Mel-Frequency Ceptral Coeffienents(MFCC) feature extraction for Sound Classification

For sound classification like the Cornell Birdcall Identification (https://www.kaggle.com/c/birdsong-recognition/overview) is usually using the MFCC feature.

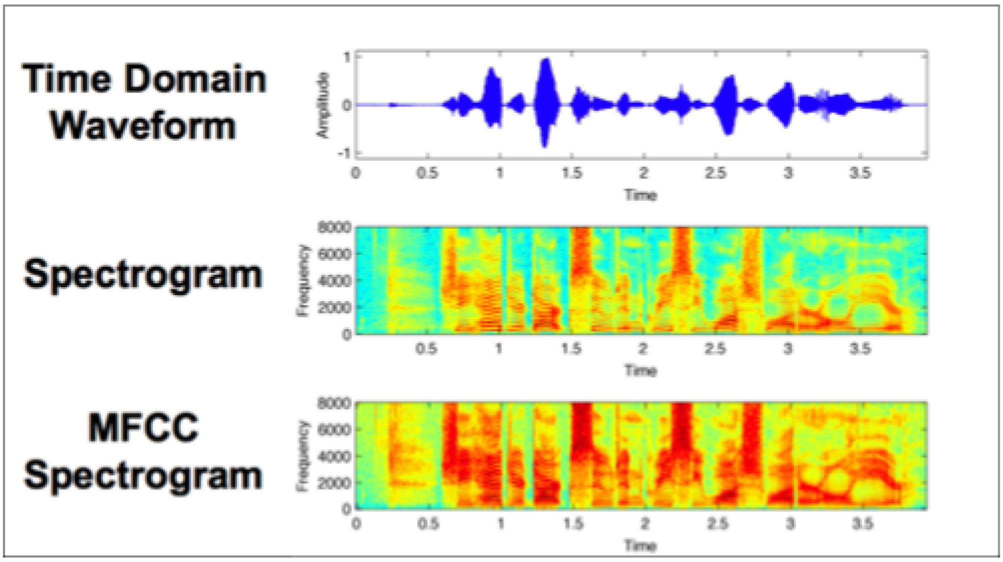
It takes few hours for Cornell Birdcall Identification datasets. I will share extracted feature as dataset after the execution in colab.

In this notebook, I just use 3 mp3 files for each bird class. (check the LIMIT variable)

Please enjoy it and don't forget to vote it.

Feel free to give an advice.

Mel-Frequency Cepstral Coefficients (MFCCs)

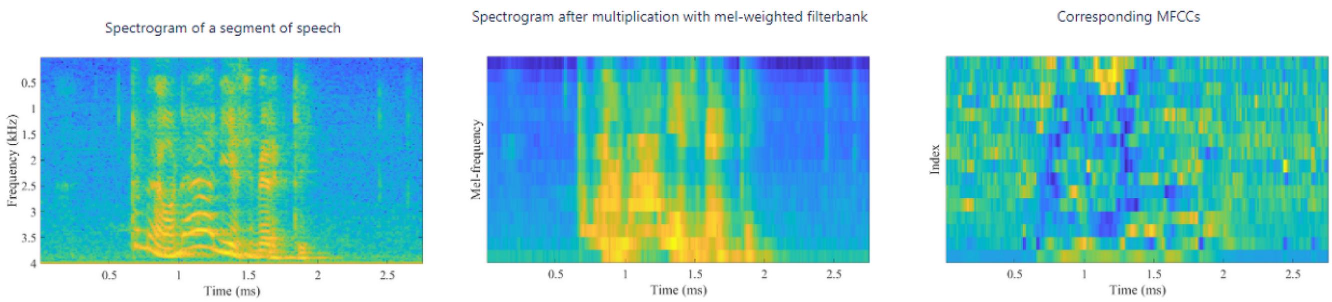


The log-spectrum already takes into account perceptual sensitivity on the magnitude axis, by expressing magnitudes on the logarithmic-axis. The other dimension is then the frequency axis.

There exists a multitude of different criteria with which to quantify accuracy on the frequency scale and there are, correspondingly, a multitude of perceptually motivated frequency scales including the equivalent rectangular bandwidth (ERB) scale, the Bark scale, and the mel-scale. Probably through an abritrary choice mainly due to tradition, in this context we will focus on the mel-scale. This scale describes the perceptual distance between pitches of different frequencies.

Though the argumentation for the MFCCs is not without problems, it has become the most used feature in speech and audio recognition applications. It is used because it works and because it has relatively low complexity and it is straightforward to implement. Simply stated,

if you're unsure which inputs to give to a speech and audio recognition engine, try first the MFCCs.



The beneficial properties of the MFCCs include:

Quantifies the gross-shape of the spectrum (the spectral envelope), which is important in, for example, identification of vowels. At the same time, it removes fine spectral structure (micro-level structure), which is often less important. It thus focuses on that part of the signal which is typically most informative. Straightforward and computationally reasonably efficient calculation. Their performance is well-tested and -understood. Some of the issues with the MFCC include:

The choice of perceptual scale is not well-motivated. Scales such as the ERB or gamma-tone filterbanks might be better suited. However, these alternative filterbanks have not demonstrated consistent benefit, whereby the mel-scale has persisted. MFCCs are not robust to noise. That is, the performance of MFCCs in presence of additive noise, in comparison to other features, has not always been good. The choice of triangular weighting filters wk,h is arbitrary and not based on well-grounded motivations. Alternatives have been presented, but they have not gained popularity, probably due to minor effect on outcome. The MFCCs work well in analysis but for synthesis, they are problematic. Namely, it is difficult to find an inverse transform (from MFCCs to power spectra) which is simultaneously unbiased (=accurate) and congruent with its physical representation (=power spectrum must be positive).

ref: https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC (https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC)

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
import glob
import librosa
import librosa.display
from tqdm import tqdm_notebook as tqdm
from keras.models import Model
from keras.utils import np_utils

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
```

In [2]:

```
LIMIT = 3
```

In [3]:

```
!ls ../input/birdsong-recognition
```

```
example_test_audio                 sample_submission.csv  train_audio
example_test_audio_metadata.csv    test.csv
example_test_audio_summary.csv     train.csv
```

In [4]:

```
df_train = pd.read_csv('../input/birdsong-recognition/train.csv')
df_train
```

Out[4]:

	rating	playback_used	ebird_code	channels	date	pitch	duration	filename
0	3.5	no	aldfly	1 (mono)	2013-05-25	Not specified	25	XC1348
1	4.0	no	aldfly	2 (stereo)	2013-05-27	both	36	XC1354
2	4.0	no	aldfly	2 (stereo)	2013-05-27	both	39	XC1354
3	3.5	no	aldfly	2 (stereo)	2013-05-27	both	33	XC1354
4	4.0	no	aldfly	2 (stereo)	2013-05-27	both	36	XC1354
...	...	...	...	...	...	...	...	...
21370	4.5	no	yetvir	1 (mono)	2019-05-15	both	28	XC4776
21371	3.5	no	yetvir	1 (mono)	2017-05-14	Not specified	52	XC5003
21372	5.0	no	yetvir	1 (mono)	2017-06-10	Not specified	96	XC5012
21373	3.5	no	yetvir	2 (stereo)	2009-05-06	level	35	XC5482
21374	3.5	no	yetvir	2 (stereo)	2010-06-09	level	103	XC5576

21375 rows × 35 columns

In [5]:

```
!ls ../input/birdsong-recognition/train_audio

train_dir = '../input/birdsong-recognition/train_audio'
test_idr = '../input/birdsong-recognition/test_audio'
```

```
aldfly   bktspa   canwre   easkin   hergul   magwar   pinsis   sa
vspa  wesmea
ameavo   blkpho   carwre   easmea   herthr   mallar3  pinwar   sa
ypho  wessan
amebit   blugrb1  casfin   easpho   hoomer   marwre   plsvir   sc
atan  westan
amecro   blujay   caster1  eastow   hoowar   merlin   prawar   sc
oori  wewpew
amegfi   bnhcow   casvir   eawpew   horgre   moublu   purfin   se
mplo  whbnut
amekes   boboli   cedwax   eucdov   horlar   mouchi   pygnut   se
msan  whcspa
amepip   bongul   chispa   eursta   houfin   moudov   rebmer   sh
eowl  whfibi
amered   brdowl   chiswi   evegro   houspa   norcar   rebnut   sh
shaw  whtspa
amerob   brebla   chswar   fiespa   houwre   norfli   rebsap   sn
obun  whtswi
amewig   brespa   chukar   fiscro   indbun   norhar2  rebwoo   sn
ogoo  wilfly
amewoo   brncre   clanut   foxspa   juntit1  normoc   redcro   so
lsan  wilsni1
amtspa   brnthr   cliswa   gadwal   killde   norpar   redhea   so
nspa  wiltur
annhum   brthum   comgol   gcrfin   labwoo   norpin   reevir1  so
ra    winwre3
astfly   brwhaw   comgra   gnttow   larspa   norsho   renpha   sp
osan  wlswar
baisan   btbwar   comloo   gnwtea   lazbun   norwat   reshaw   sp
otow  wooduc
baleag   btnwar   commer   gockin   leabit   nrwswa   rethaw   st
ejay  wooscj2
balori   btywar   comnig   gocspa   leafly   nutwoo   rewbla   sw
ahaw  woothr
banswa   buffle   comrav   goleag   leasan   olsfly   ribgul   sw
aspa  y00475
barswa   buggna   comred   grbher3  lecthr   orcwar   rinduc   sw
athr  yebfly
bawwar   buhvir   comter   grcfly   lesgol   osprey   robgro   tr
eswa  yebsap
belkin1  bulori   comyel   greegr   lesnig   ovenbi1  rocpig   tr
uswa  yehbla
belspa2  bushti   coohaw   greroa   lesyel   palwar   rocwre   tu
ftit  yelwar
```

```
bewwre   buwtea   coshum   greyel   lewwoo   pasfly   rthhum   tu
nswa  yerwar
bkbcuc   buwwar   cowscj1  grhowl   linspa   pecsan   ruckin   ve
ery   yetvir
bkbmag1  cacwre   daejun   grnher   lobcur   perfal   rudduc   ve
sspa
bkbwar   calgul   doccor   grtgra   lobdow   phaino   rufgro   vi
gswa
bkcchi   calqua   dowwoo   grycat   logshr   pibgre   rufhum   wa
rvir
bkchum   camwar   dusfly   gryfly   lotduc   pilwoo   rusbla   we
sblu
bkhgro   cangoo   eargre   haiwoo   louwat   pingro   sagspa1  we
sgre
bkpwar   canwar   easblu   hamfly   macwar   pinjay   sagthr   we
skin
```

Extract Feature using MFCC()

In [6]:

```
def mfcc_extract(filename):
    try:
        y, sr  = librosa.load(filename, sr = 44100)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13, n_fft=int(0.02*sr),ho
p_length=int(0.01*sr))
        return mfcc
    except:
        return
```

In [7]:

```
def parse_audio_files(parent_dir, sub_dirs, limit):
    labels = []
    features = []
    for label, sub_dir in enumerate(tqdm(sub_dirs)):
        i = 0
        for fn in glob.glob(os.path.join(parent_dir,sub_dir,"*.mp3")):
            if i >= limit:
                break
            features.append(mfcc_extract(fn))
            labels.append(label)
            i+=1
    return features, labels
```

In [8]:

```
%%time

train_cat_dirs = glob.glob(train_dir+'/*')
train_cat = []
for cat_dir in train_cat_dirs:
    tmp = cat_dir.split('/')[-1]
    train_cat.append(tmp)
print('the number of kinds:', len(train_cat))

class_num = len(train_cat)
features, labels = parse_audio_files(train_dir, train_cat, LIMIT)
```

```
the number of kinds: 264
```

```
100%                                    264/264 [19:57<00:00, 4.53s/it]
```

```
CPU times: user 19min 34s, sys: 2min 3s, total: 21min 38s
Wall time: 19min 57s
```

In [9]:

```
print(len(features))
print(features[0].shape)
```

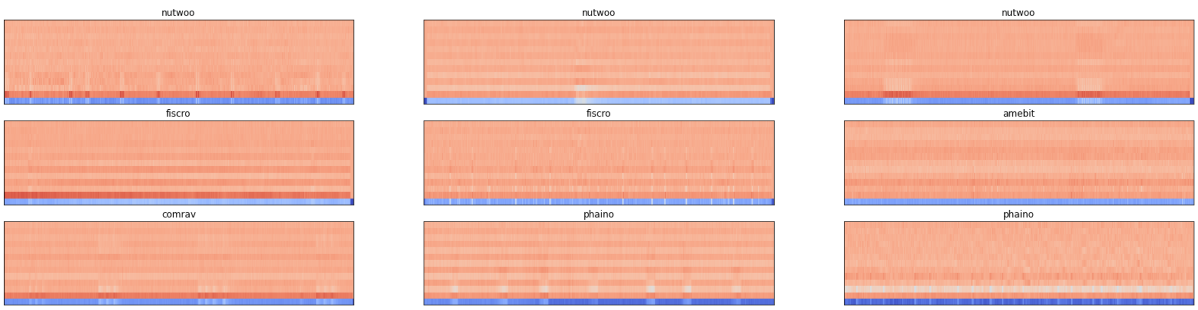
```
792
(13, 8164)
```

In [10]:

```
# plot few features

fig = plt.figure(figsize=(28,24))
for i,mfcc in enumerate(tqdm(features[:100])):
    if i%40 < 3 :
        sub = plt.subplot(10,3,i%40+3*(i/40)+1)
        librosa.display.specshow(mfcc,vmin=-700,vmax=300)
        if ((i%40+3*(i/40)+1)%3==0) :
            plt.colorbar()
        sub.set_title(train_cat[labels[i]])
plt.show()
```

```
100%                                        100/100 [00:00<00:00, 302.42it/s]
```



In [11]:

```
df_submission = pd.read_csv('../input/birdsong-recognition/sample_submission.cs
v')
df_submission.to_csv('submission.csv', index = None)
```