Machine Learning Engineer Nanodegree

Capstone Report

Kannappan Natarasan
May 12th 2018

I . Definition

Project Overview

Able to understand and classify a sound file by an application fascinated me, since I'm planning to build an App which can apply these techniques on machines in factory thus those can be monitored remotely and any mechanical malfunction could be predicted earlier.

Thanks to Deep learning content in MLND, which gave me nice exposure to handle data in array format (the assignment was classifying dog images). For the capstone project when I've been exploring Kaggle challenges , sound classifier challenge attracted me , since I already got exposure to identify and classify image files which are 3D arrays. Moving to sound may give me opportunity to understand handling audio files.

Picking a challenge from Kaggle gives me the luxury of clean dataset , so that I can focus more on solving prediction problem rather spending too much time on data preparation. Since my background is from data warehousing and Big data, I never felt data processing is a problem. I was able to download required dataset and information about labels from the following link.
https://www.kaggle.com/c/freesound-audio-tagging

Though machine learning has been exist for the past 30 years (earlier it has been known as data mining) , it popped up to common usage, due to following reasons.

i) Maturity of internet and continuously expanded network bandwidth enabled collecting data from various ends feasible
ii) Continuous improvement in hardware (Disk,RAM and CPU) enabled to store and process these data.
iii) Especially improvement in CPU, GPU and TPU power enabled to run machine learning algorithms on larger dataset possible . GPU power enabled to build Neural networks upto multiple layers which is called deep learning.

When I was exploring similar project in web found following links, Giants like Amazon Google Apple went to advanced level like able to process human speech in real time. I feel starting from here would at least put me in first step in the journey of voice processing.

https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a
https://www.iotforall.com/tensorflow-sound-classification-machine-learning-applications/
https://www.analyticsvidhya.com/blog/2017/08/audio-voice-processing-deep-learning/

Problem statement

The training dataset contains around 9000 audio files classified into 41 labels . Audio files in training set are in wav format, so finding python packages which can parse this format and convert into array format would help to prepare dataset format which could be fed into keras deep learning algorithms.

Finding necessary preprocessing steps would help to execute these algorithms efficiently. Suitable benchmark method would help to assess the model built. Defining proper metrics would be helpful to measure each learning iteration and tuning the model to improve accuracy.

Developing right CNN architecture would be vital to train the model and derive a classifier.

Metrics

The prediction model built in this project would be evaluated by Mean Average Precision@3 (MAP@3). This is a measurement method suggested in Kaggle challenge.

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

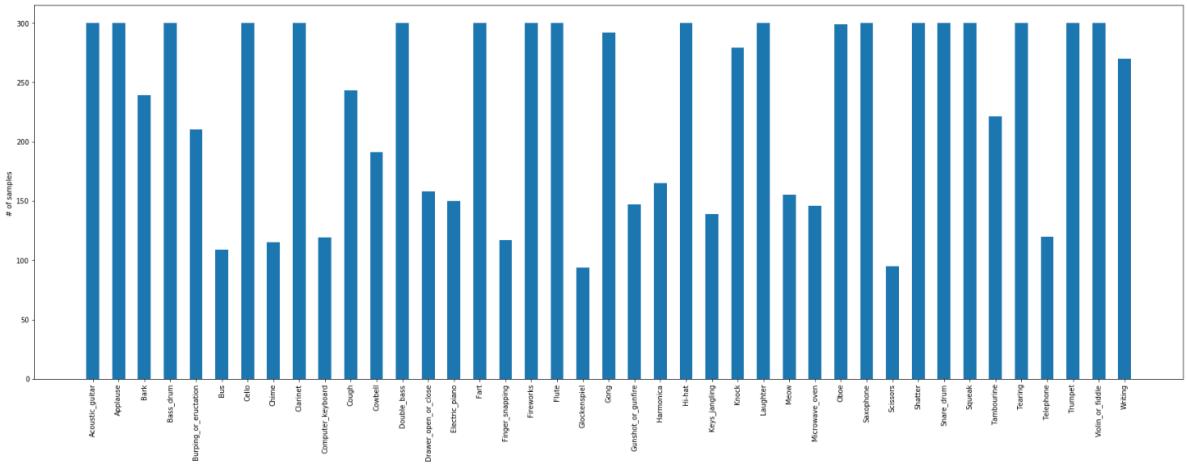
U : the number of scored audio files in the test data,
P(k) : the precision at cutoff k
n : the number predictions per audio file.

I've decided to use this method since it measures accuracy of prediction not only based on exact predicted value , but also considers next 2 predictions in pipeline, i.e , imagine an audio file from validation set, say it is Guitar, and predicted value is Double-bass. Other metrics system would immediately mark as false . But using MAP@3 would look for next two predictions , if the second prediction is Guitar , it would add some positive weightage for the accuracy measure, which would obviously less than if the first prediction is Guitar. For the App in my mind would be benefited if it can measure next two predictions.

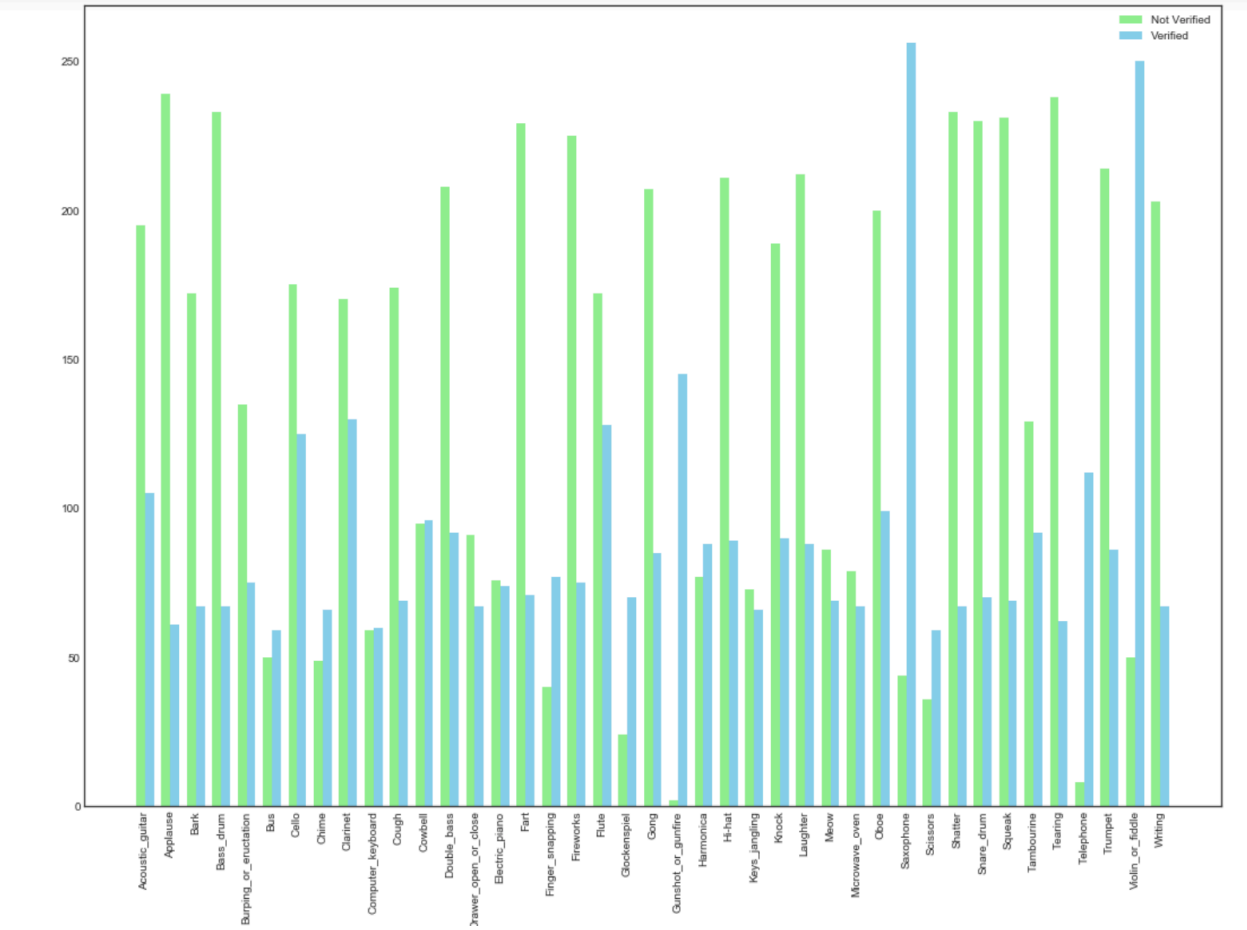
II . Analysis

Data Exploration

Training set used in this problem is made up of 9473 audio files in .wav format and classified into 41 labels. Labels like *Acoustic guitar,Applause,Bass drum,Cello, Clarinet, Double bass*, etc has 300 files each. Labels like *Bus, Scissors , Glockenspiel* has audio files close to 100 audio files.



On top of above following graph shows on each label , number of files which are human verified.



Properties of Single Audio file

To understand a single audio file python packages like scipy was useful, which revealed that following two properties vital to understand sound files.

i) **Frame rate** (frames per second):

Decides quality of sound , higher the frame rate could be clear sound, found that framerate of all audio files in this data is constant 44100, refer following snippet

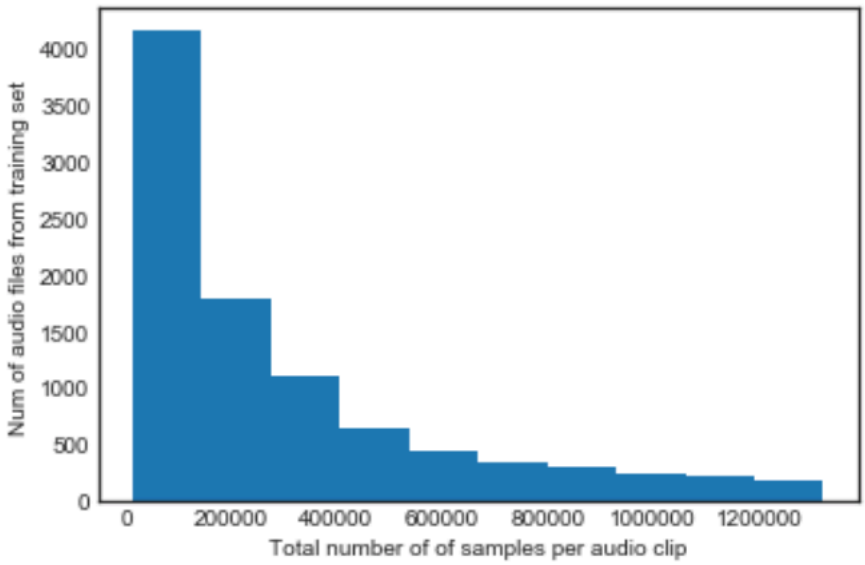
```
sample_rat=[]
sample_len=[]

for i in trainFilesList['fname']:
    sample_rate,samples=wavfile.read(home+'data/audio_train/'+i)
    sample_rat.append(sample_rate)
    sample_len.append(len(samples))

print('Sample rate on all files is ',set(sample_rat))
```

```
Sample rate on all files is  {44100}
```

ii) **Number of samples/frames per audio file** :

Size of the audio file, more samples in a file means the length of audio is longer.

Following histogram provides number of number of audio files with bucket of number of audio samples (length of audio file) in each file.

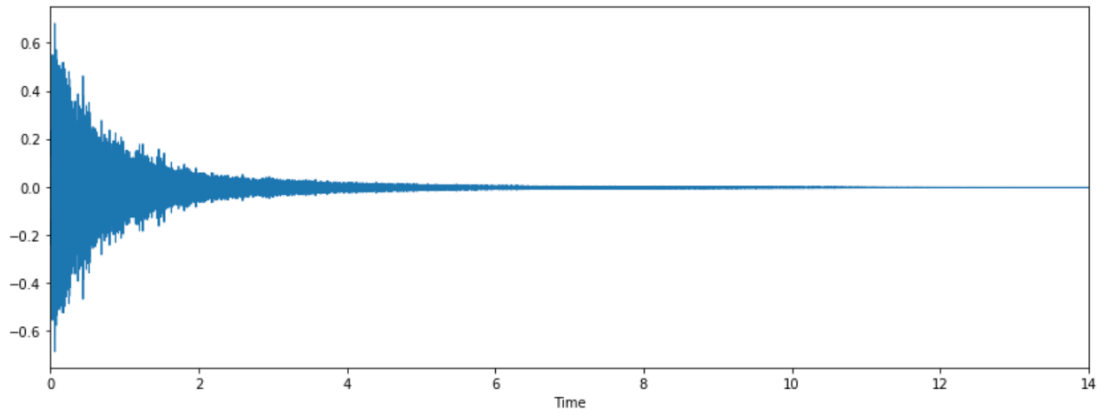


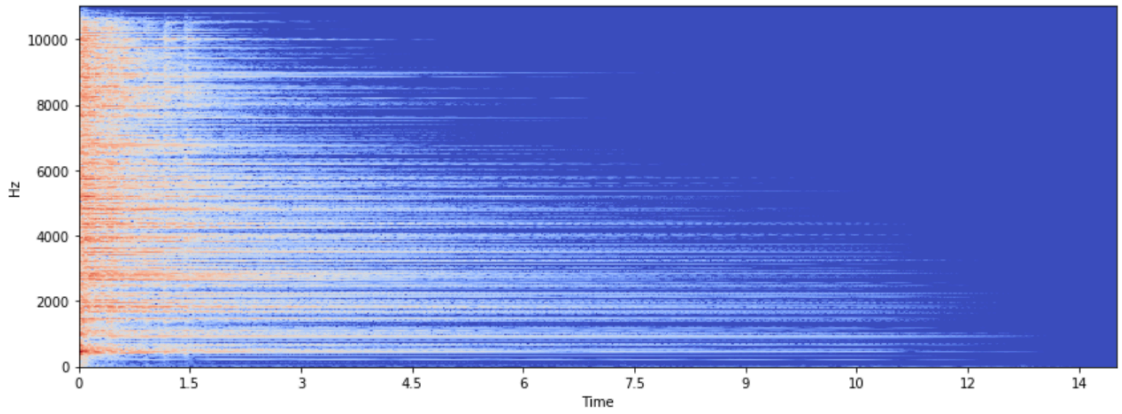
Close to 4k files have 200k audio samples, this is half of the dataset. In remaining half close to 2k files has samples between 200k and 400k . Less than 500 files have samples as high as 1200k.

Exploratory Visualization

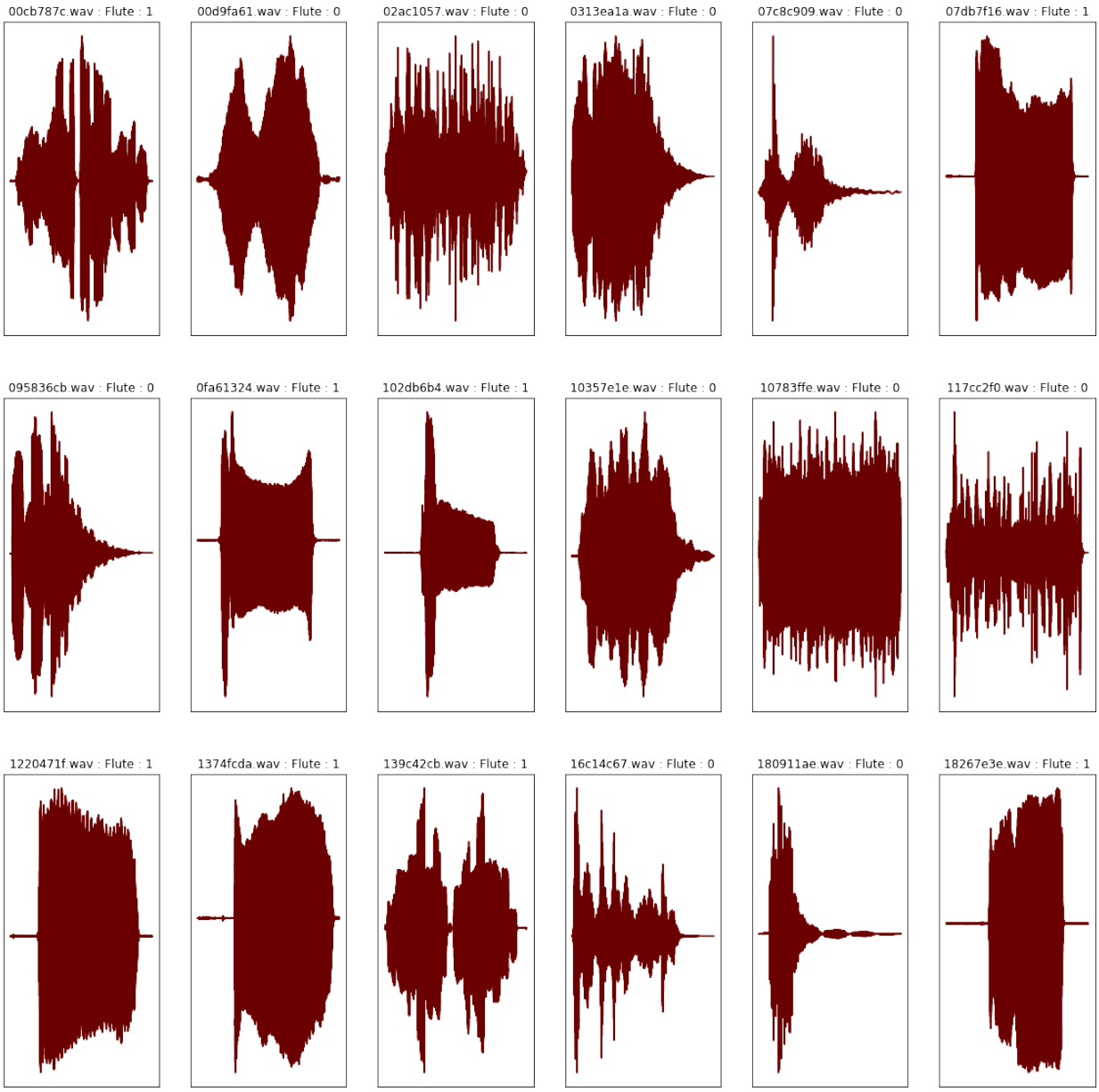
Visualizing individual audio files would be interesting to study possible features which could automatically extracted and used by CNN architecture. Package librosa helped to visualize individual files. Sound features within a second could be understood by following visuals.

First two seconds are intense, After two seconds sound intensity deteriorates gradually and becomes zero. Following two graphs depicts this effect of same audio file.





Followings are smaller size visuals of same kind of sound files (Flue files) , which would help to see the pattern of same kind of audio files.



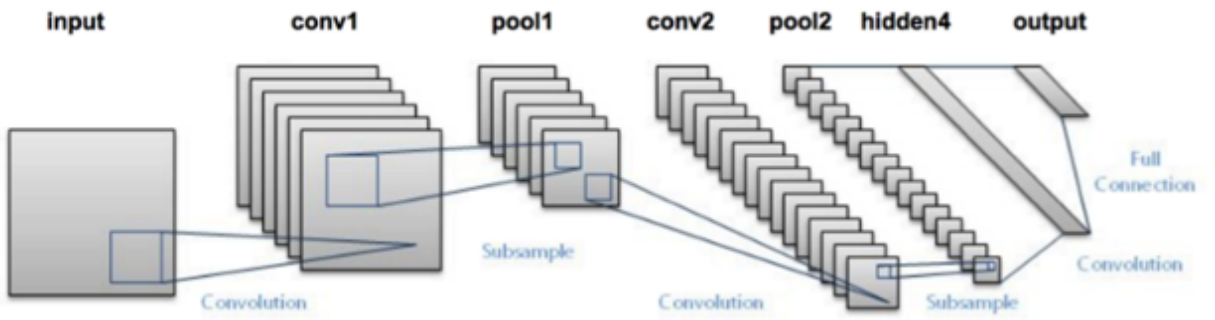
Algorithms and Techniques

Deep learning algorithms are traditional neural networks with multiple layers and huge number of neurons. Recent developments in the field of image classification using these algorithms has proven that these are doing extremely well on datasets like images, sound files, NLP etc. Computing power sought by these algorithms can affordable either by multicore CPU clusters or GPUs.

With my exposure to types of machine learning algorithms and the assignments which I've carried out on MLND , sound files similar to image files except they are single dimension arrays, so CNN could be better fit for this case.

Convolutional Neural Network

CNN is made up of multiple layers, starting with input layer, pooling layer , drop out layer . The final layer would converge into a dense layer made up of all labels. Input layer can be a matrics. Next comes convolution layer, a small window slide over entire matrix and outputs next hidden layer. Convolution layer uses randomly assigned weights to get next layer. Using activation function output from convolution layer would be transformed between 0 to 1. A filter with specific pattern traces for same pattern in all sample arrays. Weights for filter is assigned randomly later updated based reduction in loss. CNN decides what kinds patterns it want to detect based on loss function. Stride decides amount the filter moves for a particular layer. Keras is excellent package with all functionalities to implement CNN.



Source: https://blog.dataiku.com/deep-learning-with-dss

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.

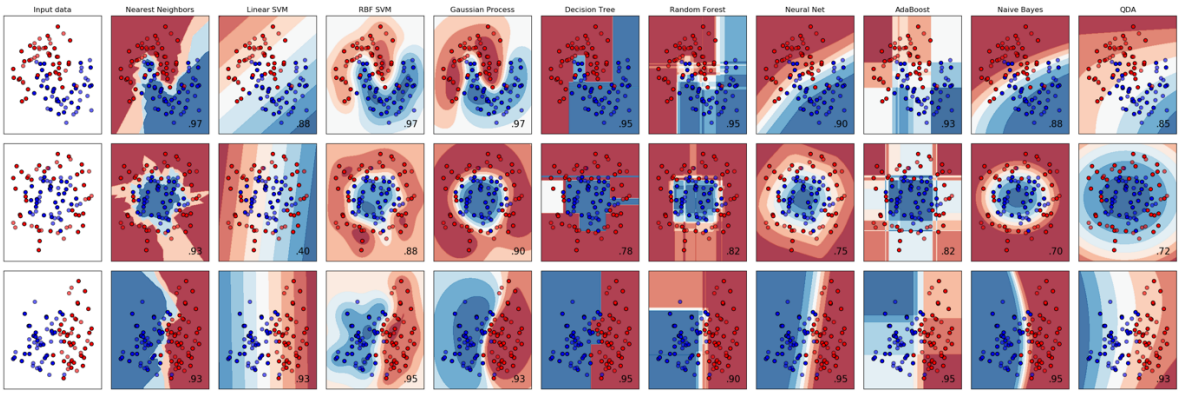
Important Parameters in CNN:

Epochs : An epoch is an iteration over the entire X and y data provided. If suppose epoch=20 , then entire dataset is iterated on training for 20 times. Between epochs weights kept updated based on reduction in loss.
Filter : Number of filter decides how many patterns to detect
Slider size: This decides number of steps to be moved by a filter
Padding : When the slider moves, action to be taken in the edge is decided by this value
Layers : This could be Input layer, Convolution layer, Pooling layer, Output layer
Activation function: smoothens the output between 0 to 1
Loss function : Helps to decide set value of weights for every pattern

From my learnings from MLND, I would consolidate my understanding in other algorithms as following ,

Supervised learning Algorithms :

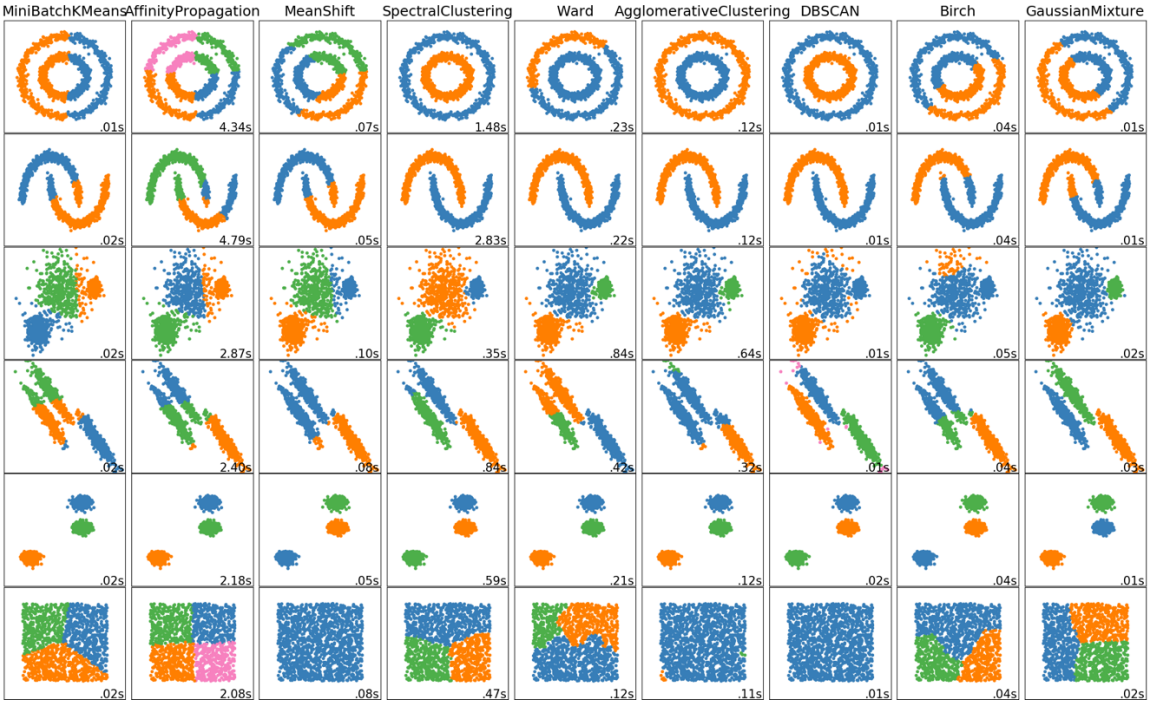
When the dataset is numeric , or can easily encoded to numeric. For example forecast house price in an area based on historic house price and other parameters like school district quality, crime rate , employment opportunity etc. In this data set all textual params could be encoded into numeric value, thus final training dataset could be a numeric array and target also could encoded into numeric array. For these scinarios I would choose algorithms like Linear regression, Decision Tree , Nearest Neighbor etc from sklearn



Courtesy : sci-kit learn

Un Supervised learning Algorithms :

When the dataset is similar to above case but if training set doesn't have predicted labels , I would choose these algorithms. When a new data point fed into this kind of model , instead of predicting a label a new data point would be clustered into a pre-classified category . Some of the examples in this category are K-Means, Affinity propagation, Mean-shift etc.



Courtesy : sci-kit learn

Reinforcement learning Algorithms:

These type of algorithms deal with dataset with result of right or wrong for each actions taken by a possibly autonomous system , for example in the case of a self driving, it makes decision of changing steering angle based on inputs that it receive . For every change made, reward is provided based on outcome, thus the system builds a model based on rewards it receive. Some of algorithms used for this category of learning is Monte-carlo, Tabular Q-learning.

Benchmark model

For the application where I'm going to use this model , a mean accuracy of 70% MAP@3 would be great. As the app I'm planning to build going to collect values from various other sensors, I can handle this 70% accuracy with other values.

III . Methodology

Data PreProcessing

Since dataset is downloaded from Kaggle, it can be assumed that the data is clean, when looking for any outliers in features like framerate, number of samples per audio I haven't seen anything significant to be removed.

However CNN would expect every data point would be same shape array for learning . But in the dataset number of samples per audio file or length of audio is not constant. I think this can happen for image files also, since image files could vary based on format , resolution and size would easily end up differing sized arrays. For this dataset I've decided to standardize size of audio that I would use for training as well. Audio length is parameterized and for files with smaller than the standard length, say 2 seconds, I would pad them with zero then audio files are normalized using following formula,

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Implementation

When I've started building CNN model, set input shape more than 2 seconds of audio, but the data set doesn't have more than 2 seconds across all samples so learning process was failing. Later decided to go with 2 seconds which also accelerated learning process. Since it is a classification problem. I've used categorical_crossentropy loss function. Optimizer I've decided to go from Adam.

Decided to go with epoch 20 and fold 2 arbitrarily, but this took lot of time to complete, at least 3 hours. I have to leave it overnight for a full run to complete. Set checkpoint to save best only , since final best model is enough , but saving all would help if overfitting occurs , but when I predicted files from test set (whose labels unknown) observed accuracy 53% which is close to training accuracy.

Input layer has to be in the shape of input files, for this project being framerate is 16000 and I've taken 2 seconds of audio for training shape of Input layer would be (32000,1).

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	(None, 32000, 1)	0
conv1d_25 (Conv1D)	(None, 31992, 16)	160
conv1d_26 (Conv1D)	(None, 31984, 16)	2320
max_pooling1d_10 (MaxPooling	(None, 1999, 16)	0
dropout_13 (Dropout)	(None, 1999, 16)	0
conv1d_27 (Conv1D)	(None, 1997, 32)	1568
conv1d_28 (Conv1D)	(None, 1995, 32)	3104

```
_________________________________________________________________
max_pooling1d_11 (MaxPooling (None, 498, 32)           0
_________________________________________________________________
dropout_14 (Dropout)         (None, 498, 32)           0
_________________________________________________________________
conv1d_29 (Conv1D)           (None, 496, 32)           3104
_________________________________________________________________
conv1d_30 (Conv1D)           (None, 494, 32)           3104
_________________________________________________________________
max_pooling1d_12 (MaxPooling (None, 123, 32)           0
_________________________________________________________________
dropout_15 (Dropout)         (None, 123, 32)           0
_________________________________________________________________
conv1d_31 (Conv1D)           (None, 121, 256)          24832
_________________________________________________________________
conv1d_32 (Conv1D)           (None, 119, 256)          196864
_________________________________________________________________
global_max_pooling1d_4 (Glob (None, 256)               0
_________________________________________________________________
dropout_16 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_10 (Dense)             (None, 64)                16448
_________________________________________________________________
dense_11 (Dense)             (None, 1028)              66820
_________________________________________________________________
dense_12 (Dense)             (None, 41)                42189
=================================================================
Total params: 360,513
Trainable params: 360,513
Non-trainable params: 0
_________________________________________________________________
_________________________________________________________________
```

Refinement

CNN algorithm uses number of parameters for input, after execution of every epoch, CNN would make a decision either to save model with current learning or not based on validation loss between previous epoch and this one.

A mean accuracy of 70% would be great, but able to reach only 52% through the current process , Increasing number of epochs could increase accuracy.

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 .

Accuracy on Training set	Accuracy on Test set
52%	53%

Since accuracy on test set is closer to accuracy on training set the model generalizes well.

https://www.kaggle.com/c/freesound-audio-tagging/leaderboard

Here is the sample output

Predicted	**Actual**	**Predicted-next-3-values**
Acoustic_guitar	Flute	[Acoustic_guitar Laughter Cello]
Tearing	Tearing	[Tearing Keys_jangling Computer_keyboard]
Fireworks	Writing	[Fireworks Writing Knock]
Cello	Clarinet	[Cello Acoustic_guitar Gong]
Cough	Cough	[Cough Laughter Bark]
Acoustic_guitar	Double_bass	[Acoustic_guitar Cello Double_bass]
Cello	Cello	[Cello Double_bass Acoustic_guitar]
Hi-hat	Tambourine	[Hi-hat Tambourine Shatter]
Harmonica	Harmonica	[Harmonica Violin_or_fiddle Gong]
Cello	Clarinet	[Cello Clarinet Flute]

Justification

After multiple iterations the accuracy is 52%

V . Conclusion

Free-form visualization

Reflection

To recap steps I've followed.

- Prepared dataset using packages like librosa,pyaudio and scipy which are very helpful to process sound files and convert them to single dimension array.
- Prepared plots to using numpy and matplotlib packages,which helped to understand overall dataset. Using the package librosa plotted single audio file.

- Standardized dataset by taking equal length of data sample only, for the smaller sound files padded with 0, and normalized the data set.
- Built CNN architecture using Convolution1D, MaxPool, Dropout and Dense layers.
- Executed learning process by splitting dataset into multiple folds.
- Verified model through few sample predictions .
- Able to derive a model with AP@3 accuracy of 52% .

Improvement

Exploring mechanics of sound would help to tune the learning process and thus improve accuracy

Reference:
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/