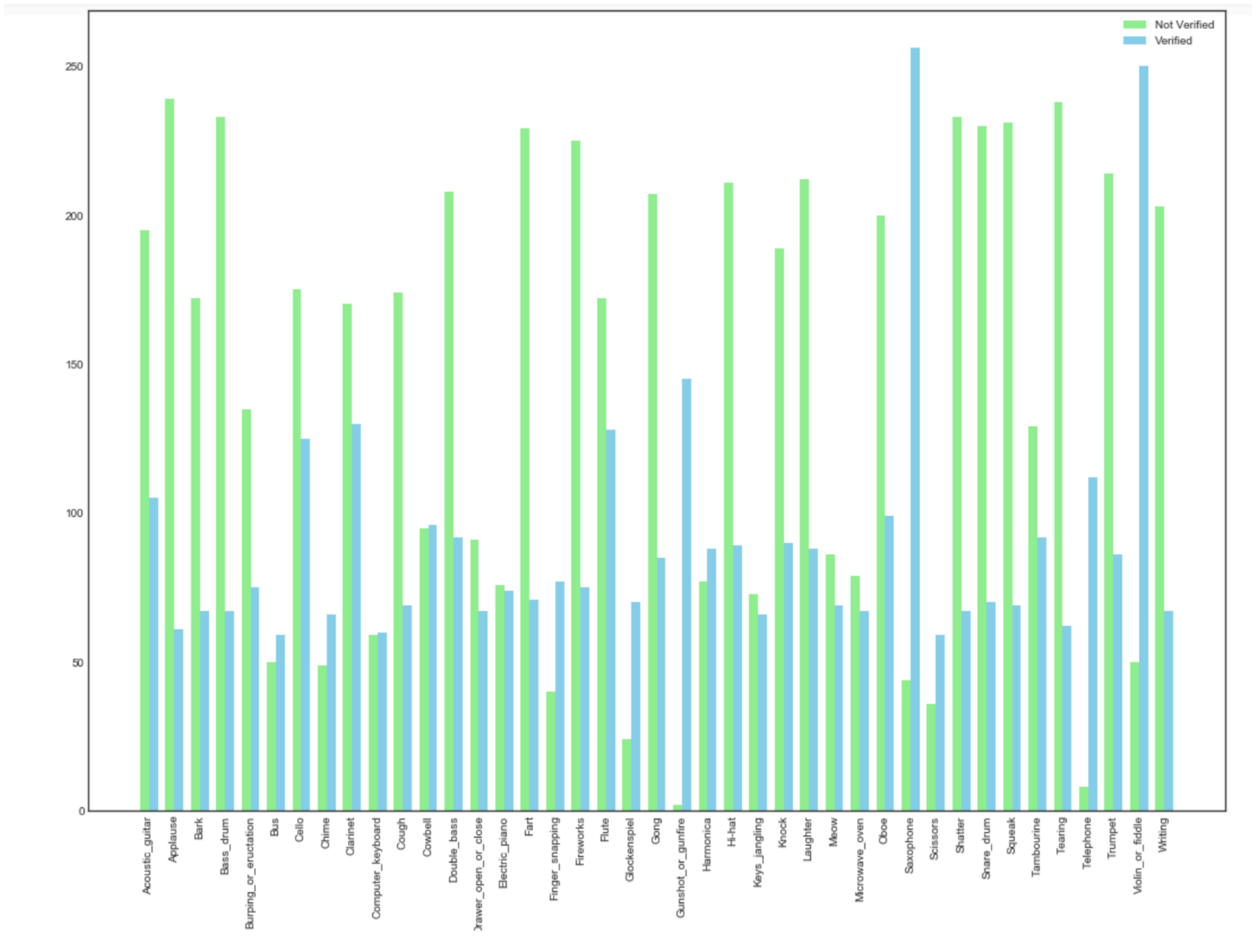
Close to 4k files have 200k audio samples, this is half of the dataset. In remaining half close to 2k files has samples between 200k and 400k . Less than 500 files have samples as high as 1200k.

Visualization would be useful to understand the dataset . Following visualization shows number of samples on each label category.



Labels like Acoustic guitar,Applause,Bass drum,Cello, Clarinet, Double bass, etc has 300 files each, it is likely these labels are predicted with better accuracy.

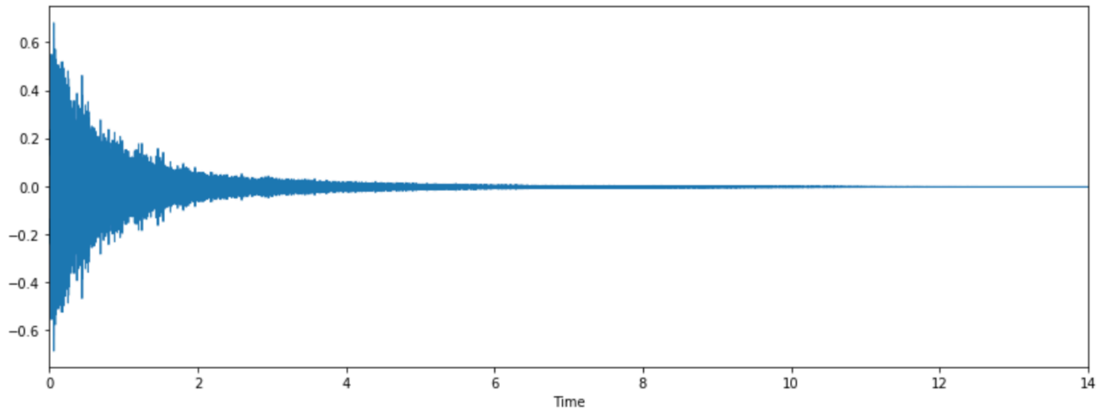
Following graph provides labels number of files with human verified and not.

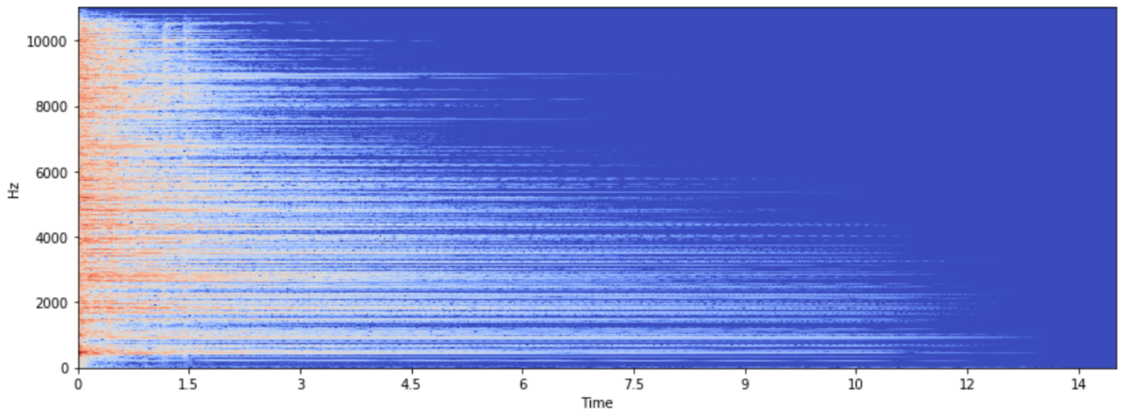


Exploratory Visualization

Analyzing visuals of individual audio files would be useful to track mechanisms of how a machine learning algorithm could use these data to build learning models.

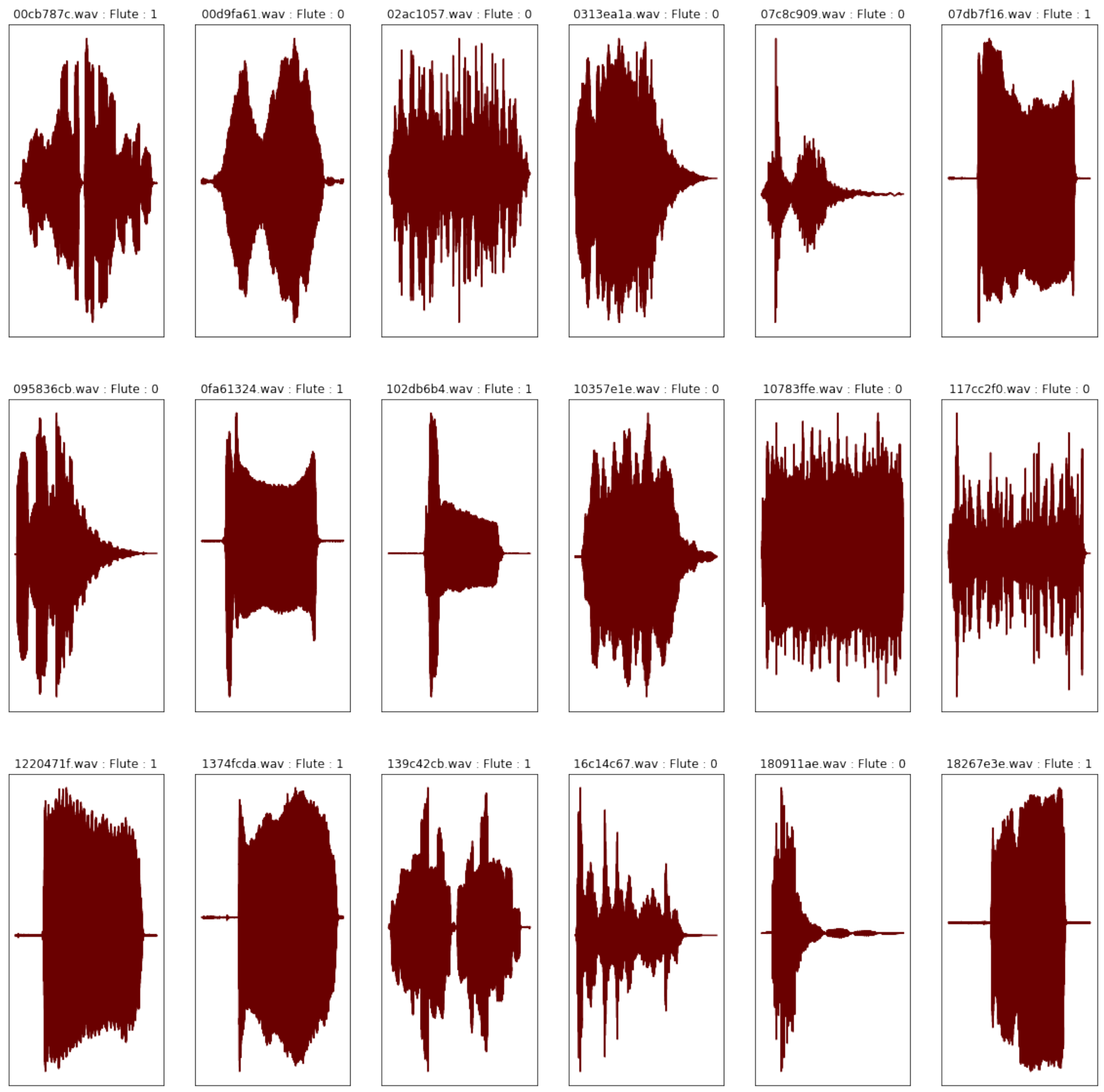
Following helps to visualize time vs samples.





Following helps to visualize many sound files from same label,

Sound Files from Flute audio:



Algorithms and Techniques

Since sound files are temporal in nature and single dimension arrays, keras deep learning algorithms are proven to be efficient.
https://keras.io/getting-started/sequential-model-guide/

Benchmark model

Since this dataset has 41 labels and they are well distributed approaches like random guess or naïve wont work. So I've chosen to use the same a CNN model with a single epoch . By this approach prediction accuracy is 5.2%

III . Methodology

Model Building

An CNN is architecture is built with following model

Input layer:

This is initial layer the window has to be equal to input file array size .

Conv1D: This is Convolution layer number of filters size of slide window , padding strategy and activation methods are set here .

Max Pooling : This layer helps to consolidate many neurons to single, this can be selected from average pooling or max pooling.

Dropout : This layer is to dropout un interesting features and deepen the network only with most impacting features.

Dense : This are final layers to predict labels from derived features.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 16000, 1)          0         
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 15992, 16)         160       
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 15984, 16)         2320      
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 999, 16)           0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 999, 16)           0         
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 997, 32)           1568      
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 995, 32)           3104      
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 248, 32)           0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 248, 32)           0         
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 246, 32)           3104      
_________________________________________________________________
```

```
conv1d_6 (Conv1D)            (None, 244, 32)           3104      
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 61, 32)            0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 61, 32)            0         
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 59, 256)           24832     
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 57, 256)           196864    
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 256)               0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 64)                16448     
_________________________________________________________________
dense_2 (Dense)              (None, 1028)              66820     
_________________________________________________________________
dense_3 (Dense)              (None, 41)                42189     
=================================================================
Total params: 360,513
Trainable params: 360,513
Non-trainable params: 0
_________________________________________________________________
```

Implementation

Above model has been executed with StratifiedKFold with fold count 2

IV . Results

Model Evaluation and Validation

Model has been evaluated with Average Precision@3 , since APK@3 considers next 3 predicted values , I prefer this than other accuracy measures .

V . Conclusion

I've started preparing data using packages like librosa,pyaudio and scipy which are very helpful to process sound files and convert them to single dimension array. Followed exploratory analysis with numpy and matplotlib packages. Used custom normalization to normalize data. Built CNN model using keras packages. Finally able to derive a model with AP@3 accuracy of 55% which is way better than bench mark.