**Baby crying Audio classification project:**

**Usage:** Identify and classify baby crying reasons into different categories using CNN .  
You can record the voice and then classify the reason for baby crying

**DataSet** :

[Link : https://drive.google.com/drive/folders/1VD9TBpRibSBc4n4QcpCEBMMK6BptOduf?usp=sharing](https://drive.google.com/drive/folders/1VD9TBpRibSBc4n4QcpCEBMMK6BptOduf?usp=sharing)

We have 4 different categories of data

1.Discomfort - 197

2.Hungry= 217

3.Tired-187

4.Others- 265 (contains all the noises other than baby cry)

Total 866

Dataset is mostly collected from <https://github.com/gveres/donateacry-corpus/tree/master/donateacry_corpus_cleaned_and_updated_data>

And some are collected from <https://freesound.org/search/?q=baby+cry&f=&s=score+desc&advanced=0&g=1>

And some are collected from youtube

We have only selected 3 categories because data for other two category was not enough for model to get train which is burping and belly pain

**Preprocessing of Data :**

To extract the useful features from sound data, we will use *Librosa* library

* *melspectrogram*: Compute a Mel-scaled power spectrogram
* *mfcc*: Mel-frequency cepstral coefficients
* *chorma-stft*: Compute a chromagram from a waveform or power spectrogram
* *spectral\_contrast*: Compute spectral contrast
* *tonnetz*: Computes the tonal centroid features (tonnetz)

**Building a Model :**

Used Convolutional Neural Network (CNN). CNN’s typically make good classifiers and perform particularly well with image classification tasks due to their feature extraction and classification .

We will use a sequential model, starting with a simple model architecture, consisting of four Conv2D convolution layers, with our final output layer being a dense layer. Our output layer will have 4 nodes which matches the number of possible classifications

**Below are some links referred for the project:**

Understanding baby cries

<https://www.youtube.com/watch?v=Oyur-q0gGAs>

<https://www.petitjourney.com.au/understand-the-different-cries-of-your-baby/>

Technical understanding

<https://www.researchgate.net/publication/277935107_Audio_Pattern_Recognition_of_Baby_Crying_Sound_Events>

<https://towardsdatascience.com/deep-learning-for-classifying-audio-of-babies-crying-9a29e057f7ca>

<https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7>

Research

<http://cdn.iiit.ac.in/cdn/ltrc.iiit.ac.in/icon2015/icon2015_proceedings/PDF/39_rp.pdf>

Data sources

<https://github.com/gveres/donateacry-corpus/tree/master/donateacry_corpus_cleaned_and_updated_data>

<https://freesound.org/search/?q=baby+cry&f=&s=score+desc&advanced=0&g=1>

<https://www.audiomicro.com/hungry-baby-crying-sound-effects-837943>

<https://www.soundsnap.com/search/audio/baby+crying/score?page=1>

Similar products

<https://chatterbaby.org/pages/#how>

Media coverage

<https://www.digitaltrends.com/cool-tech/chatterbaby-app-deciphers-baby-crying/>

<https://www.wxyz.com/news/national/new-app-deciphers-your-babies-cry>

<https://www.cbsnews.com/news/chatterbaby-app-could-help-parents-figure-out-why-their-baby-is-crying/>

Figure1 : Signals

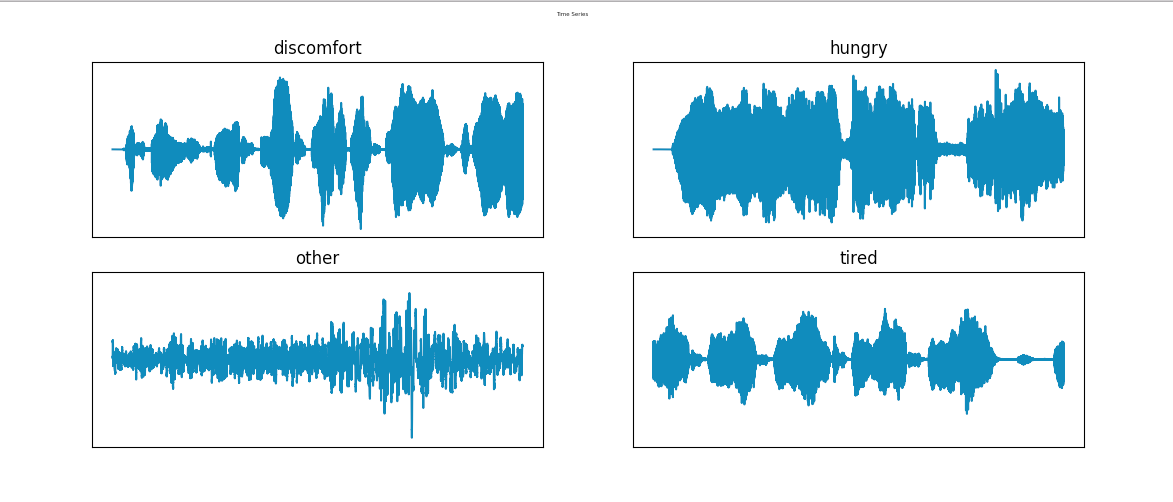


Figure1 : After envelope

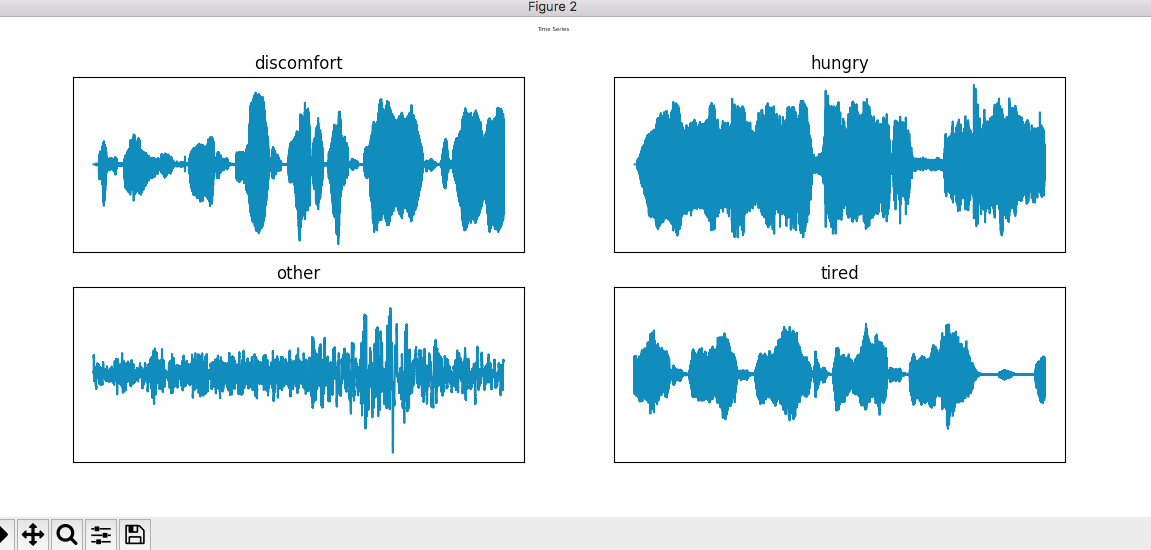


Figure2 : Fourier Transforms

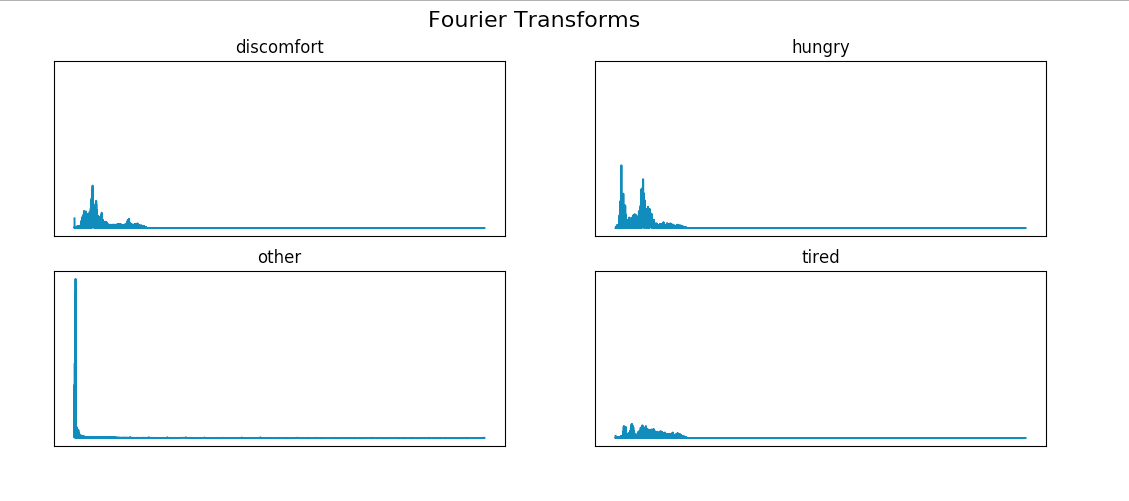


Figure2 : After envelope

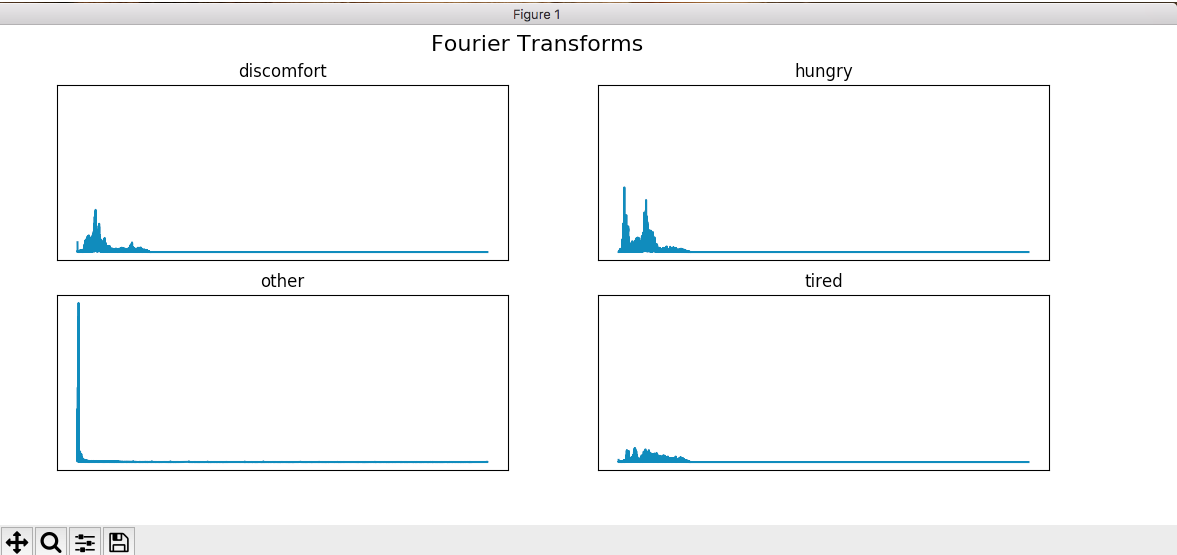


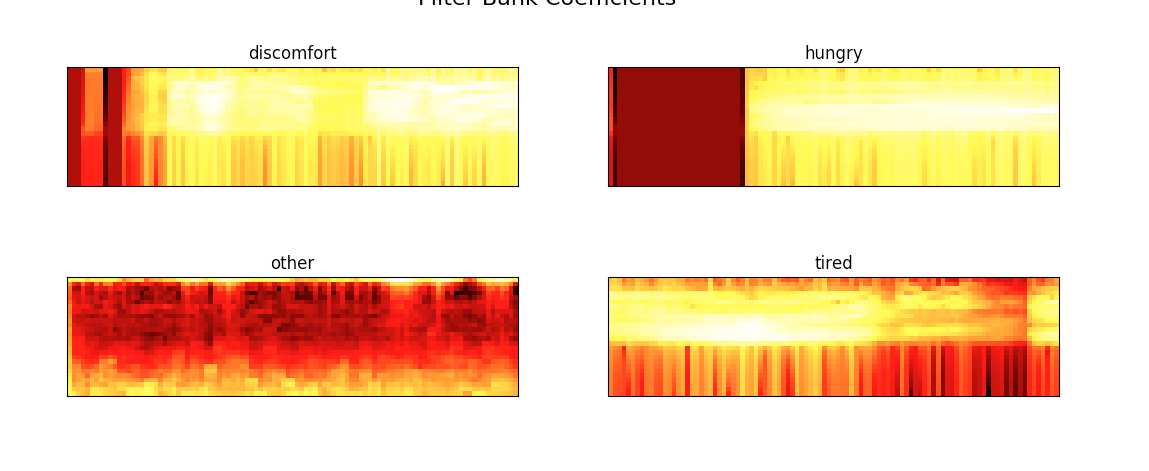
Figure 3 : Filter bank Coefficients

Figure3 : After adding envelope

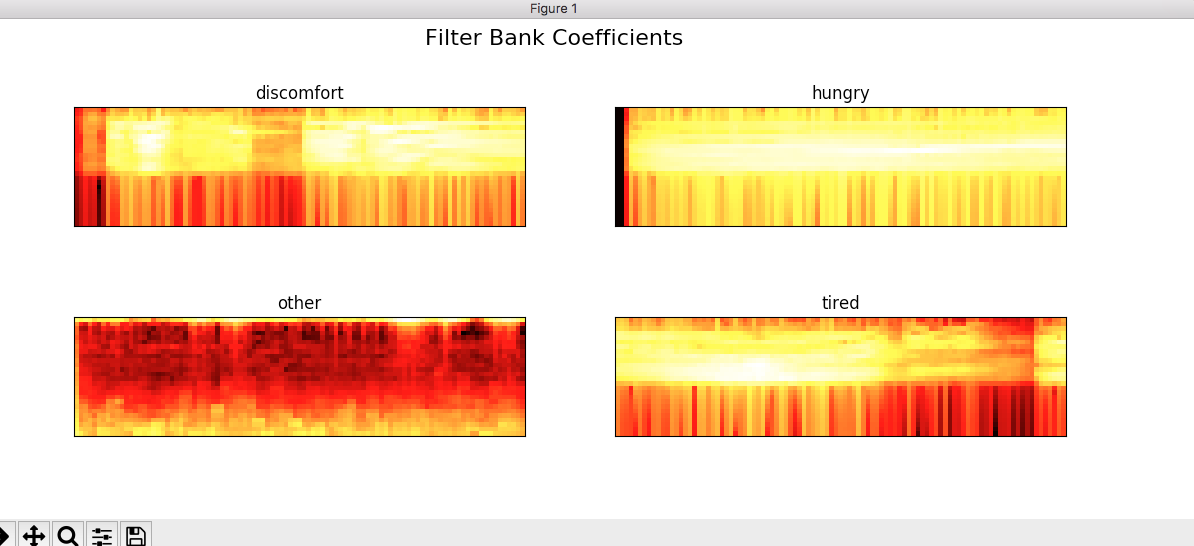


Figure 4 :MFCC

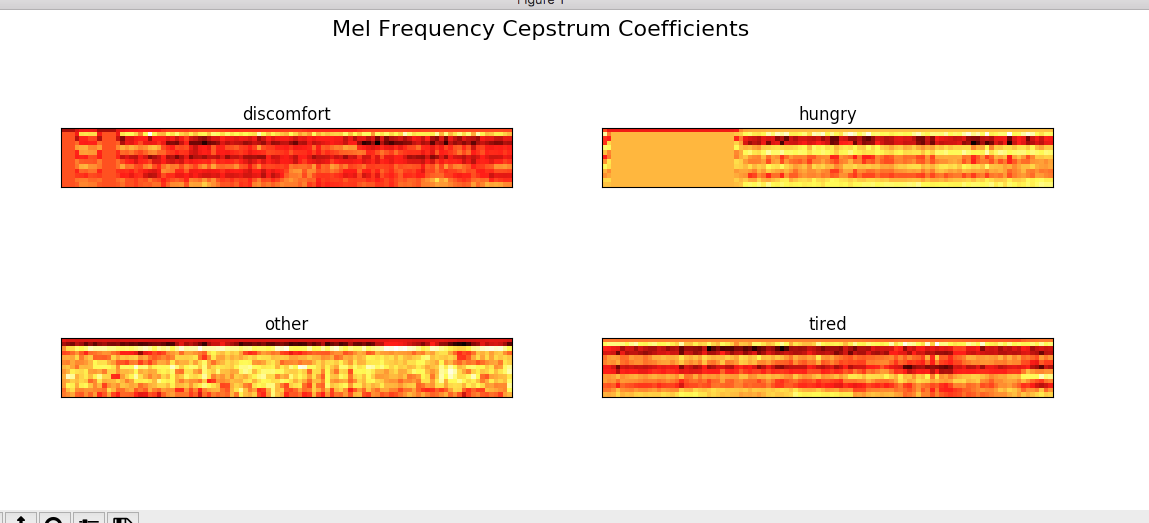
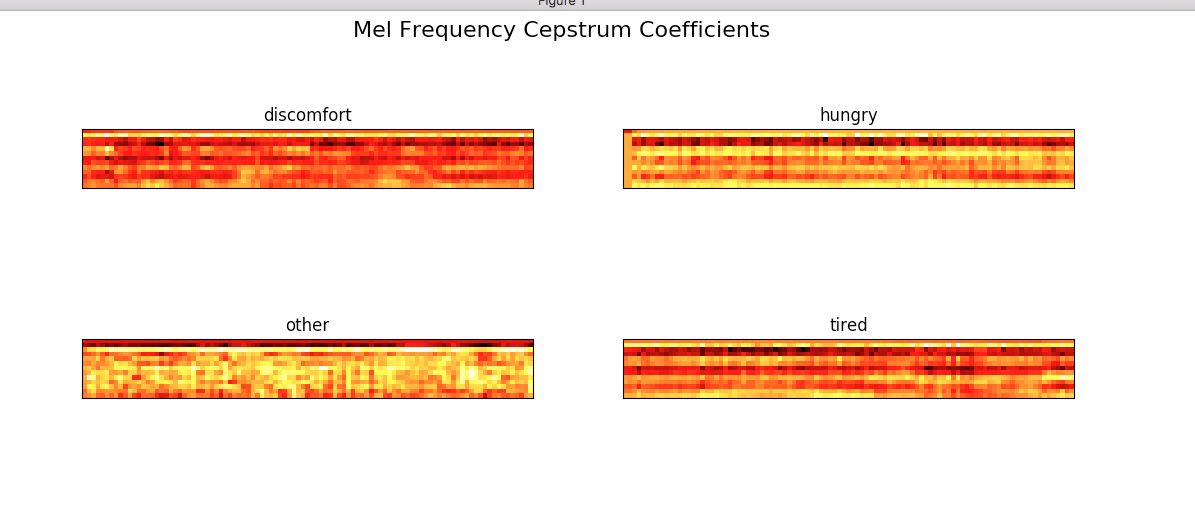


Figure4 : After adding envelope



**FLOW :Train\_model**

1. **Predict (Post Api )**



1. **Predict Recorded Audio**

**Other files used name and functions :**

1. **Remove\_noise.py :**

**def remove\_noise(file\_name):**

-> remove lower and higher frequency data and again store the data

1. **Pyaudio classification   
   1.\_\_init\_\_.py   
    def feature\_extraction(data\_path):  
    - Features of the wav files are extracted and stored in labels and features**

**def train(features, labels, type='cnn', num\_classes=None, print\_summary=False,**

**save\_model=False, lr=0.01, loss\_type=None, epochs=50, optimizer='SGD', verbose=True):**

***epochs: The number of iterations. Default is 50.***

***- lr: Learning rate. Increase to speed up training time, decrease to get more accurate results (if your loss is 'jumping'). Default is 0.01.***

***- optimiser: Default is 'SGD'.***

***- print\_summary: Prints a summary of the model you'll be training. Default is False.***

***- type: Classification type. Default is categorical for >2 classes, and binary otherwise.***

**def predict(model, data\_path):**

**Function for model.predict()**

**def print\_leaderboard(pred, data\_path):**

**Function to print all the classes and its prediction in leader board format**

**2.models.py   
 def cnn(num\_classes):  
 Cnn model we are using is stored in this function   
 - 4 layer cnn network is used**

**- Conv1D for image data**

**- Drop out = 0.1**

**- Activation function = ‘relu’**

**Train & Test result on Data set**

**Distribution — 80/20**

**Iteration 1 : train —> 0.77**

**Test —>0.85**

**Iteration 2 : train—>0.80**

**Test—>0.82**

**Iteration3: train —>0.80**

**Test —>0.82**

**Distribution —70/30**

**Iteration4: train —>0.80**

**Test—>0.85**

**Iteration5: train—>0.80**

**Test—>0.82**

**=======================================================**