Homework #1: Musical Instrument Recognition

20204310 Haemin Kim

Algorithm Description

〈 feature_extraction.py 〉

1. First, the audio files are loaded and following functions extract features from the raw signals.
2. extract_mfcc
   : Extracts MFCC features from audio and store them in 'mfcc' folder
3. extract_rms
   : Extracts RMS features from audio and store them in 'rms' folder

〈 feature_summary.py 〉

: Includes functions that compute summary of the extracted features

1. MFCC
   a. load_mfcc
      : Load MFCC data stored in 'mfcc' folder
   b. mean_mfcc
      : Apply mean pooling to loaded MFCC data
   c. delta_mfcc
      : Compute delta between MFCC from adjacent frames
   d. double_delta_mfcc
      : Compute delta between MFCC delta from adjacent frames

2. RMS
   a. load_rms
      : Load RMS data stored in 'rms' folder
   b. delta_rms
      : Compute delta between RMS from adjacent frames
   c. double_delta_rms
      : Compute delta between RMS delta from adjacent frames

〈 utils.py 〉

: Uses argparse library to set training options with flags

1. Parser description: Training simple instrument classifier
2. Options:

mfcc_delta	: Whether MFCC delta summary will be used
rms_delta	: Whether RMS delta summary will be used
compress	: Feature compression (PCA or DCT)
dimension	: Resulting dimension after feature compression
classifier	: Classifier (SGD, kNN, SVC)
neighbors	: Number of neighbors when trained with kNN (used only when kNN is selected)

〈 train_test.py 〉

With functions from feature_summary.py, audio features are loaded and summarized to train a classifier. The classifier aims to recognize 10 different instruments which are bass, brass, flute, guitar, keyboard, mallet, organ, reed, string and vocal.

1. MFCC data is loaded based on the 'mfcc_delta' user options.
    - If 'none', raw MFCC data is loaded to train_X and valid_X.
    - If 'd', delta MFCC is loaded to train_X and valid_X.
    - If 'dd', double delta MFCC is loaded to train_X and valid_X.

2. The number of training(1100) and validation data(300) is saved. Then the MFCC feature is reshaped so that the 1D vector describes a single audio file.

3. RMS data is loaded based on the 'rms_delta' user options.
    - If 'none', raw RMS data is loaded to rms_train and rms_valid.
    - If 'd', delta RMS is loaded to rms_train and rms_valid.
    - If 'dd', double delta RMS is loaded to rms_train and rms_valid.

4. The rms_train and rms_valid are concatenated to train_X and valid_X, respectively.

5. Concatenated vectors train_X and valid_X are then compressed to lower dimensions based on the 'compress' user options.

- If ‘pca’, PCA is fit based on the samples in train_X and then transforms train_X and valid_X to lower dimensions. The number of components is given by the user through the ‘dimension’ option.
- If ‘dct’, data samples are compressed with cosine kernels. The number of kernels is given by the user through the ‘dimension’ option.

6. The ground truth label is generated for training and validation.

7. The train_X is normalized with mean and standard deviation of train_X. The valid_X is also normalized with the same statistics.

8. The parameter alphas provide options for hyper parameters in SGD classifiers.

9. Classification model is trained when the train_model function is called. The type of classifier is given by the user with ‘classifier’ option

10. The trained models are saved in the ‘model’ array for accuracy comparison.

11. The saved models are evaluated by predicting instruments of validation audio samples.

12. The ‘wrong’ array keeps track of the number of wrong guesses for each class. After evaluation, wrong guesses are printed out and show the correct answer for that particular data.

Experiments

Based on simple domain knowledge, various experiments are conducted to find the best combination of features and classifiers. Following part of the report illustrates how specific features are chosen, and what parameters showed better results than others.

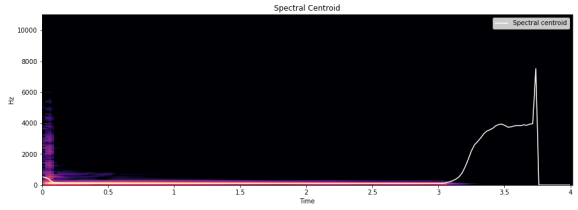
1. Selecting audio features

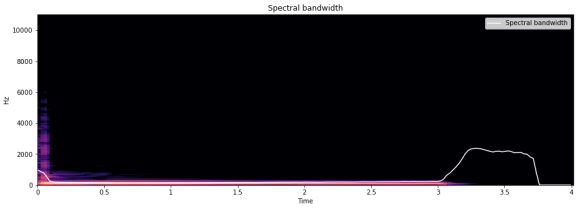
: In order to train the instrument classifier, Mel-Frequency Cepstral Coefficient (MFCC) was first taken into account because it is a widely used feature to capture timbre information. Then another feature was searched for to achieve higher accuracy. Among many possible audio features, features that display similar data when given audio from different classes are excluded in the experiment.

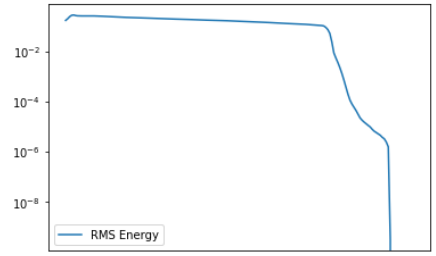
〈 Audio feature visualization 〉

A few features were extracted from audio files and visualized to figure out if they were appropriate for this classification task. Part of the training audio data was selected, and features such as spectral centroid, spectral bandwidth, and Root-Mean-Square energy are visualized and compared below.

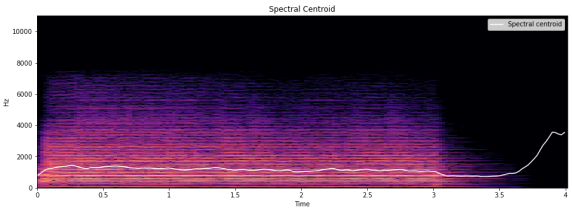
- bass_electronic_000-029-100.wav

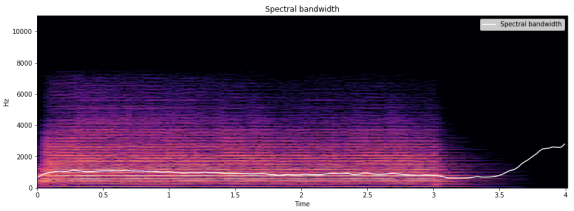


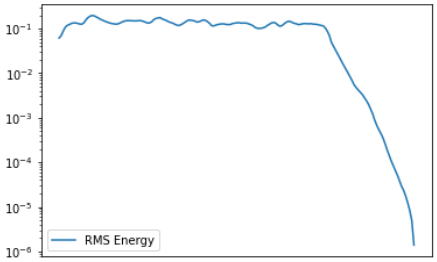




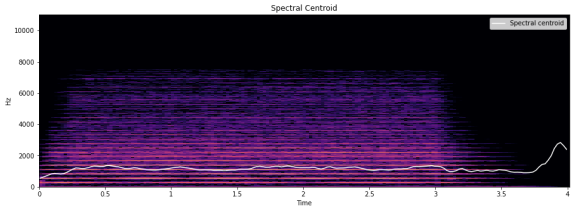
- brass_acoustic_000-043-100.wav

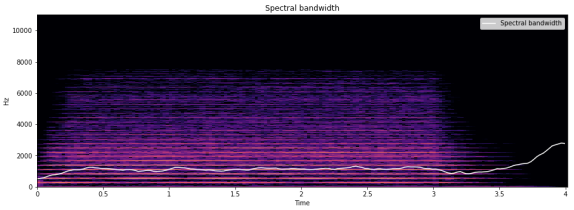


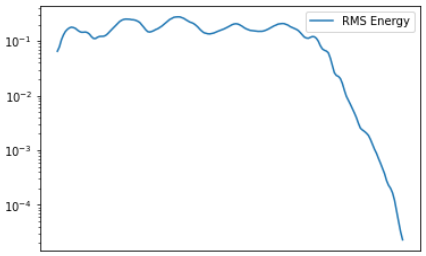




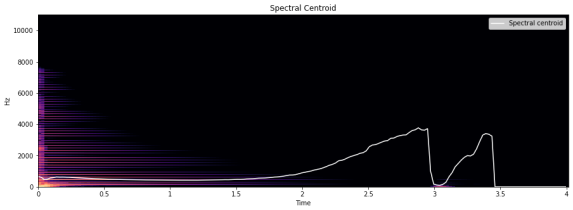
- flute_acoustic_001-061-100.wav

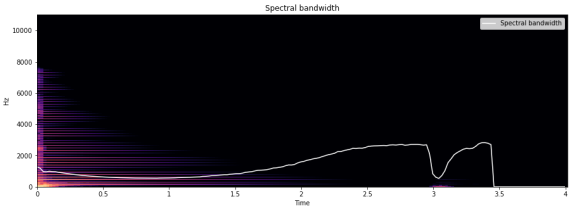


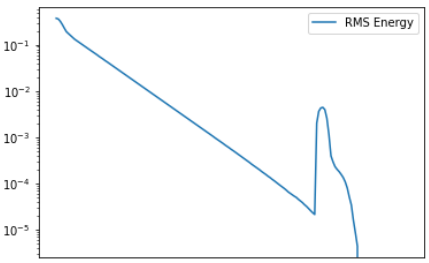




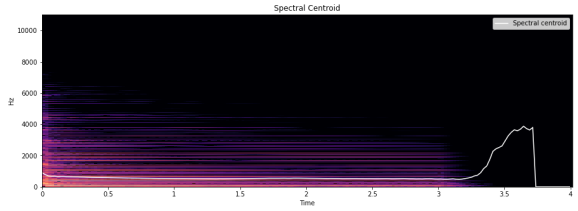
- guitar_acoustic_009-051-100.wav

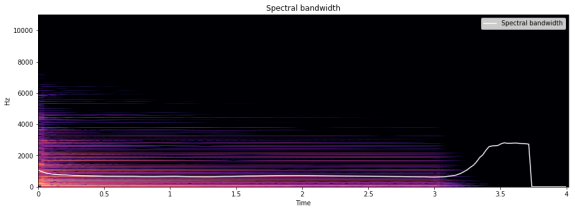


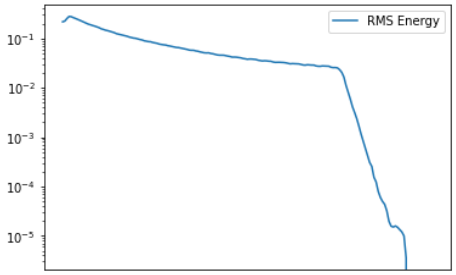




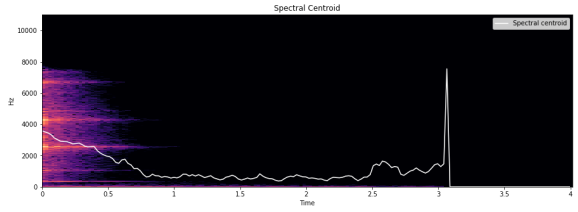
- keyboard_acoustic_000-042-100.wav

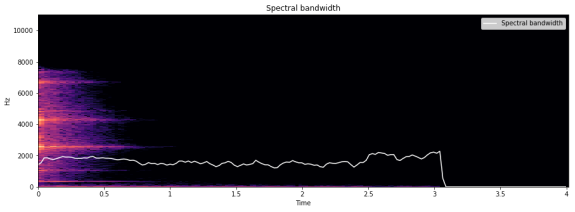


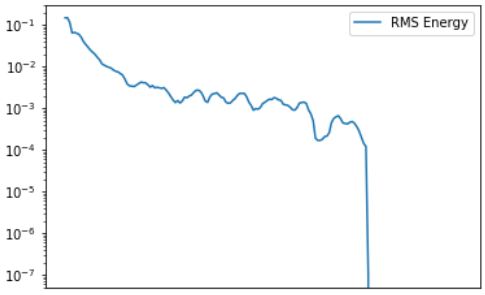




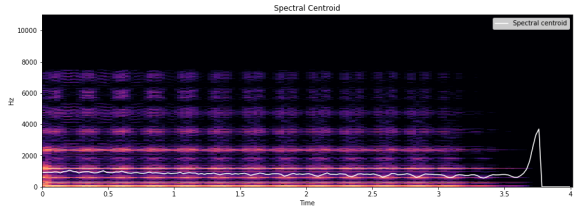
- mallet_acoustic_000-065-100.wav

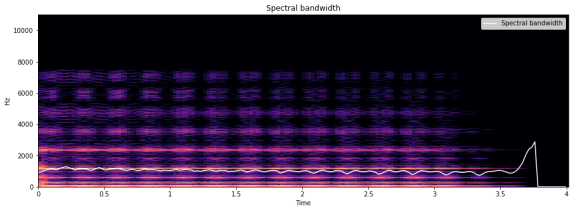


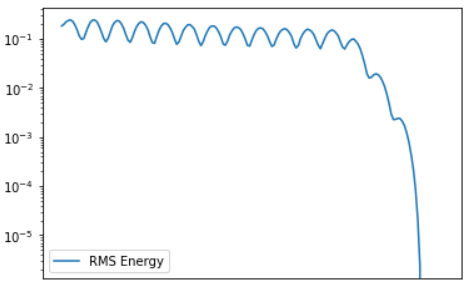




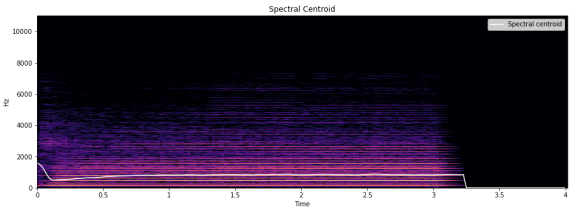
- organ_electronic_000-031-100.wav

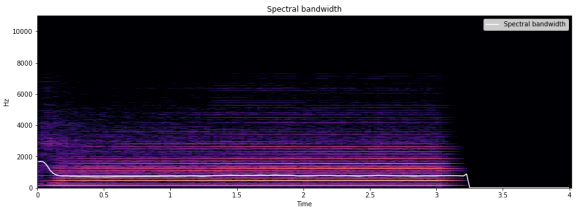


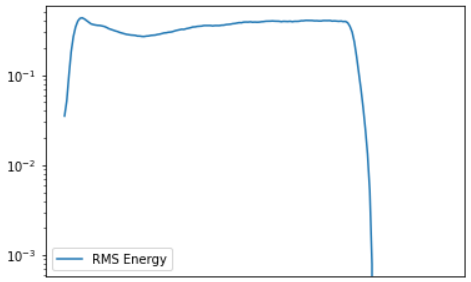




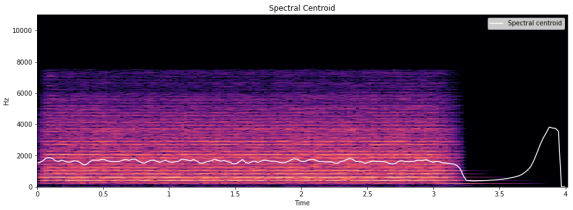
- reed_acoustic_000-051-100.wav

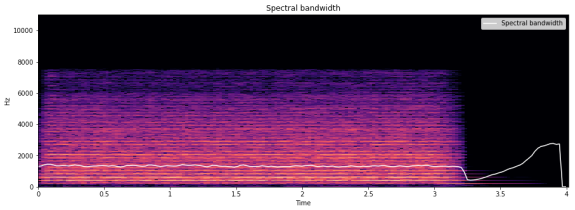


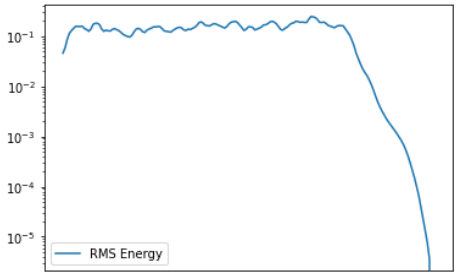




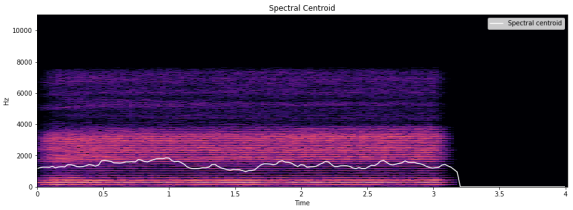
- string_acoustic_006-056-100.wav

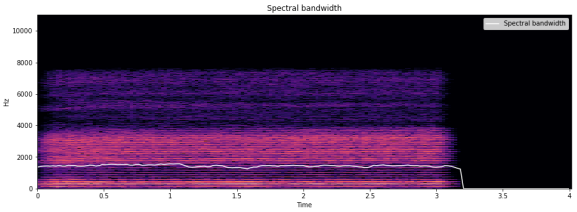


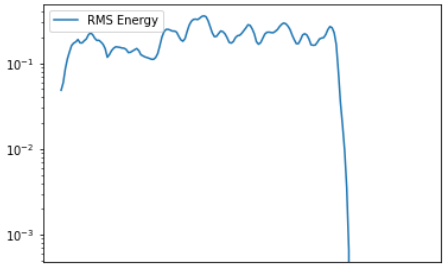




- vocal_acoustic_000-050-100.wav







As seen in the visualizations, spectral centroid shows a similar flat pattern when given different classes of audio files, while RMS energy graph differs based on the instruments. Spectral bandwidth also results in a flat pattern similar to that of spectral centroid. Therefore, RMS was used as an input feature along with MFCC. MFCC with 13 dimensions described timbre information, and RMS represented change in amplitude in the final input vector.

2. Preprocessing selected features

: Given MFCC and RMS, different summary methods are tested to get best performance while decreasing the dimension of the input data. All experiments use MFCC_dim = 13.

a. Delta, Double delta

: All tested with SGD classifier

SGDClassifier(verbose=0, loss="hinge", alpha=hyper_param1, max_iter=1000, penalty="l2", random_state=0)

Delta, Double delta		
Input	parameter	Acc(%)
MFCC, RMS	alpha = 1e-5	97
MFCC_delta, RMS	alpha = 0.001	93
MFCC, RMS_delta	alpha = 1e-5	**95.3**
MFCC_delta, RMS_delta	alpha = 0.0001	93
MFCC_double_delta, RMS	alpha = 0.0005	83
MFCC, RMS_double_delta	alpha = 1e-5, 5e-5, 0.0001	**95.3**
MFCC_double_delta, RMS_double_delta	alpha = 0.0005	83

MFCC_delta, RMS_double_delta	alpha = 0.0005	93.3
MFCC_double_delta, RMS_delta	alpha = 0.001	84.6

The classifier showed lower accuracy when trained with delta/double delta MFCC. Therefore, the delta/double delta RMS feature is used if delta/double delta is tested in the later experiments.

b. Principal Compoenent Analysis (PCA)
: PCA was tested to futher compress the input vector.
All tested with SGD classifier
SGDClassifier(verbose=0, loss="hinge", alpha=hyper_param1, max_iter=1000, penalty="l2", random_state=0)

PCA			
Input	n_components	parameter	Acc(%)
MFCC, RMS	PCA not used (input dimension = 2422)	alpha = 1e-5	97
MFCC, RMS	32	alpha = 0.0005	96.3
MFCC, RMS	64	alpha = 1e-5, 5e-5, 0.0001	**97.6**
MFCC, RMS	128	alpha = 5e-5, 0.0001	**97.6**
MFCC, RMS	16	alpha = 0.001	92.3

Compressing input feature vectors with 64~128 PCA components produced higher performance compared with not using PCA at all. The model finds it easier to learn with summarized information.

c. Discrete Cosine Transform (DCT)
: DCT was tested to futher compress the input vector.
All tested with SGD classifier with different alpha values

SGDClassifier(verbose=0, loss="hinge", alpha=hyper_param1, max_iter=1000, penalty="l2", random_state=0)

DCT			
Input	n_components	parameter	Acc(%)
MFCC, RMS	DCT not used (input dimension = 2422)	alpha = 1e-5	97
MFCC, RMS	32	alpha = 5e-5, 0.001	85.3
MFCC, RMS	64	alpha = 1e-5	91.6
MFCC, RMS	128	alpha = 0.0001	**92.3**
MFCC, RMS	16	alpha = 0.0001	82.6

From the above two sets of experiements, PCA was chosen over DCT because it showed better performance while keeping dimensions of the input vectors smaller. This result could be explained by the inherent attribute of the PCA, which takes into account the overall distribution of the given training data.

3. Selecting classifiers

: In this section, different classifiers such as k-Nearest Neighbor(kNN) and Support Vector Classifier(SVC) are tested.

Classifiers			
Input	Classifier	Parameters	Acc(%)
MFCC, RMS	SGD (default setting)	alpha = 1e-5	97
MFCC, RMS	SVC	default setting	**97.3**
MFCC, RMS	KNN	n_neighbors = 1	**97.3**
MFCC, RMS	KNN	n_neighbors = 5	94.3
MFCC, RMS	KNN	n_neighbors = 11	91.6

SVC classifier is a Support Vector Machine with Radial Basis Funcions non-linear kernel. This non-linear kernel might have allowed the model to classify the data easier compared to linear kernels, resulting in high accuracy among others. The k-Nearest Neighbor classifier with n_neighbors=1 could be chosen as a final classifier by just looking at the numbers. However, the kNN classifier might easily fail when given a different unseen dataset because it only considers one closest data point. Therefore, SVC was chosen for the classifier for this task.

Results

Based on the experiments done with various parameters, the best combination was explored to reach the highest accuracy.

The default setting is used for the SVC classifier.
SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None)

Musical Instrument Recognition				
Input	Compress	Classifier	Parameters	Acc(%)
MFCC, RMS	None	SVC	default setting	97.3
MFCC, RMS	PCA	SVC	n_components = 64	**99.0**
MFCC, RMS	PCA	SVC	n_components = 128	97.0
MFCC, RMS_delta	PCA	SVC	n_components = 64	**99.0**
MFCC, RMS_delta	PCA	SVC	n_components = 128	98.0
MFCC, RMS_delta	PCA	SVC	n_components = 32	97.6
MFCC, RMS_double_delta	PCA	SVC	n_components = 64	**99.0**

99% of accuracy is achieved when MFCC and RMS features are concatenated and compressed with PCA to 64 dimensions, and classified with SVC.

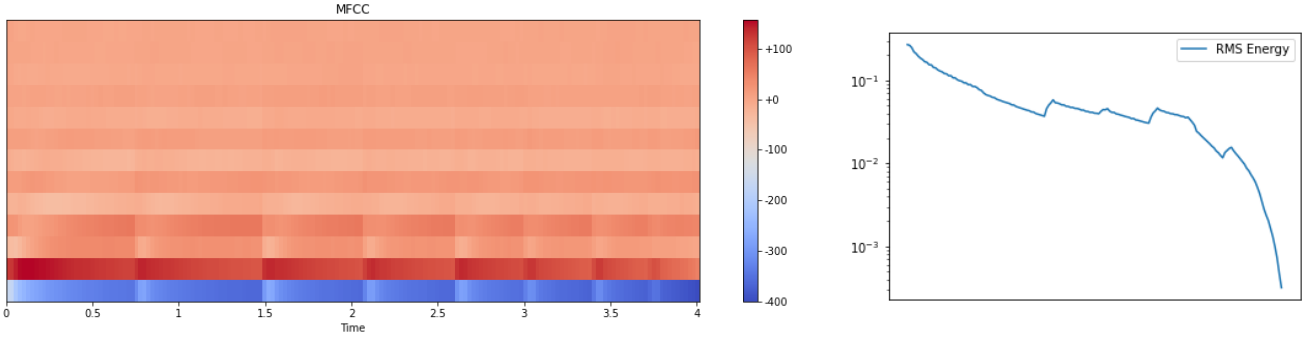
Discussion

To come up with a classification model that recognizes instruments from audio, different parameters were tested in the above section. Through signal visualizations, MFCC and RMS were chosen for input features. Then delta/double delta MFCC and delta/double delta RMS were tested to figure out that using delta information with RMS displays superior results. Data compression methods were tried out to shrink the size of the input data and PCA was selected. Among a few classifiers, SVC reached highest accuracy given the same condition. The best combination was found based on these experiments and achieved 99% of validation accuracy.

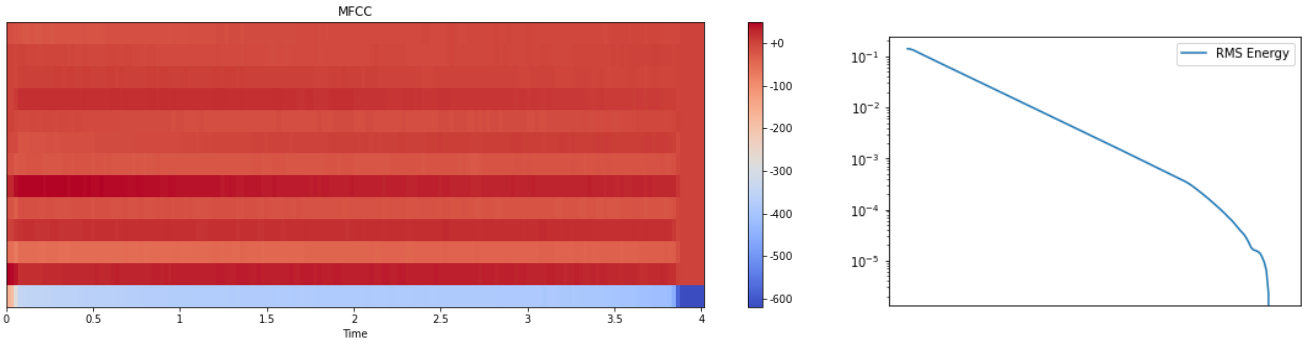
```
train MFCC shape: (1100, 173, 13)
valid MFCC shape: (300, 173, 13)
train RMS shape: (1100, 173)
valid RMS shape: (300, 173)
train_X shape before pca: (1100, 2422)
train_X shape after pca: (1100, 64)
-->  Input feature dimension: (1100, 64)

Final validation accuracy = 99.0 %

Index: 116, Answer: 4, Wrong guess: 7
Index: 179, Answer: 6, Wrong guess: 7
Index: 283, Answer: 10, Wrong guess: 3
Wrong quesses: [0 0 0 1 0 1 0 0 0 1]
```

With the setting that produced 99% of accuracy, the model still confused acoustic guitar with electronic organ, acoustic mallet with electronic organ, and vocal with acoustic flute. Especially, throughout the experiment, the model tends to come up with the electronic organ label when given different audio. It was commonly mistaken for an acoustic guitar sound.

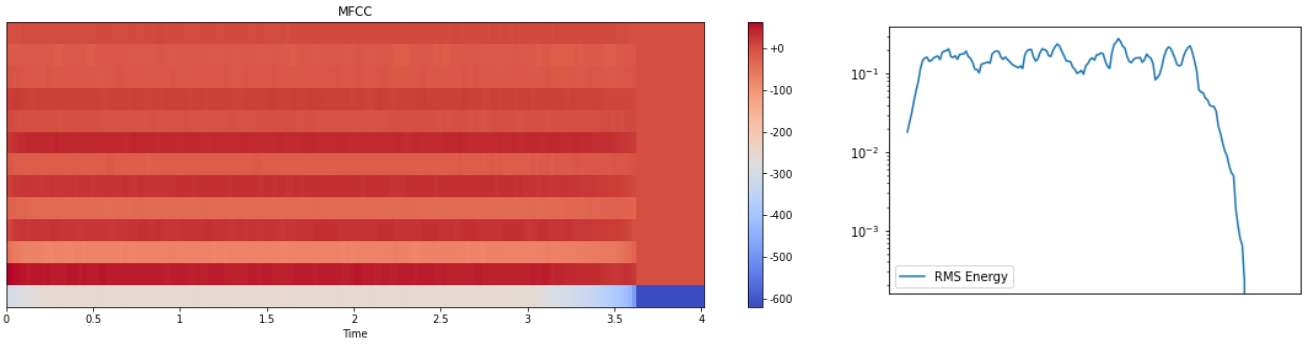
- Index 116, guitar_acoustic_030-045-100.wav



- Index 179, mallet_acoustic_056-091-100.wav



- Index 283, vocal_acoustic_002-082-100.wav



The model would be able to distinguish instruments better if a bigger dataset is given, or model design itself can be improved by using deep neural networks. With deep neural networks, it may learn more implicit attributes of different audio signals to achieve higher accuracy.