MUSICAL INSTRUMENT FAMILY CLASSIFICATION USING SYNTHETIC DATA

Somesh Ganesh, Xiangyu Li, Alexander Lerch
Georgia Institute of Technology, Center for Music Technology
{someshg94, alan.xy.li, alexander.lerch}@gatech.edu

ABSTRACT

This paper proposes a method for musical instrument family classification. Using sparse coding, activations are extracted for audio files and a support vector machine is trained on these activations. Separate datasets for acoustic and synthetic audio are created and the impact synthetic data has on this classification task is studied. The results show that synthetic data could potentially have a positive impact for this task.

1. INTRODUCTION

Musical instrument classification is a recognized topic in the music information retrieval (MIR) community. This widely researched topic involves classifying a musical piece and/or segment into its constituent instruments. This classification could be on the song level or on segment level (of 2 seconds, for example). Musical instrument classification can be considered as a subset of musical instrument family classification, which involves classifying different families of instruments throughout the musical piece/segment. These families could include woodwinds, strings, brass, etc. Music instrument classification disregard the pitch element of the sound and focus only on the timbre aspect which is not very well defined.

A lot of the work done in this topic has been primarily on acoustic instruments [6–9, 11]. With the rise in number and use of virtual studio technology (VST) plugins and synthesizers, a lot of synthetic and electronically generated data can be made available. In this experiment, we have explored the impact of using such synthetic data to classify instrument families using the concept of sparse coding.

The paper is structured as follows. Section 2 goes over the background and related work in the field of musical instrument family classification. Section 3 talks about the algorithm we propose in this paper for classification using sparse coding (introduced later in section 3.1). Section 4 includes the implementation details, our results, observations, discussion and future work. We finally conclude the paper in section 5.

2. RELATED WORK

Historically, most of the approaches are data driven. Some of the researches tries to discover more scientific categories or taxonomies for music instruments by analyzing and clustering audio samples into groups by their perceptual and acoustic characteristics, including Self-Organized Maps with Principal component analysis (PCA) analysis [6], and hierarchical clustering with features selected using Linear Discriminant Algorithm (LDA) on 19 instruments based on Bhattacharyya distances [9]. Those researches produces similar clustering of music instruments which correspond to common knowledge of music instrument taxonomies. In this paper, we use a simplified version introduced by Martin [11] based on human perception and convention. As a result, there can be different stages of the experiment where high level classifications are done first on music instrument families, and then on individual instruments [8] [7].

Feature selection is always an important task for data driven approaches, and traditionally nearly all the features from perception based zero crossing rate, spectural features, Mel-frequency cepstral coefficients (MFCC), raw Cepstral Coefficients and Moving Picture Experts Group Phase 7 (MPEG-7) features with PCA derived features have been used [7] [10] [8]. Raw audio has been used directly in neuron networks without handcrafted features [13]. "Sparse coding is a class of unsupervised methods for learning sets of over-complete bases to represent data efficiently" [2]. In the other word, it tries to make the feature vector larger rather than smaller like what PCA does. The target of sparse coding is to learn a large vector called spare dictionary to map feature space to a mostly close to zero matrix, making the later classification steps easier. Sparse feature learning has also been tried to derive new features for music instrument classification [13].

Different classification models have been tried in related researches. In early years, [11] and [8] selected k-Nearest-Neighbor (kNN) as classification model. Naive Bayesian Classifiers [5], Support Vector Machine (SVM) [4] and recently Convolutional Neuron Network (CNN) [13] have been experimented.

3. PROPOSED METHODOLOGY

A block diagram for our proposed algorithm is shown in figure 1.

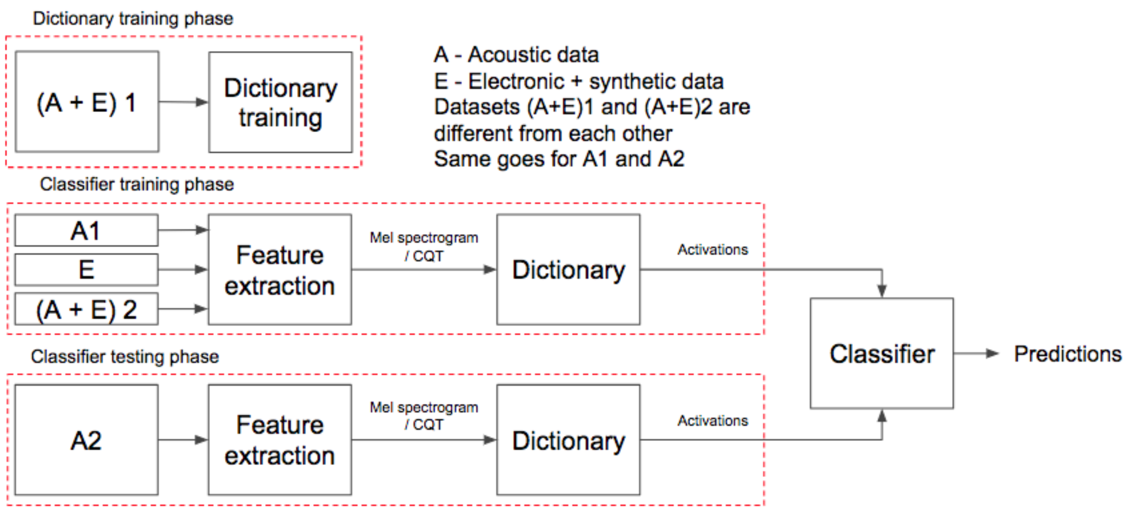

Figure 1. Proposed methodology

3.1 Dictionary learning

Let us briefly introduce the concept of sparse coding here. Sparse coding is an unsupervised algorithm which tries to map an input feature space to an over defined feature space. The output of the algorithm are generally referred to as activations which try to achieve the following properties:

- data fitting, i.e., maximum reconstruction of data
- sparse or latent representation for dictionary

These two terms are weighted according to the desired specifications. The first dotted rectangle in figure 1 represents the dictionary learning process required to extract sparse features. We feed training data into the dictionary which iteratively learns its values over the whole data to achieve the above properties.

3.2 Classification

We now have the trained dictionary to extract sparse features. We now move on to the second dotted rectangle in figure 1. A new training set is multiplied with the dictionary to generate activations. These features are sent to train a classifier (support vector machine, neural network, etc.). Once this is done, a testing set is passed through the dictionary and sent to the classifier, which then predicts the labels. This section is pretty straight forward. The implementation details constitute the chunk of the content which are elaborated in section 4.

4. EVALUATION

4.1 Implementation

4.1.1 Dataset

For this experiment, we use the *NSynth* dataset from Google [1]. This dataset is a collection of audio samples spanning different instruments with a total number of over 300,000 normalized files. Each audio sample is four seconds long with the same attack, decay, sustain, and release (ADSR) characteristics with varying different pitches and strength, which are generated either acoustically, electronically or synthetically. We refer to electronic and synthetic data interchangeably. We have defined our instrument families by grouping the instruments from the *NSynth* dataset in a manner as shown in table 1.

Family name	**Instruments from *NSynth***
String	Bass, guitar & string
Woodwind	Flute, reed & organ
Brass	Brass
Non-sustained	Keyboard & mallet

Table 1. Family distribution from *NSynth*

4.1.2 Data Normalization and Feature Extraction

As shown in figure 1, different and non-overlapping data are used for different phases of the data normalization process. We train two dictionaries, one using constant Q transform (CQT) and another using mel spectrogram. For the dictionary training phase, we randomly select 1000 samples from the NSynth-train dataset of different all three acoustic, electronic, and synthetic sound sources. These 1000 samples are normalized on family level, each of which consists around 250 samples with a few exceptions of missing data, further more source levels balancing the acoustic and non-acoustic samples, and finally instrument level consisting equal number of instruments per family. The data used in classifier training phase are not overlapping with the dictionary training. Different mixes of sound sources from NSynth-train dataset with around 500 each with fore mentioned data normalization are selected to later further examine and compare whether non-acoustic data will influence the result of model. Finally the testing phase uses 1008 acoustic samples from Nsynth-valid dataset which is not overlapping with Nsynth-training dataset with data normalization. This test set contains solely acoustic data since we want to find the impact synthetic data has on this classification task.

Python Library librosa is used for feature extraction [12]. All the files are normalized by their maximum value before the feature extraction. Mel-spectrum, constant Q transform features are extracted for the sparse dictionary training phase, and 20 bin MFCCs with its first and second differences are extracted for baseline approach training and testing on the same dataset.

All our training and testing sets are evenly distributed across the instrument families.

4.1.3 Baseline implementation

For the baseline implementation, we perform a straight forward classification task. We feed the extracted MFCCs (along with their first and second order derivatives) into a support vector machine (SVM).

4.1.4 Dictionary training

We now move on to training the dictionary for sparse coding. The SPAMS library [3] is used for training. The mel spectrogram and constant Q transform (separately) of each of the 1000 audio files are concatenated and sent as input to the dictionary learning block. The dictionary here is of dimension (n x K), where n is the number of K is the number of activations per spectrum. We use K values of 256 and 512. This training set is evenly distributed among the 4 classes and sources. As shown in figure 1, both acoustic and synthetic data are used for this process. Some other parameters varied are the batch size and the lambda value for sparse coding which favors either reconstruction or sparsity, depending on the lambda value. For the final experiments, we use the dictionary with K = 512, batch size = 400 and lambda = 0.15.

4.1.5 Classification Experiment

We now move on to the classification. Here, we use six different configurations for training. We use separate subsets from *NSynth* for acoustic data, synthetic data and mixed data (mel spectrogram & CQT separately). We then test the model against acoustic data.

We use a support vector machine (SVM) classifier for this experiment. We tweak the hyper parameters for a radial basis function kernel and have reported the results below.

4.2 Results

4.2.1 Baseline results

After hyper parameter tuning for the SVM, we get a final mean cross validation accuracy of 70.8% for our baseline approach. We then test this model on an unknown test set of 1008 evenly distributed files. We achieve an accuracy of 57.04% and a confusion matrix as shown in table 2.

4.2.2 Proposed approach results

The results for the different configurations are shown in table 3. We see that the confusion matrix for one configuration (where training set includes both acoustic and synthetic data) is shown in table 4.

Classes	**0**	**1**	**2**	**3**
0	60	59	11	122
1	21	164	63	4
2	89	9	149	5
3	44	5	1	202

Table 2. Confusion matrix for baseline approach

4.3 Discussion

From the results obtained in the previous section, we can clearly make some interesting observations. The results may suggest the following:

- CQT along with sparse coding performs better than mel spectrogram (which performs just better than chance, i.e., 25% in this case) even though the dimensions of CQT are quite compressed with respect to the mel spectrogram (84 vs 128 bins per time frame). This may indicate that either the mel spectrogram is not a suitable feature for this task or that there needs to be more parameter tuning done to the feature extraction. Since we do not get good results from the mel spectrogram, we will not use it any further for our discussion.

- We can clearly see that the model trained solely on acoustic data (using the activations from sparse coding as features) outperforms the model trained on synthetic or the combination of both while testing on the unknown test set (containing only acoustic data).

- Using only synthetic data to train the model performs worse than chance on the test set. However, it outperforms the other models when only the cross-validation accuracy is considered. The model trained using acoustic + synthetic data performs reasonably better than chance too. This can be indicative of synthetic data (being augmented with acoustic data) being useful to classify real world acoustic data.

- Since the baseline implementation performs far better than any of the proposed methods, it is probable that a lot more success can be achieved using parameter tuning for the dictionary training, feature extraction and the classifier. A more complex classifier like a neural network may perform better than the SVM.

5. CONCLUSION

In this paper, we study the impact of synthetic data on musical instrument family classification. We use sparse coding to obtain activations from certain features of the audio files and train an SVM to predict instrument families. From our experiments, we conclude that synthetic data augmented with acoustic data might have a positive impact on this classification task. Some future directions as pointed out in the previous section include a more complex

Configuration of training set	**Feature**	**5-fold cross-validation accuracy**	**Accuracy on unknown test set**
Acoustic	CQT	48.62%	45.23%
Synthetic	CQT	52.68%	20.93%
Acoustic + synthetic	CQT	42.86%	37.5%
Acoustic	Mel spectrogram	30.15%	33.63%
Synthetic	Mel spectrogram	26.79%	29.36%
Acoustic + synthetic	Mel spectrogram	24.41%	32.83%

Table 3. Accuracies for proposed approach

Classes	**0**	**1**	**2**	**3**
0	84	57	36	75
1	33	124	53	42
2	59	75	101	17
3	43	34	104	71

Table 4. Confusion matrix for proposed approach (acoustic + synthetic training set using CQT)

classifier and extensive parameter tuning for the classifier and dictionary training. A better result for this task means that one could possibly have infinite data (since you can generate synthetic data) to train complex and data hungry machine learning models which can obtain a higher classification rate.

6. REFERENCES

[1] The NSynth dataset. retrived 12-04-2017. url: https://magenta.tensorflow.org/datasets/nsynth.

[2] Sparse coding. retrived 12-04-2017. url: http://ufldl.stanford.edu/wiki/index.php/sparse_coding.

[3] Sparse modeling software: an optimization toolbox for solving various sparse estimation problems. retrived 12-04-2017. url: http://spams-devel.gforge.inria.fr/.

[4] Giulio Agostini, Maurizio Longari, and Emanuele Pollastri. Musical instrument timbres classification with spectral features. *EURASIP Journal on Applied Signal Processing*, 2003:5–14, 2003.

[5] Judith C Brown. Computer identification of musical instruments using pattern recognition with cepstral coefficients as features. *The Journal of the Acoustical Society of America*, 105(3):1933–1941, 1999.

[6] Giovanni De Poli and Paolo Prandoni. Sonological models for timbre characterization. *Journal of New Music Research*, 26(2):170–197, 1997.

[7] Jeremiah D Deng, Christian Simmermacher, and Stephen Cranefield. A study on feature analysis for musical instrument classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(2):429–438, 2008.

[8] Antti Eronen and Anssi Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II753–II756. IEEE, 2000.

[9] Slim Essid, Gaël Richard, and Bertrand David. Inferring efficient hierarchical taxonomies for mir tasks: Application to musical instruments. In *ISMIR*, pages 324–328, 2005.

[10] Perfecto Herrera-Boyer, Geoffroy Peeters, and Shlomo Dubnov. Automatic classification of musical instrument sounds. *Journal of New Music Research*, 32(1):3–21, 2003.

[11] Keith D Martin and Youngmoo E Kim. Musical instrument identification: A pattern-recognition approach. *The Journal of the Acoustical Society of America*, 104(3):1768–1768, 1998.

[12] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, Dan Ellis, Fabian-Robert Stoter, Douglas Repetto, Simon Waloschek, CJ Carr, Seth Kranzler, Keunwoo Choi, Petr Viktorin, Joao Felipe Santos, Adrian Holovaty, Waldir Pimenta, and Hojin Lee. librosa, a python library for audio signal processing and music analysis. DOI: 10.5281/zenodo.293021.

[13] Taejin Park and Taejin Lee. Musical instrument sound classification with deep convolutional neural network using feature fusion approach. *arXiv preprint arXiv:1512.07370*, 2015.