Musical Instrument Classification

Deniz Alpay
1650403
deniza@uw.edu

Aidan Johnson
1431797
johnsj96@uw.edu

6 June 2018

EE 443, Spring 2018
Final Capstone Project Report
Design and Application of Digital Signal Processors
Department of Electrical Engineering
University of Washington

Introduction

The intent of this capstone project was to design a system that automatically detects and distinguishes musical instruments in real-time. The ultimate goal was for the system to automatically apply preset filters on music consisting of the classified, solo instrument. We set out to develop a machine learning model based on the frequency-content information pertinent to individual musical instruments. In musical and perceptual terms, this distinguishing factor is called timbre (more on this later). The platform in which this design would be implemented is the Texas Instruments (TI) OMAP-L138 C67488 DSP LCDK (Low Cost Development Kit). This digital signal processor (DSP) development kit, which will hereby referred to as the LCDK, would in real-time classify incoming audio signals (via the Line In port) using the C/C++ language machine learning classification model designed *in silico.*

Music information retrieval (MIR) systems like our proposed design have garnered interest recently in academia and industry, as well as audio information processing in general (e.g., speech recognition) [10]. Applications of a robust musical instrument classifier include consumer and professional audio processing. The interest in this engineering application is best illustrated by commercial-grade and open-source audio processing software packages like Essentia, which has been adopted commercially, and its accompanying machine learning package Gaia[1] [12]. Additionally, in 2016 [4] implemented a real-time system similar to our design proposal, but on the System on Chip (SoC) FPGA and ARM architecture.

Regardless of hardware implementation, intrinsic to any machine learning classification problem is identifying metric or quantifiable attribute to distinguish between data point representations. MIR and musical instrument classification is no exception. No set of definitive fundamental characteristics of audio (speech, musical instruments, etc.) have been established as necessary for timbric feature classification. Outside hardware implementation of acoustic classification, understanding how to define timbre has roots in neuroscience and computational biology. Better understanding of how the brain in humans (and other animals) processes and recognizes timbre definition audio has direct applications to solving the timbre feature classification problem encountered in this project. While we will not be adequately discussing these neuroengineering approaches as it is not pertinent to our project, having this broader scientific perspective is important. The perceptual-nature of timbre and sound classification is the primary driver of why our efforts in this project became more about defining timbre in terms of objective metrics so a classifier could be implemented in hardware rather than the straightforward problem of targeted filtering. To this end, this report will first address the background of instrument classification, then describe our methods for timbric feature extraction and classification, and conclude with a discussion of our results and areas for improvement.

[1] http://essentia.upf.edu/documentation

Background

For humans, distinguishing musical instruments by ear is relatively simple. Our ability to discriminate between phonemes (e.g., vowels and consonants) in speech, sounds in the environment, and instruments in a musical performance. The perceptual and defining attribute that allows us to determine the source of the acoustic information (i.e., sound) we receive on a daily basis is called timbre. Timbre is the discriminating acoustic quality or characteristic of sounds at the same pitch, loudness and time-duration [1]. It is how we can easily tell the difference between the sound of a cello and a saxophone without regard to the sound's pitch quantifiable and objective properties, for example [9]. More generally, it is fundamental in the perception of vocalized animal communication. However important timbre is, it has not been thoroughly studied. This partially due to the difficulty to systematically define quantitatively inherent in the human perception of timbre occuring the neural auditory cortex [1]. It requires identifying the attributes of sound so its source can be inferred. Most strikingly, it is the *perceived* qualities that inform the brain's inference [9]. Although the neural basis of musical timbre and timbre biologically is a highly interesting psychoacoustic and neuroscience topic in itself, this capstone project was not about the human neural networks essential for auditory and timbric processing. Ostensibly, this capstone and this report is about developing a real-time, automatic—read machine learning—classification device on the LCDK provided.

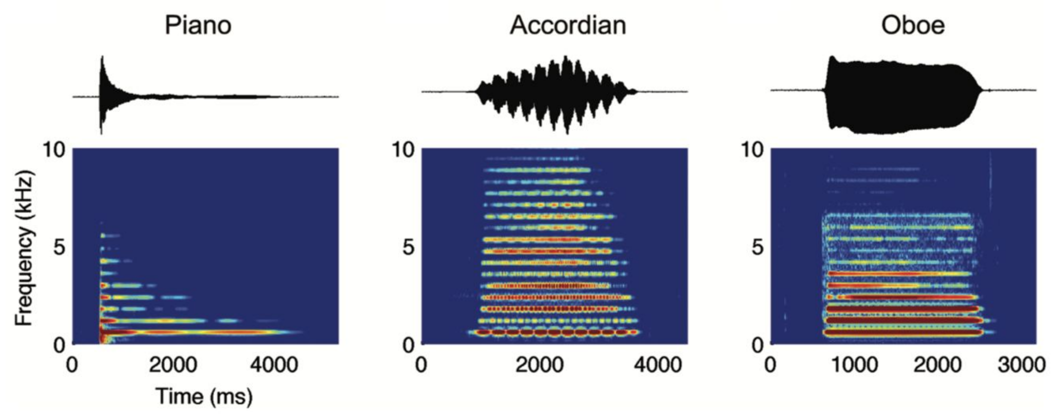


Figure 1: A side-by-side comparison of the frequency-domain or spectral characteristics of three distinct instruments. Notice the differences in the spectrograms (bottom) for the time signal representation of the instrument (top) [1].

Musical instruments can be arranged into five instrument families. Each instrument produces sound (pressure) waves by vibrating air using each family's disturbance modality. Instruments that vibrate air: (1) directly are aerophones; (2) with an oscillating string are chordophones; (3) with its body are idiophones; (4) with a membrane (e.g., reed) are membranophones; and (5) with electrical signals are electrophones [2]. Figure 2 (below) shows each of these families and their respective taxa (i.e., instruments). In music, these instruments are primarily distinguished by ear using the attribute of timbre, as discussed above. To that end, musical instrument classifiers must exploit the distinguishing qualities of the timbre of individual instruments.

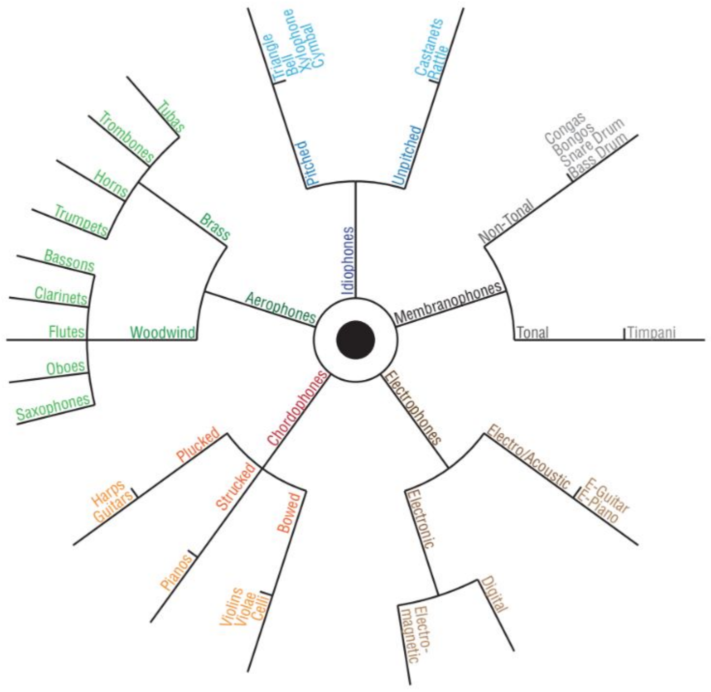


Figure 2: Musical instrument taxonomy of the five instrument families and their taxa instruments [2].

Despite limited research on the underlying mechanisms of timbre, the spectral characteristics of sound and, in this specific case, music have been proposed as one domain to define timbre. Temporal features also play a role in defining perceived timbre. Popular measurements of timbre in literature include the spectral metrics of Mel-Frequency Cepstral Coefficients (MFCCs) and Linear or Perceptual Predicted Coding (LPC). On the temporal side, onset and attack time measurements have been used [9]. The lack of defined dimensionality of timbre means the problem of identifying timbre and corresponding sources (i.e., instruments) is highly multi-dimensional [6].

Methods

Central to the classification problem is choosing a machine learning model. The literature has attempted solving instrument classification using various model types. To account for the complexity introduced by multi-dimensionality, authors in the literature have tried to construct the most parsimonious (e.g., orthogonal) feature space representation of timbre [13]. Others have used Principal Component Analysis (PCA) to reduce the dimensionality [16]. One novel approach to the modeling of timbre in musical instruments has been by developing a model based on the neural spectral-temporal fields produced by neurons in the auditory cortex and simulated 'neurons' of the non-linear classifier, Support Vector Machines (SVM) [9]. This technique detailed in [9] was particularly effective, with a 98.7% reported accuracy at identifying the musical instrument identity. This is reminiscent to the neural inspired artificial neural networks (ANN) albeit to a lesser extent as the foundations of this models are in mathematical and statistical optimization, clustering, and regression, not neurobiology [21].

Apart from the computational biology approach presented in [9], non- and/or loosely-neuro-inspired classification techniques have mostly been used. The most popular and robust models have utilized SVM models in addition to k-Nearest Neighbor (kNN) classifiers [3], [10] —a clustering model that groups data points that are closest typically measured in Euclidean distance and sometimes in an hierarchy (e.g., family taxonomy)—and Gaussian Mixture Models (GMMs) [16], which will be discussed in more depth later. Other classifiers include Hidden Markov Models (HMMs) and ANNs [6], [19]. The effectiveness of ANNs have been particularly promising [19]. SVM models have deemed particularly effective and successful by some authors [2], [5–8], [11]. Considering this consensus of approximately 80–90% accuracy using SVM, we chose to implement an SVM model. Due to hardware constraints, a GMM was instead attempted to be implemented, where similar roadblocks were encountered. Nevertheless, the spectral and temporal features we found to accurately classify the musical instrument audio samples, as well as those used in literature will be discussed at length in the following subsection.

Feature Extraction

Deciding on what features to use was a balancing act between the greater accuracy that additional features provide and the computational cost that is associated with extracting and processing more features. We looked at publications in the area of timbre classification to see what features were most commonly used, but found that each paper used a different combination of features. However, several features were consistently used such as the MFCCs, spectral moments, and zero crossing rate. In an effort to keep the feature extraction and classification faster, we decided on the first thirteen MFCC and the first four spectral moments as the features. These features provide important descriptions of the spectrum in only seventeen data points, compressing the information in the spectrum significantly.

While the MFCCs are not as easily interpreted, the moments of the spectrum give intuitive insights of the shape of the spectrum. There are four measurements to describe the spectral shape. For the i^{th} raw moment:

$$\mu_i = \frac{\sum_{k=1}^{N-1} k^i A(k)}{\sum_{k=1}^{N-1} A(k)}$$

where $A(k)$ is the magnitude of the spectrum, the specific metrics are:

1. The mean or spectral centroid μ_1 gives the frequency which is the center of the energy.

2. The spectral spread or variance, tells us how spread out the spectrum is from the centroid.

$$spread = (\mu_2 - \mu_1^2)^{1/2}$$

3. The skewness describes how asymetric the spectrum is about the centroid.

$$skewness = \frac{2\mu_1^3 - 3\mu_1\mu_2 + \mu_3}{S_v^3}$$

4. The kurtosis or spectral flatness tells us whether the spectrum is concentrated around the centroid or more uniformly spread out.

$$kurtosis = \frac{-3\mu_1^4 + 6\mu_1\mu_2 - 4\mu_1\mu_2 + \mu_4}{S_v^4} - 3$$

Each frame consisted of 512 samples and a filter bank of 48 filters was used in the MFCC computations. The LCDK was set to use a sampling frequency of 48 kHz as it ensured that the 44.1 kHz in the audio files that were played was received. The algorithmic steps are graphically represented in Figure 3. At the end of the process, 13 cepstral coefficients are calculated.

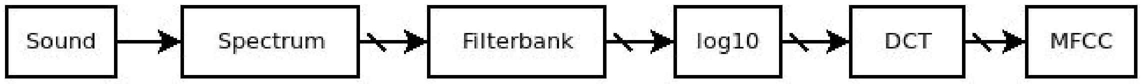


Figure 3: Flowchart for computing MFCCs [13].

To extract these features, we used the Yet Another Audio Feature Extractor (Yaafe)[2] software package from Télécom ParisTech with its Python binding to the C++ API [18]. The extracted features were then used to the train our classification model (see Classification). Issues with the compatibility between the feature extraction on the LCDK and Yaafe's feature extraction led us to implement the feature extraction code in MATLAB (more details are given in the Results section).

Data Source

The dataset used to train a classification model is one of most vital—if not the essential—design aspects of any machine learning problem. Figure 4 (below) illustrates the integral role the training dataset in the entire classification workflow. Before the design of the classification model began, a quality, multi-instrument, representative (at least for Western music) solo performance dataset was found courtesy of Universitat Pompeu Fabra in Barcelona. The original IRMAS dataset[3] consisted of 6,705 training audio files that were 2 seconds of 2,000 recordings, and 2,874 testing audio files that were 5 to 20 second samples from professional

[2] http://yaafe.github.io/Yaafe/index.html
[3] https://www.upf.edu/web/mtg/irmas

recordings (which were not used for training or testing later on) for the classical, jazz and blues, and pop and rock genres. The training data had 388 cello, 505 clarinet, 451 flute, 637 acoustic guitar, 760 electric guitar, 682 organ, 721 piano, 626 saxophone, 577 trumpet, 580 violin, and 778 voice recordings [14]. All were 44.1 kHz stereo WAV format files.

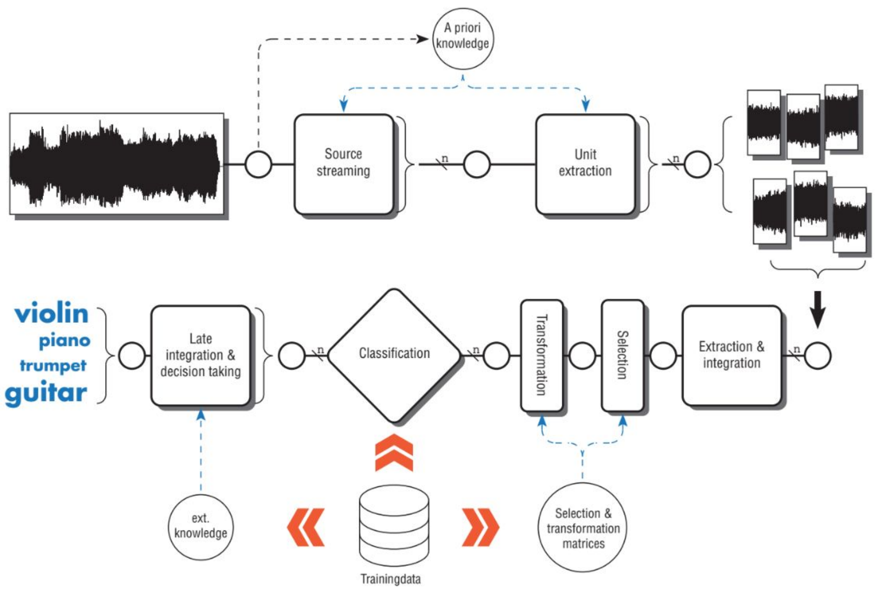


Figure 4: The general workflow block diagram of an instrument classifying system [2].

The training audio either had distinct solo-instrument performances or indeterminable background instruments, and were labeled as such. Our assumption was that any additional background instruments would introduce error into the support vector calculations. To create a more representative dataset, the 'true' solo-instrument performances were divided into 80% training and 20% testing (with voice samples thrown out to simplify the training). It was found during the training process that using all the instruments proved to be non-generalizable such that the SVM model never converged during training. We attribute this to most of the instrument subsets including 'solo' performances where the instrument was weakly dominant in a multi-instrument ensemble. Part trial-and-error, we selected cello, saxophone, and violin to the three instrument classes because they contained the most true 'solo' performances (i.e., where the labeled instrument was actually the predominant source). More than three instruments inhibited convergence.

Classification

SVMs fundamentally transform a non-linear problem into a linear problem by projecting a low-dimensional space to a high-dimensional feature space [19]. In the feature space linear methods can now be used [2]. It is also considered the best option for timbre classification by some authors[8]. An optimal hyperplane is calculated by maximizing the separation, that is the geometric margin, between it and the closest of N data points for k classes [6], [11]. If each data point $x_i = \{x_1, x_2, ..., x_D\}$ has D dimensions, where each has a class y_i that is $+1$ for the positive class and is -1 for the negative class, a hyperplane can separate the k classes if the problem is linearly separable in the feature space. The hyperplane has the equation $w^T x + b = 0$ for its normal vector w, and a perpendicular distance between the plane and the origin $b/\|w\|$. The margins are defined by:

$$w^T x + b \geq 1$$
$$w^T x + b \leq -1$$

where x_i has class $y_i = +1$ for the former bound and where x_i has class $y_i = -1$ for the latter bound. If there are two classes ($k = 2$, i.e. binary classification), then hyperplanes H_1 and H_2 contain these margins (support vectors) and divide the feature space. We find the optimum by solving the minimization problem:

$$\arg\min_{w,b} \tfrac{1}{2} \|w\| \text{ such that } y_i(w^T x + b) - 1 \geq 0 \text{ for } i = 1, \ldots, l$$

where $\|w\| = w^T w$. Figure 1 below visually summarizes the components in the SVM concept. The problem is solved by linear estimation using the equation [11]:

$$\widehat{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i) \text{ for the kernel function } \phi(x_i) \text{ and the Lagrangian multipliers } \alpha_i$$

The kernel function is defined by the relationship:

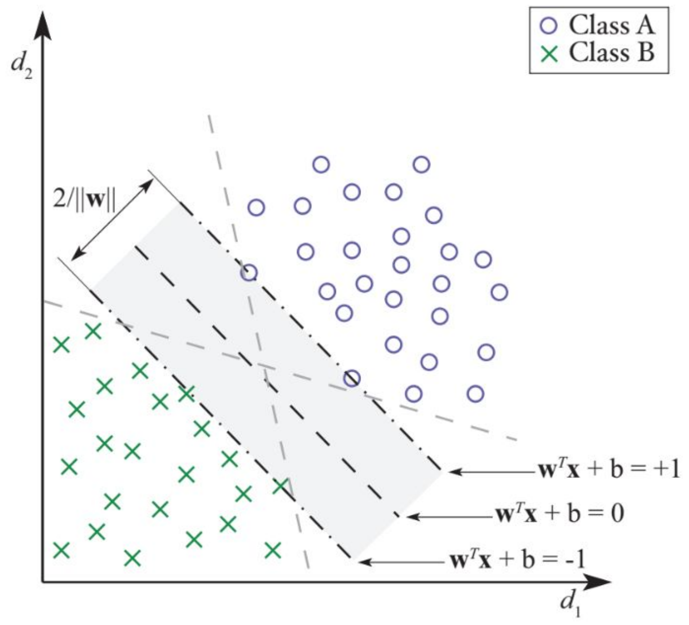
$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

where

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

which is otherwise known as the radial basis. The only catch: the γ parameter is required to be estimated. The parameters defining γ, and therefore the final SVM model, are checked during cross-validation [11]. A disadvantage of the SVM model is the high-level of complexity, which has been suggested to be overcame by fewer kernel evaluations [7]. To train the SVM model, first the desired features were extracted from the training data using Yaafe with the Python interface (see Feature Extraction). Then the features were exported to a local file to be read in by a Python script to be executed by SVMTorch[4], a C++ library [15]. The exported SVM feature model was then cross-validated with the testing data features. As a sanity check, the training data features was also tested.

[4] http://bengio.abracadoudou.com/SVMTorch.html



○ Class A
× Class B
d_2
d_1
$2/||\mathbf{w}||$
$\mathbf{w}^T\mathbf{x} + b = +1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Figure 5: Optimal boundary corresponds to the black dashed line, the hyperplane frame to the black dot-dashed lines, and the hyperplanes that do not maximize distance to the dashed grey lines [2].

Since instrument classification necessitates multiple k-classes, the mathematical definition for binary-classes is insufficient. Multiple class SVMs come in two types: one vs. one and one vs. all [5]. The one vs. one approach creates a binary classifier for each pair of classes. A testing sample is classified to the class with which it has the highest positive classifications. A one vs. all approach makes a classifier for each class to distinguish its class from the rest. A testing sample is classified as being in a certain class if it maximizes the distance from a margin. SVMTorch implements the one vs. all approach, which is the generally accepted higher-performing option [20]. Our results below in Table 1 (see Results and Discussion) suggest that, while giving better performance in terms of accuracy, one vs. all creates a unwieldy number of support vectors. Due to the sheer number of vectors, this one vs. all SVM model bears a high computational cost and memory footprint.

Since SVM has been reported to the highest performance [6], ideally SVM would be implemented in hardware rather than *in silico*. However, its high computational burden prevents it from being implemented on the low-memory LCDK provided for this university course. As has been reported to the instructors, the LCDK has run out of memory multiple times for the lab assignments (it was especially protracted in lab 3). An alternative, GMMs have been reported by the literature to have been used audio classification problems such as instrument classification [11], [16], [17]. Nonetheless, they have been reported to have lower accuracy [7]. The guiding assumption made in GMMs is that a probability distribution (i.e., of a feature) $p(x|\lambda)$ is a weighted sum of Gaussian density function $b_m(x)$. In mathematical terms, for x random vector with D dimensions and M mixtures,

$$p(x|\lambda) = \sum_{m=1}^{M} c_m b_m(x) \text{ where } b_m(x) = \frac{1}{2\pi^{D/2}|\Sigma^{1/2}|} \exp\left(-\frac{(x-\mu)^T \Sigma^{-1}(x-\mu)}{2}\right)$$

for weight c_m, mean μ_m, and covariance matrix Σ_m. The GMM can even be used in higher dimensions, as our problem requires [11]. For the GMM, instead of Python, MATLAB was used to extract the features from the training data and then train. The same features in the SVM-Python implementation were extracted for again. (All code for the SVM and GMM model training, and DSP hardware implementation of the GMM can be found in the .zip file.)

Results

After the dataset culling and rebuild (as discussed in the Data Source subsection), the dataset consisted of the audio files summarized in Table 1 below. The dataset was greatly reduced in size after the removal of non- and pseudo-solo instrument performances. However, the culling greatly improved the performance, which is illustrative of the importance of the dataset. After trial-and-error, it was determined that the largest and best performing (highest separation) instrument subset of flute, violin, cello, saxophone, and clarinet. Subsets greater than these five instruments was detrimentally affected by poor separation, and consequentially accuracy nosedived below ~60%.

The highest-performing feature set for these instruments was determined, after yet more trial-and-error, to be 15 features extractable using Yaafe. Sparing the reader of needless definitions, the feature set extracted were (with literature references that also used them to characterize the spectrum, which is directly related to timbric features):

1. MFCCs [19].
2. Spectral shape statistics (centroid, spread, skew, and kurtosis) [7], [19].
3. Amplitude modulation (AM) to characterize tremolo at 4–8 Hz and grain at 10–40 Hz [3], [7].
4. Onset (start of a musical note) by analyzing the spectral energy flux [3].
5. OBSIR (Octave Band Signal Intensity Ratio) between consecutive octaves [7].
6. LPCs (Linear Predictor Coefficient) by autocorrelation methods, are time-domain features that also describe the envelope in the frequency-domain [2], [19].
7. Spectral crest factor per log-spaced band of quarter octave [3].
8. Spectral decrease [7].
9. Spectral flatness, or ratio of geometric and arithmetic mean, per log-spaced band of quarter octave [7], [19].
10. Spectral slope by linear regression of the spectral amplitude [3], [19].
11. Spectral variance by (normalized) correlation of consecutive frames.
12. Spectral irregularity or the difference of consecutive Constant-Q frequency transform

bins [7], [19].

13. ZCR (zero-crossing rate) or count of sign inversions [6], [7], [19].

The best performance of a larger set of classification labels, using the above features, is summarized in Table 1. The labels it considered were flute, violin, cello, saxophone, and clarinet. Note the reduction in dataset size. A dataset with uniformly 'true' solos would likely be beneficial; increasing the size of the training and testing data is a reasonable rule-of-thumb. The cross-validation with the testing data for the trained model is summarized in Table 2.

Table 1: Rebuilt dataset summary for the best subset of labels (classes).

Instrument	Training Data	Testing Data
Cello	77	40
Saxophone	82	35
Clarinet	82	37
Flute	27	8
Violin	120	41

Table 2: Summary of five class SVM cross-validation; multiclass accuracy of 65.63%.

	Cello	Saxophone	Clarinet	Flute	Violin
Accuracy	84.06%	87.44%	85.02%	95.307%	82.33%
Type I Error (False Positives)	4.14%	4.28%	2.28%	4.141%	9.66%
Type II Error (False Negatives)	11.8%	8.28%	12.7%	0.552%	8.01%
Number of Support Vectors	1,324	1,634	1,446	1,051	1,796

When considering what features to use on the LCDK we needed to consider time required to compute the features on each frame. Given that our design requires that the LCDK is always listening and classifying any audio that passes the silence detection check, a quick feature extraction process is valuable. As a result, we narrowed the number of features from the best performing ones in an effort to decrease the computation time while maintaining classification accuracy. We used the MFCC and spectral shape statistics as they improved the classification accuracy the most.

The results of the cross-validation are shown in Table 3. The accuracy of each instrument class is quite different. In the training and testing data split, the violin had more audio in the classical music in the training data and some more audio in the jazz and blues, and pop and rock genres in the testing. There were fewer files with the genres outside of classical music for the remaining instruments that were also solo performances, so both their training and testing contained more classical music. This is reflected in the violin classification error mostly being due to type II error (false negatives), as the classifier encounters more music of different styles that it does not identify as being violin music.

Table 3: Summary of three class SVM cross-validation.

	Cello	**Saxophone**	**Violin**
Accuracy	76.2%	82.2%	63.7%
Type I Error (False Positives)	11.2%	17.8%	3.36%
Type II Error (False Negatives)	12.6%	0%	33.0%
Number of Support Vectors	25,469	20,799	31,416

The support vectors are used to define the decision boundary between the classes. In the simpler case were the data is separable and the decision boundary is not complex, few support vectors are needed to define the decision boundary. In our case, the number of support vectors is very large and which suggests that the decision boundary is more complex and harder to define. In particular, the instrument with the poorest classification rate, the violin, has the most support vectors as its decision boundary may be more ill-defined than the other instruments.

Due to the large number of support vectors, each of which contain 17 floating point numbers, the file size of the model parameters was 11.2 MB. Knowing that the LCDK's memory is restricted and insufficient for the SVM model we looked to using the GMM as an alternative. The GMM appeared as more feasible to implement on the LCDK due to the fact that it is a parametric model. Unlike the SVM that decides how many support vectors are necessary to define the decision boundary, the GMM has a fixed number of parameter given by the dimensionality. This is very advantageous since the we had already decided on a low dimensionality for our features. The GMMs parameters are also interpretable, as the parameters give the means and covariances between the features.

When implementing the GMM on the LCDK we encountered many issues pertaining to the feature extraction process. In particular, we found that the features we were training the model on in MATLAB were unlike the features extracted on the LCDK. Testing the feature computations in MATLAB and the LCDK found that the two were incompatible and would produce different results on the same frame input. To resolve this issue, we wrote our own spectral moment code and used the MFCC code provided for Lab 3 to generate the training features. The C code used to generated the features was tested against the MATLAB feature extraction code to ensure that the two were identical. There was some noticeable numerical error between the two, which was corrected by changing the MATLAB setting to use single precision (32 bit floating point) as the LCDK does. Despite this, the GMM still failed to correctly classify audio and some features.

Discussion

In this section we will briefly discuss the potential improvements that could be made to our design. Despite only using 17 features, our design is fairly slow to process each frame. In each iteration, the LCDK collects 512 samples that make a frame, computes the average of the frame (which is used to eliminate the DC bias and silence detection), computes the FFT, MFCC, and spectral moments, and finally classifies the features by computing the membership probabilities in the GMM. Due to the time it takes to complete this process, the frames are processed infrequently. In addition, since the sampling frequency is 48 kHz, each frame is quite short and doesn't contain so much information. The system could potentially be improved by pooling features over many frames so that the LCDK can consider many frames, and decide on the instrument based on which class is identified the most in the frames. This could resolve the issue of the current design only receiving very short and infrequent portions of the audio stream. In this way, the dimensionality of the classification task is not increased but the classifier receives more information.

To address the high-dimensionality previously mentioned, PCA has been integrated into the machine learning algorithms. In SVM, PCA reduces the dimensions in the feature space. It can remove noise and extract the most essential feature vectors in the model [16]. PCA accomplishes this also without a lost in discrimination information. Data are projected to their abstract, higher dimension feature space such that fewest the chosen projections best describe the maximum variance of the data, while not reducing the model's discrimination. This is achieved by selecting the uncorrelated and orthogonal feature vectors out of the set of partially unrelated features [13], [19]. A principal component analyzed timbre feature space would have a reduced number of support vectors thus reducing the computational cost of a robust classification model such as the preferable SVM.

Conclusion

Implementing timbre classification involved many challenges that lead to design choices involving trades-offs that manage hardware resources effectively. Since timbre classification involves classifying not a particular sound but all sounds that an instrument can create, the feature space is more complex and requires a sophisticated classifier and sufficient features.

Keeping in mind the resource constraints that became apparent in lab 3, we approached the task by first finding the best performing features using the SVM. While the SVM was accurate and more compact, the number of features required a long frame processing time. Scaling back the number of features lead to a reasonably performing SVM, but it was compensated with a large number of support vectors. We attempted to resolve these issues by using a small number of features (only 17) with a simpler classifier model (the GMM), to reduce the resource demands of the classification. However, the reduction in complexity lead to an unsuccessful classifier. We suggested some potential improvements such as PCA and feature pooling, but the issue remains that our finalized model was too simplified for the task we had set. We found that a successful classifier needs a sufficient number of features that compress the spectrum with minimal losses and classifier that can find the complex decision boundary within those features. This task was instructive in demonstrating the trade-offs required in translating challenging tasks solved in software onto resource-constrained hardware platforms.

References

[1] S. M. Town and J. K. Bizley, "Neural and behavioral investigations into timbre perception," *Front. Syst. Neurosci.*, vol. 7, 2013 [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnsys.2013.00088/full. [Accessed: 05-Jun-2018]

[2] F. Fuhrmann, "Automatic musical instrument recognition from polyphonic music audio signals," Universitat Pompeu Fabra, 2012. [Online]. Available: http://mtg.upf.edu/node/2494. [Accessed: 05-Jun-2018]

[3] A. Eronen, "Comparison of features for musical instrument recognition," in *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*, 2001, pp. 19–22.

[4] A. I. Lita, L. M. Ionescu, A. G. Mazare, G. Serban, and I. Lita, "Real time system for instrumental sound extraction and recognition," in *2016 39th International Spring Seminar on Electronics Technology (ISSE)*, 2016, pp. 456–461.

[5] J. Marques and P. Moreno, "A Study of Musical Instrument Classification Using Gaussian Mixture Models and Support Vector Machines," *Cambridge Research Library*, Sep. 1999.

[6] G. Agostini, M. Longari, and E. Pollastri, "Musical Instrument Timbres Classification with Spectral Features," *EURASIP J. Adv. Signal Process.*, vol. 2003, no. 1, p. 943279, Dec. 2003.

[7] C. Joder, S. Essid, and G. Richard, "Temporal Integration for Audio Classification With Application to Musical Instrument Classification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 1, pp. 174–186, Jan. 2009.

[8] E. Guven and A. M. Ozbayoglu, "Note and Timbre Classification by Local Features of Spectrogram," *Procedia Computer Science*, vol. 12, no. C, pp. 182–187, 2012.

[9] K. Patil, D. Pressnitzer, S. Shamma, and M. Elhilali, "Music in Our Ears: The Biological Bases of Musical Timbre Perception," *PLOS Computational Biology*, vol. 8, no. 11, p. e1002759, Nov. 2012.

[10] T. Giannakopoulos, “pyAudioAnalysis: An Open-Source Python Library for Audio Signal Analysis,” *PLOS ONE*, vol. 10, no. 12, p. e0144610, Dec. 2015.

[11] D. Ciminieri, “Musical genre classification and tracking based on clustering driven high level features,” Laurea Magistrale / Specialistica, Politecnico di Milano, 2011 [Online]. Available: https://www.politesi.polimi.it/handle/10589/12881. [Accessed: 05-Jun-2018]

[12] D. Bogdanov *et al.*, “ESSENTIA: an Audio Analysis Library for Music Information Retrieval,” in *International Society for Music Information Retrieval Conference (ISMIR’13)*, Curitiba, Brazil, 2013, pp. 493–498 [Online]. Available: http://hdl.handle.net/10230/32252 [Accessed: 05-Jun-2018]

[13] H. Terasawa, M. Slaney, and J. Berger, “Perceptual distance in timbre space,” in *in Proc. Int. Conf. Auditory Display, 2005*, pp. 61–68.

[14] J. J. Bosch, J. Janer, F. Fuhrmann, and P. Herrera, “A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals,” *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, pp. 559–564, Jan. 2012.

[15] R. Collobert and S. Bengio, “SVMTorch: Support Vector Machines for Large-scale Regression Problems,” *J. Mach. Learn. Res.*, vol. 1, pp. 143–160, Sep. 2001.

[16] S. Essid, G. Richard, and B. David, “Musical instrument recognition on solo performances,” in *2004 12th European Signal Processing Conference*, 2004, pp. 1289–1292.

[17] J. Aucouturier and F. Pachet, “Improving Timbre Similarity: How high’s the sky?,” *J. Negat. Results Speech Audio Sci*, vol. 1, May 2004.

[18] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard, “YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software.,” 2010, pp. 441–446.

[19] P. Herrera-Boyer, G. Peeters, and S. Dubnov, “Automatic Classification of Musical Instrument Sounds,” *Journal of New Music Research*, vol. 32, no. 1, pp. 3–21, Mar. 2003.

[20] R. Rifkin and A. Klautau, “In Defense of One-Vs-All Classification,” *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, Dec. 2004.

[21] M. H. Hassoun, *Fundamentals of artificial neural networks*. Cambridge, Mass.: MIT Press, 1995.