Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Aims and Objectives

1.2 Overview of the Report

Chapter 2

Literature Survey

2.1 Music Features

Available music clips usually have a sampling rate about 22050 or 44100 Hz. Using all the samples without processing in Content-Based Music Information Retrieval tasks is not time and memory efficient. (Murthy & Koolagudi, 2018). To solve this problem, a number of features which integrates information in music recordings can be extracted to simplify the work.

2.1.1 Feature Taxonomy

An paper in 2006 summarized music's audio features in three categories: timbre, melody and rhythm. It also (Scaringella, Zoia, & Mlynek, 2006) This work is significant because it established the framework in feature extraction which is still followed by recent researchers.
Based on this summary, (Fu et al., 2010) further categorized timbre and temporal features as low-level features which can be easily extracted after framing and Fourier Transformation, while rhythm, pitch and harmony are as mid-level features which can describe the intrinsic characteristics of music. This paper is important because it explained all features' roles as descriptors, whether they resemble information in a long period or a time frame and, whether they contain more intuitive properties. Table 2.1 in this paper summarized widely used low-level features.

Table 2.1: Common low-level features from Fu et al. (2010)

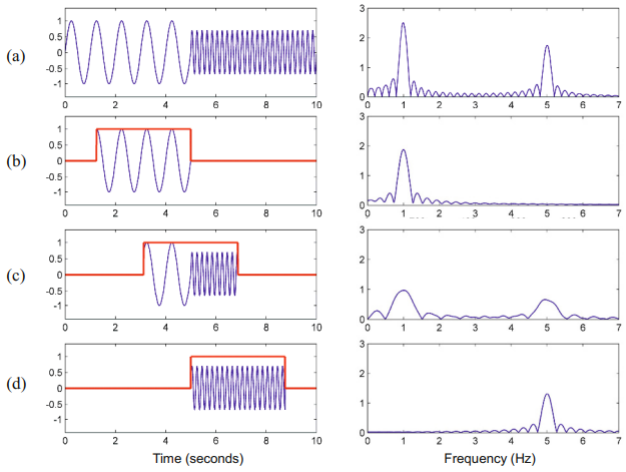
Class	Feature Type
Timbre	Zero Crossing Rate (ZCR)
	Spectral Centroid (SC)
	Spectral Rolloff (SR)
	Spectral Flux (SF)
	Spectral Bandwidth (SB)
	Spectral Flatness Measure (SFM)
	Spectral Crest Factor (SCF)
	Amplitude Spectrum Envelop (ASE)
	Octave based Spectral Contrast (OSC)
	Daubechies Wavelet Coef Histogram (DWCH)
	Mel-frequency Cepstrum Coefficient (MFCC)
	Fourier Cepstrum Coefficient
	Linear Predictive Cepstrum Coefficient (LPCC)
	Stereo Panning Spectrum Features (SPSF)
Temporal	Statistical Moments (SM)
	Amplitude Modulation (AM)
	Auto-Regressive Modeling (ARM)

Also, a number of researches proposed new features or mothods to learn features. For example, OSCCs are able to differentiate instrumental sound from voices(Maddage, Xu, & Wang, 2004), which can help to classify classical, ambient music from pop and jazz. At 2011, Costa suggests extracting features from spectrograms, the visual presentation of an audio and finally improved the classification accuracy(Y. M. Costa, Oliveira, Koericb, & Gouyon, 2011). This work is vital because it casts a new light to apply algorithms in image processing on audio processing. At 2014, Huang et al use adaptive harmony selection algorithm to search more relevant features, this method halves the dimensionality, combined with ensemble classifier which contains binary SVM classifiers for each pair of genres(Huang, Lin, Wu, & Li, 2014). Their work reaches 97.2% accuracy and is considered as the state-of-art approch.

2.1.2 Log-amplitude Mel-spectrogram

As a visual representation of audio clip, spectrograms shows how energy of each frequency changes over time. It is an extensively used feature to measure timbre in voiceprint recognition, instrument recognition and music classification tasks.

Figure 2.1: Fourier transformation on waveform frames from Müller (2015, p. 54)



The first step is to split the signal into equally long parts. In Figure 2.1, (a) is the the original audio. Left figures in (b) (c) and (d) are the windowed waveform which is the visual representation of audio on time domain. These segments can be partially overlapped to each other.
The second step is to convert waveforms to spectrums which is frequency domain representation. It can be done with discrete short-time Fourier Transformation (STFT) on each subsignal x. The k^{th} Fourier coefficient of the m^{th} window, i.e. each element in metrix $\mathcal{X}$ is given by:

$$\mathcal{X}(m,k) := \sum_{n=0}^{N-1} x(n+mH)w(n)\exp(-2\pi ikn/N)$$

By square every Fourier coefficients, the spectrogram can be computed:

$$\mathcal{Y}(m,k) := |\mathcal{X}(m,k)|^2$$

Figure 2.2: Convert waveform to Spectrogram from Müller (2015, p. 56)

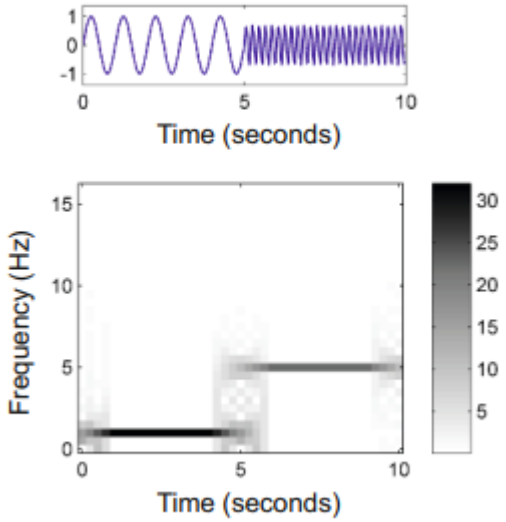


Fig 2.2 shows the waveform of an audio clip and its correspondent spectrogram. The horizontal axis represents time positions while the vertical axis represents frequencies. The shaded part means the squared coefficients and in this example darker pixles imply higher amplitudes.
Mel-scale, a psycho-acoustic concept is widely introduced in audio processing to measure frequency. The idea is that human's perception of sound is nonlinear and a same frequency gap in hertz can be less noticable at high pitch (Oppenheim & Magnasco, 2013). By scaling the y-axis in mel-scale, the mel-spectrogram can describe how amplitude changes over time and frequency from human hearing's perspective. The formula to convert hertz-scale to mel-scale is given as

$$m = 2595 \log_{10} \left(1 + \frac{f}{700}\right)$$

2.1.3 Chromagram

Chromagram, a middle-level pitch representation, is robust to different timbres of various instruments. It is calculated based on the spectrogram mentioned above. It is extensively used in MIR tasks like music synchronization and chord recognition.
In order to convert spectrogram (b) to chromagram (d) in Fig 2.3, firstly convert it to log-frequency spectrogram (c) using logarithmic scale to make the frequency depends on music notes which are linearly distributed. Then assign the signal into 12 equal temperaments which is

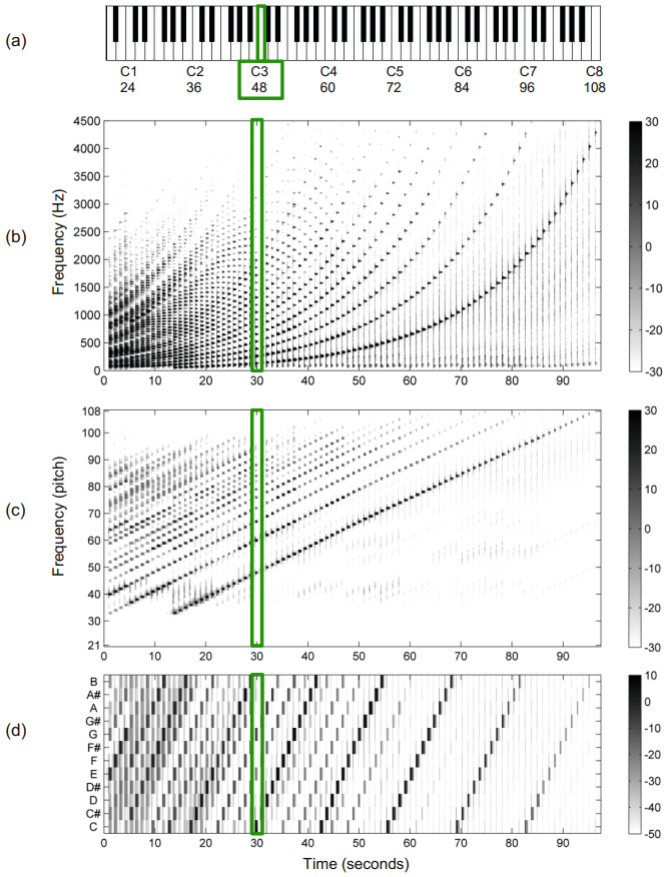
$$\{C, C^\sharp, D, D^\sharp, E, F, F^\sharp, G, G^\sharp, A, A^\sharp, B\}$$

irrespective to the octave, C_0, C_2, C_4 will all be aggregated as C.

2.2 Deep learning

Deep learning is a cutting-edge subfield of machine learning using models called Artificial Neural Networks (ANN) whose structures are inspired by human's brain. Fig 2.4 shows

Figure 2.3: Convert spectrogram to chromagram from Müller (2015, p. 119)

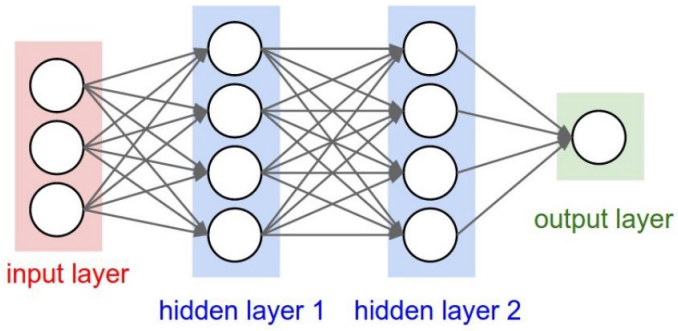


the the most common ANN structure which starts with an input layer, with least one hidden layer followed by and end up with an output layer.

2.2.1 Activation Functions

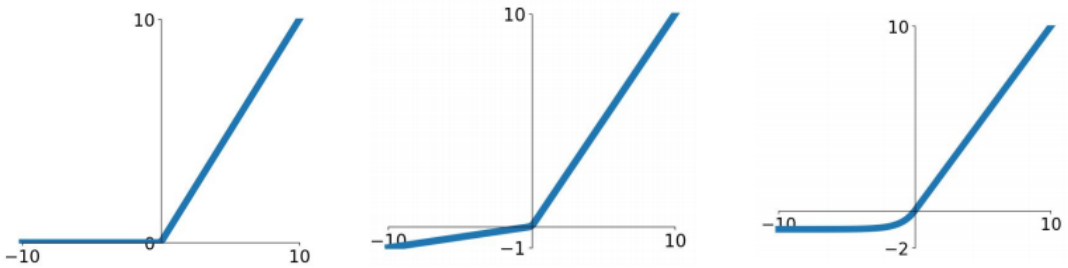
In ANN an activation function take the output of a node as input, and it's output will be feed forward to the next layer. It decides how a cell is fired or activated with respective to its value. Activation functions can't be linear functions in order to solve nonlinear problems.
Fig 2.5 shows the linear unit family and their output when $x < 0$. A popular example is ReLU(rectified linear unit) which takes the maximun between input and 0 as the output. However, an problem of ReLU is killing neuron, cause weights never update again because of 0 gradient when $x < 0$. As an alternative, leaky ReLU solved this problem by giving a tiny weight when $x < 0$ rather than simply set to 0. However, leaky ReLU's linearity makes it lag behind nonlinear functions in complex problems. As a powerful alternative of ReLU, ELU slowly smoothes when the input is negative. Its nonlinearity makes it able to solve nonlinear problems like music genre classification.

Figure 2.4: A basic ANN model



http://cs231n.github.io/neural-networks-1

Figure 2.5: ReLU, leaky ReLU and ELU



(a) ReLU
$f(x) = 0$

(b) leaky ReLU
$f(x) = \alpha x$

(c) ELU
$f(x) = \alpha(\exp(x) - 1)$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture6.pdf

2.2.2 Convolutional Neural Network

CNN(Convolutional Neural Network) is a network structure which do computation specifically on grid-like data types which includes images or time series. Convolutional network mainly consists three parts, convolution layer, activation function and pooling layer.

Convolutional Layer

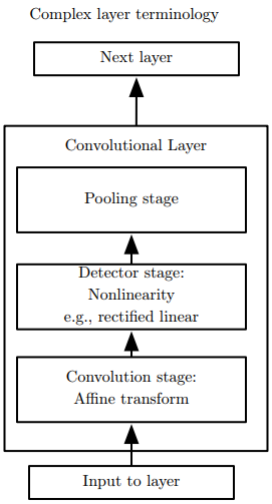
Fig 2.7 and the equation below shows that convolutional operation is applying a filter named kernel K on input grid data I, and then sum their element-wise product as the element on the output S.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

Pooling Layer

Another important operation in CNN is Pooling which is designed to compress the grid-like data after applying convolution operation and activation functions to simplify the

Figure 2.6: Convolutional network structure LeCun et al. (2015)



model to avoid overfitting and accelerate learning parameters by aggregating values in a pooling window using their maximum or average. Fig 2.8 illustrates the max pooling operation.

Inception Module

Inception module is proposed in 2014 as the core component of GoogLeNet, which achieved a dominant position in ILSVRC14 (Szegedy et al., 2015). This module can effectively avoid the waste of computation resources as the computation increase exponentially when the number of kernels increased in each of serially connected convolutional layers. Fig 2.9(a) is the naive version of Inception module which does convolution operations with three different sizes of kernels and max pooling parallelly to extract higher and lower level features and then concatenate them together. Fig 2.9(b) makes use of 1x1 convolution which can not only enable the model to be wider and deeper by reducing the dimension of input before expensive 3x3 and 5x5 convolutions but also address overfitting by simplifying the model, reducing the number of parameters.

2.2.3 Recurrent Neural Network

RNN is a network structure which takes sequence-like data $x^{(1)}, \ldots, x^{(\tau)}$ includes videos, audios and sentences as inputs. The recurrence formula applied on every vectors in the sequence shows that parameters of each time step are shared, so there's no predetermined limit on the input size.

$$h_n = \sigma\left(\mathbf{W}_h h_{n-1} + \mathbf{W}_x x_n\right)$$

The Fig 2.10 gives the structure of Vanilla RNN. Every hidden states h depend not just on memory until the previous inputh_{n-1} but also the current input x_n while generating outputsy_n. However, Vanilla RNN sufffers from vanishing gradients which can cause network forget earlier inputs when the sequence is relatively long.

Figure 2.7: Converlutional operation from LeCun et al. (2015)

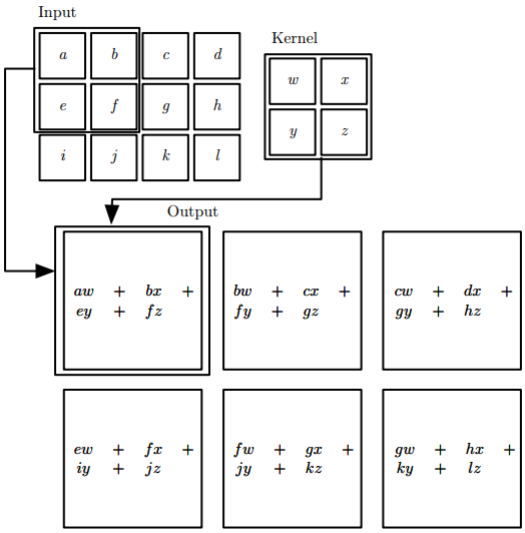
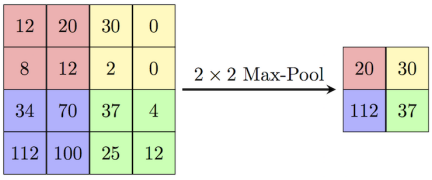


Figure 2.8: Max pooling



https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

Gated Recurrent Unit

As a strong alternative, GRU(Gated Recurrent Unit) is designed to solve the shrinking memory problem by using gates to control which information to keep or forget. The structure of GRU shown in Fig 2.11 contains Reset gate.

$$\mathbf{R}_t = \sigma\left(\mathbf{X}_t\mathbf{W}_{xr} + \mathbf{H}_{t-1}\mathbf{W}_{hr} + \mathbf{b}_r\right)$$
$$\mathbf{Z}_t = \sigma\left(\mathbf{X}_t\mathbf{W}_{xz} + \mathbf{H}_{t-1}\mathbf{W}_{hz} + \mathbf{b}_z\right)$$

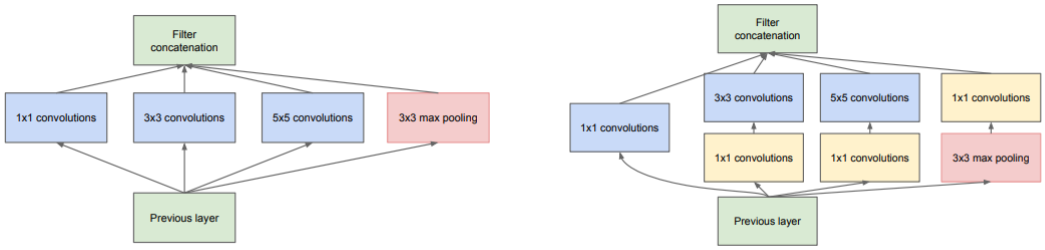
By taking current input $\mathbf{X}_t$ and last moment's hidden state $\mathbf{H}_{t-1}$ as inputs, $\mathbf{R}_t$ and $\mathbf{Z}_t$ can be calculated with different weights. Then the candidate hidden state can be calculated by taking the Hardmard product of $\mathbf{R}_t$ and $\mathbf{H}_{t-1}$, a larger $\mathbf{R}_t$ means the Reset gate keeps more memory.

$$\tilde{\mathbf{H}}_t = \tanh\left(\mathbf{X}_t\mathbf{W}_{xh} + \left(\mathbf{R}_t \odot \mathbf{H}_{t-1}\right)\mathbf{W}_{hh} + \mathbf{b}_h\right)$$

Then the hidden state can be calculated by the following equation in which if more memory are kept then more information of the current input will be forgotten.

$$\boldsymbol{H}_t = \mathbf{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$

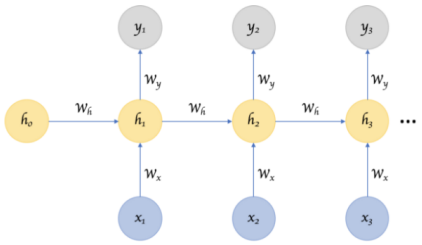
Figure 2.9: Inception Module from Szegedy et al. (2015)



(a) Inception module, naïve version

(b) Inception module with dimensionality reduction

Figure 2.10: Recurrent Neural Network



https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Bidirectional RNN

Bidirectional RNN can include information from unseen timesteps by splitting RNN units into forward and backward directions. It helps RNN adapt to more domains which need contextual data such as speech recognition, handwritting recognition and bioinformatics.

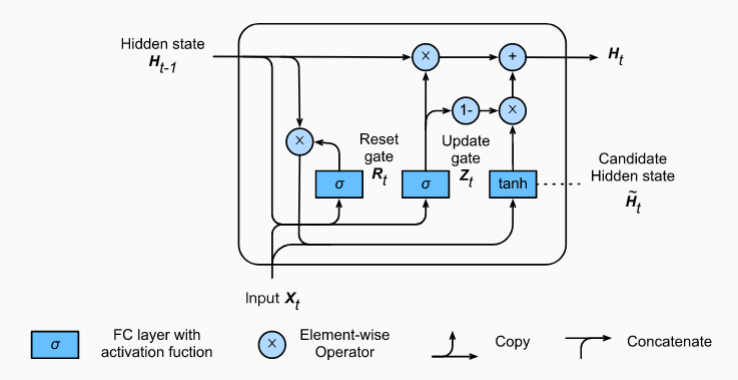
2.3 The Task: Genre Classification

Genre classification is a core task in music information retrieval (MIR). The goal is to classify and annotate a music clip to its most relevant genre like jazz, ambient, pop and so on or a subgenre like tango and post-rock. Three areas in this task are studied intensively, music features extraction, classification algorithm and the taxonomy of music genres.

2.3.1 Traditional Machine Learning Solutions

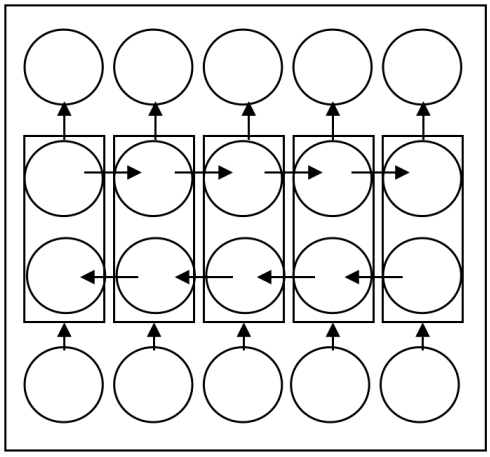
The earliest and most influential genre classification is George Tzanetakis work in 2002 (Tzanetakis & Cook, 2002). GTZAN, the dataset they provided is widely recognized as one of the benchmarks for the genre classification task. It's a significant work because firstly, it innovatively proposed three sets of features: timbre, rhythm and pitch, which is a prototype of today's feature framework. Secondly, it trains several statistical pattern recognition classifiers includes Gaussian, GMM and KNN and compares their performances. Also, for those short-term a.k.a window-based features, it gives the effect of window size on the performance. All of those are vital parts in classification tasks.
However, George's work used 30 seconds of clips which may not be feasible and time-efficient. For the length of the segment, a research at 2006 suggests 2-5 seconds audio provides sufficient information for classification(Bergstra, Casagrande, Erhan, Eck, &

Figure 2.11: Gated Recurrent Unit



https://d2l.ai/chapter_recurrent-neural-networks/gru.html

Figure 2.12: Bidirectional RNN



https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks

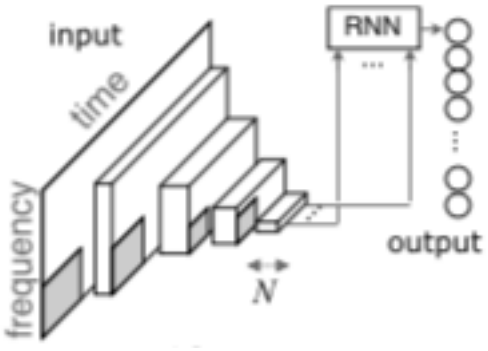
Kégl, 2006). This work uses Adaboost, fits each feature with gaussian then use mean and variance of to feed the classifier and eventually get a 82% accuracy.

2.3.2 Deep Learning Solutions

Nevertheless, all the works mentioned above use the means and variances of short-term features in every window as descriptors, which means the relative position of each window is ignored. In spite of that, conventional machine learning algorithms depends on feature engineering, but deep learning can solve it(LeCun et al., 2015). In this decade, more researches make utility of deep neural networks. In 2010, it was the first time that convolutional neural network is applied on genre classification task(Li, Chan, & Chun, 2010). This work is innovative because it provides the idea that using a CNN classifier on a matrix of MFCCs whose height is the number of windows while the width is the number of MFCCs in each window. However, this model is severely overfitted because of the absence of other features bottlenecked the performance. RNN can also be used exclusively to classify genres. In 2018, Tang et al. proposed a hierarchical LSTM RNN(Tang, Chui, Yu, Zeng, & Wong, 2018) which first classify "strong" and "mild", then do sub-

classification to predict which specific genre it is in all "strong" genres (pop, rock, rap and so on). This model is novel and explainable, but the performance is not decent because they use MFCCs as the only feature.

Figure 2.13: CRNN by Keunwoo Choi

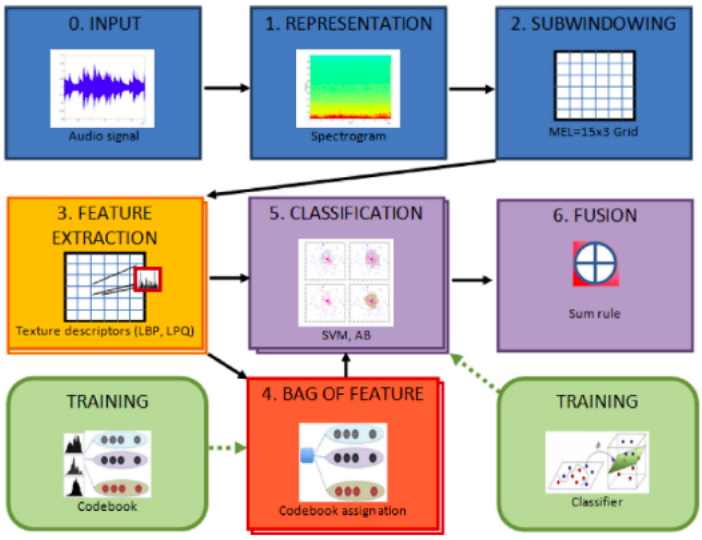


Doing image classification on spectrograms is another promising trend. After Costa suggests extracting features from spectrograms in 2011. He introduces ideas like LBP(local binary patterns), Garbor filter and LPQ come from computer vision area to classify spectrograms(Y. M. Costa, Oliveira, Koerich, Gouyon, & Martins, 2012)(Y. Costa, Oliveira, Koerich, & Gouyon, 2013) and get accuracy higher than 80%. Based on these researches, in 2016, Loris et al proposed a pipeline (see Figure 2.14) which can resemble features and fusion classifiers. Their experiment proves that fusion classifier overperforms single classifier (Nanni, Costa, Lumini, Kim, & Baek, 2016). In 2015, Piczak et al apply CNN to classify environment sounds' spectrograms (Piczak, 2015), this work is influential and inspires similar approaches on music genre classification. In 2017 Keunwoo Choi applied the CNN model but use log-amplitude mel-spectrograms as input alternatively and find the accuracy is lifted significantly, he also combines the CNN model and RNN which aggregate temporal features to further imporve the performance(see Figure 2.13). In 2018 an experiment shows that the spectrogram classification method outperformes handcraft features classification on GTZAN dataset(Bahuleyan, 2018).

2.4 Genre Taxonomy

An widely cited paper published in 2000 suggests several principles should be followed when design a music taxonomy (Pachet & Cazaly, 2000), they are:
1. Objectivity: music's genre should be decided with respective to its traits which can differentiate it from its father and neighbors rather than the listeners' fillings.
2. Independence: each leave genre can only have one father node. Situations such as Pop>Punk and Rock>Punk can't coexist.
3. Similarity: the similarities between each related genres can be traced. For example, after defined genres objectively, we can find that both post rock and ambient are usually non-vocal.
4. Consistency: the advent of new genre is compatible to the taxonomy, which means a new type of music can be assigned as a sub-genre of a existed father genre.
A widely recognised music genre taxonomy is musicmap which provides the genealogy of music genres from 1870 to 2016. This taxonomy obeys the rules mentioned above.

Figure 2.14: pipeline for combine visual features



Genre tags in the database GTZAN are mutually independent, nine of them are father genres in musicmap and disco is a subgenre of R&B which is also one of the father genres. However, some songs are repeated or mislabelled in this database(Sturm, 2013). Another widely used dataset is ISMIR2004 which includes 6 genres: classical, electronic, jazz_blues, metal_punk, rock_pop and world. These genres don't correspondent to the taxonomy and wikipedia's definition, because punk is a subgenre of rock. There's also repitition in dataset MagnaTagATune: both new age and ambient are treated as genres but the former is the subgenre of the latter.

Chapter 3

Methodology

3.1 Work Analysis

3.1.1 Datasets

GTZAN

Several commonly used music dataset for genre classification task is compared in the last chapter. GTZAN is chosen to benchmark the performance of each model and the contribution of changes in model architectures as it contains balanced and well-defined genre labels. This dataset includes 100 roughly 30-seconds audio clips for each of 10 general genres: Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Raggae, Rock.

Extended Ballroom

This dataset is an extended version of Ballroom(Marchand & Peeters, 2016). It contains 4180 songs from 13 subgenres in ballroom dance. This one is also used to evaluate the final model's performance on classify subgenres and imbalanced classes.

Table 3.1: Extended Ballroom dataset

genre	number
Chacha	455
Jive	350
Quickstep	497
Rumba	470
Samba	468
Tango	464
Viennesewaltz	252
Waltz	529
Foxtrot	507
Pasodoble	53
Salsa	47
Slowwaltz	65
Wcswing	23
Total	4180

3.1.2 Programming Language and Toolkits Selection

Python 3.7 is selected for the implementation of this project because of its popularity in data science and machine learning fields. Anaconda is used as the Environment to manage packages. After the dataset is downloaded, features such as spectrograms and chromagrams can be extracted with LibROSA. The network is implemented in Keras with Tensorflow backend. These libraries are introduced as follows.

LibROSA

LibROSA is a Python package designed to fill the gap in music information retrieval since there's no stable tool in audio and music processing (McFee et al., 2015). Parameters are available to tune for functions in it, which provides flexibility for users with domain knowledge. This module can participate in extractions, transformation, and display of a range of fundamental and principle features such as MFCC, Constant-Q chromagram, and tempogram. LibROSA is intensively involved in audio processing and feature extraction of this project.

Keras

Keras is a high-level API that can run on the top of Tensorflow. It is designed for researchers because the rapid implementation of neural network architecture allows ideas to be developed and evaluated with least delay. Tensorflow, Keras' backend, is an open-source deep learning framework which gains popularity in recent years. It introduced the concept of computation graph and tensor, which makes the logic and structure of a deep learning model more transparent and understandable. Tensorflow-GPU backend is used in this project as computations can be accelerated significantly on Cuda-supported GPUs.

3.2 Feature Extraction & Audio Segmentation

Log-amplitude Mel-spectrogram and chromagram over twelve temperaments are extracted as features. In the training phase, instead of building a giant classifier taking fixed 30 seconds clips as inputs, each clip is segmented into 6 5-second frames all of which are labelled as the genre of the original audio. Testing phase is inspired by ensemble algorithms in machine learning, every clip in test set is also segmented into 5-second frames and the output probabilities on continuous $n(1 \leq n \leq 6)$ frames are averaged to represent the output over continuous $5n$ seconds audio.

3.3 Model Design

Three fundamental model architectures are proposed in this project, SBM (Serial Blocks Model), MLM (Multi-Level Model), and HIM (Hybrid Inputs Model). They are partially inspired by some researches and consider music's characteristics.

3.3.1 Model 1: SBM (Serial Blocks Model)

3.3.2 Model 2: MLM (Multi Level Model)

3.3.3 Model 3: HIM (Hybrid Inputs Model)

3.4 Evaluation

3.4.1 Cross Validation

3.4.2 Evaluation Metrics

3.4.3 Confusion Matrix

Chapter 4

Experiments and Results

4.1 SBM performances

4.1.1 SBM CNN

4.1.2 SBM Inception

4.2 MLM performances

4.2.1 MLM CNN

4.2.2 MLM Inception

4.3 HIM performances

4.3.1 SBM CNN

4.3.2 SBM CNN + GRU

4.3.3 SBM Inception + GRU

4.4 Comparison with Recent Researches

Chapter 5

Conclusion

5.1 Further Work

5.1.1 Music Preference Analysis

5.1.2 Content-based Music Encoder

References

Bahuleyan, H. (2018). Music genre classification using machine learning techniques. *arXiv preprint arXiv:1804.01149*.

Bergstra, J., Casagrande, N., Erhan, D., Eck, D., & Kégl, B. (2006). Aggregate features and a da b oost for music classification. *Machine learning*, *65*(2-3), 473–484.

Costa, Y., Oliveira, L., Koerich, A., & Gouyon, F. (2013). Music genre recognition using gabor filters and lpq texture descriptors. In *Iberoamerican congress on pattern recognition* (pp. 67–74).

Costa, Y. M., Oliveira, L., Koerich, A. L., Gouyon, F., & Martins, J. (2012). Music genre classification using lbp textural features. *Signal Processing*, *92*(11), 2723–2737.

Costa, Y. M., Oliveira, L. S., Koericb, A. L., & Gouyon, F. (2011). Music genre recognition using spectrograms. In *2011 18th international conference on systems, signals and image processing* (pp. 1–4).

Fu, Z., Lu, G., Ting, K. M., & Zhang, D. (2010). A survey of audio-based music classification and annotation. *IEEE transactions on multimedia*, *13*(2), 303–319.

Huang, Y.-F., Lin, S.-M., Wu, H.-Y., & Li, Y.-S. (2014). Music genre classification based on local feature selection using a self-adaptive harmony search algorithm. *Data & Knowledge Engineering*, *92*, 60–76.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

Li, T. L., Chan, A. B., & Chun, A. H. (2010). Automatic musical pattern feature extraction using convolutional neural network. *Genre*, *10*, 1x1.

Maddage, N. C., Xu, C., & Wang, Y. (2004). Singer identification based on vocal and instrumental models. In *Proceedings of the 17th international conference on pattern recognition, 2004. icpr 2004.* (Vol. 2, pp. 375–378).

Marchand, U., & Peeters, G. (2016). The extended ballroom dataset.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference* (Vol. 8).

Müller, M. (2015). *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer.

Murthy, Y., & Koolagudi, S. G. (2018). Content-based music information retrieval (cb-mir) and its applications toward the music industry: A review. *ACM Computing Surveys (CSUR)*, *51*(3), 45.

Nanni, L., Costa, Y. M., Lumini, A., Kim, M. Y., & Baek, S. R. (2016). Combining visual and acoustic features for music genre classification. *Expert Systems with Applications*, *45*, 108–117.

Oppenheim, J. N., & Magnasco, M. O. (2013, Jan). Human time-frequency acuity beats the fourier uncertainty principle. *Phys. Rev. Lett.*, *110*, 044301. Retrieved from `https://link.aps.org/doi/10.1103/PhysRevLett.110.044301` doi: 10.1103/PhysRevLett.110.044301

Pachet, F., & Cazaly, D. (2000). A taxonomy of musical genres. In *Content-based multimedia information access-volume 2* (pp. 1238–1245).

Piczak, K. J. (2015). Environmental sound classification with convolutional neural networks. In *2015 ieee 25th international workshop on machine learning for signal processing (mlsp)* (pp. 1–6).

Scaringella, N., Zoia, G., & Mlynek, D. (2006). Ref-196 automatic genre classification of music content: a survey. *IEEE Signal Processing Magazine*, *23*(2), 133–141.

Sturm, B. L. (2013). The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint arXiv:1306.1461*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1–9).

Tang, C. P., Chui, K. L., Yu, Y. K., Zeng, Z., & Wong, K. H. (2018). Music genre classification using a hierarchical long short term memory (lstm) model. In *Third international workshop on pattern recognition* (Vol. 10828, p. 108281B).

Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, *10*(5), 293–302.