Masked Conditional Neural Networks for Sound Recognition

Fady Medhat

Ph.D.

Electronic Engineering

University of York

March 2018

Abstract

Sound recognition has been studied for decades to grant machines the human hearing ability. The advances in this field help in a range of applications, from industrial ones such as fault detection in machines and noise monitoring to household applications such as surveillance and hearing aids. The problem of sound recognition like any pattern recognition task involves the reliability of the extracted features and the recognition model. The problem has been approached through decades of crafted features used collaboratively with models based on neural networks or statistical models such as Gaussian Mixtures and Hidden Markov models. Neural networks are currently being considered as a method to automate the feature extraction stage together with the already incorporated role of recognition. The performance of such models is approaching handcrafted features. Current neural network based models are not primarily designed for the nature of the sound signal, which may not optimally harness distinctive properties of the signal.

This thesis proposes neural network models that exploit the nature of the time-frequency representation of the sound signal. We propose the ConditionaL Neural Network (CLNN) and the Masked ConditionaL Neural Network (MCLNN). The CLNN is designed to account for the temporal dimension of a signal and behaves as the framework for the MCLNN. The MCLNN allows a filterbank-like behaviour to be embedded within the network using a specially designed binary mask. The masking subdivides the frequency range of a signal into bands and allows concurrent consideration of different feature combinations analogous to the manual handcrafting of the optimum set of features for a recognition task. The proposed models have been evaluated through an extensive set of experiments using a range of publicly available datasets of music genres and environmental sounds. For example, our proposed model achieved 92.1% for a music genre recognition task and 85.5% for environmental sound classification compared to 87% and 80% achieved by state-of-the-art Convolutional Neural Networks for either task, respectively. In addition, it surpasses several hand-crafted attempts.

Acknowledgements

I want to thank my supervisors, Dr. David Chesmore and Prof. John Robinson.

Dr. David is a great mentor who gave me all the support, encouragement and freedom to pursue my own research ideas. I want to express my deepest gratitude and appreciation to have the opportunity of being his student.

I would like to thank Prof. John for all the advice and the insightful comments throughout the whole PhD. I am grateful for him finding me the time for meetings despite his very busy schedule.

I would like to thank my dearest officemate and PhD companion, Chiara Picardi for all the nice and interesting talks we had together and all the discussions about the ups and downs in our PhDs. It was lots of fun to be in the same office with you.

I am grateful for being part of the CAPACITIE project and I would like to thank all the CAPACITIE fellows: Xiu Gao, Rina Siyengwa, Emily Burns, Magdalena Kruza, Jagannath Biswakarma, Prado Domercq, Michelle Wang, Elena Koutsoumpeli, Kyle Stevens, Mayank Parmar, Xinwei Fang and Gabor Makrai and special thanks to Prof. Alistair Boxall and Dr. Lorraine Youds.

Last but not least, I would like to thank my Dad, missing you and wished you were around for this day, my Mom and my Sister for their continuous support along the PhD journey.

European Commission

The project has received funding from the European Union’s Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 608014

To the memory of my Dad

Declaration

I declare that this thesis is a presentation of original work. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References. The research in this thesis is featured in a number of author's publications as listed below.

Publications

Fady Medhat, David Chesmore, and John Robinson, " Masked Conditional Neural Network for Sound Classification", submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence, (TPAMI), 2018.

Fady Medhat, David Chesmore, John Robinson, "Automatic Classification of Music Genre using Masked Conditional Neural Networks," in IEEE International Conference on Data Mining (ICDM), 2017

Fady Medhat, David Chesmore, and John Robinson, " Masked Conditional Neural Networks for Audio Classification", in International Conference on Artificial Neural Networks (ICANN), 2017.

Fady Medhat, David Chesmore, and John Robinson, " Music Genre Classification using Masked Conditional Neural Networks ", in International Conference on Neural Information Processing (ICONIP), 2017.

Fady Medhat, David Chesmore, and John Robinson, "Masked Conditional Neural Networks for Automatic Sound Events Recognition", in IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2017.

Fady Medhat, David Chesmore, John Robinson, "Environmental Sound Recognition using Masked Conditional Neural Networks," in International Conference on Advanced Data Mining and Applications (ADMA), 2017.

Fady Medhat, David Chesmore, John Robinson, " Masked Conditional Neural Networks for Environmental Sound Classification," in International Conference on Artificial Intelligence (AI), 2017

Fady Medhat, David Chesmore, John Robinson, "Recognition of Sound using Masked Conditional Neural Networks," in IEEE International Conference on Machine Learning and Applications (ICMLA), 2017

Contents

List of Figures

List of Tables

1 Introduction

URBAN city pollutants are increasingly becoming a concern to authorities and legislators. The spread of pollutants in air, water and soil are affecting all aspects of human life. The effect is not just confined to humans, but animals and natural life are also affected considerably. Accordingly, environmental studies are attracting the attention of research institutions and funding bodies.

CAPACITIE (Cutting Edge Approaches for Pollution Assessment in Cities) is a research project aiming to find innovative solutions for assessing and monitoring pollutants in cities. The project is composed of several work packages targeting different types of pollutants, e.g. noise, air and water. Noise pollution is one of the concerns of the project and the subject of this thesis.

A question needs to be put forward before continuing the discussion on how to tackle the noise problem further: "*Why is it important to control noise?*" Many researchers [1-4] have investigated the answer to this question through finding out the hazardous effects of noise on humans and the environment. These show a direct link between noise and several human illnesses, e.g. hearing impairments, anxiety, cardiovascular diseases and annoyance to name a few. Accordingly, it is required to lessen and hopefully eliminate the hazardous effect of noise, and clearly, there are shortcomings in the methods used to do so. Elaborating on the shortcomings of noise control, the collective measure of environmental sounds in an assessed location does not specify which sound sources are contributing to the total noise level, where several sound sources of different nature contribute to the environmental noise.

1.1 Background

Environmental noise is very complex in nature with several overlapping sounds that have to be associated with the sources generating them to judge the level of annoyance objectively. To elaborate on this, considering the sound of a waterfall, it may record high dB levels from a microphone very near to its location, or the sound of a flock of birds nesting in a tree may even give similar readings. But do humans feel annoyed from these categories of sounds compared to similar dB levels measured from a highway due to moving vehicles? The answer to this question was investigated by Landstrom et al. [5], where they found that there is a weak connection between the annoyance of people and loudness of the sound. On the other hand, it was people exposed to noise with low tonal components (generated by machinery in general including vehicles) who were much more annoyed. Another study by Leventhall [6] shows that low-frequency components, 10Hz-200Hz, strongly affect human annoyance. These studies show the need for effective and smart methods for noise monitoring

Environmental noise monitoring needs to be assessed over long durations to capture the change in the soundscape. This is very similar to the identity of a place but based on acoustics. For example, a park possesses, generally, quiet levels of sound composed mainly of the sound of wind blowing through trees, birds sounds and maybe the voices of some children playing. Another example is the soundscape of a busy road; generally loud levels of sounds composed of vehicle engines, horns, footsteps on concrete sidewalks. The ability to recognize the sound components making up the environmental sound scene will help authorities in devising their action plans and feeding noise data to the public users. More importantly, defining the environmental sound sources contributing to the total noise level through knowing the sound components in the mixture can help in targeting the most hazardous sources, e.g. vehicle sounds.

1.2 Objective and Methodology

Sound recognition investigates the possibility of transferring the human ability to distinguish sounds and embedding it into a machine. The problem has been the focus of the research community for decades. The sound recognition problem captures advances in several subfields such as signal processing, features extraction and crafting and pattern recognition. Advances have accumulated over decades in each of these fields, and the identification of sound has not been entirely solved yet.

The problem is faced with too many challenges that span: a) the type of sound, e.g. speech or music or environmental sounds, b) the sensitivity of the capturing device and the noise around it, which affects the quality of the captured signal, c) the quality of the discretization, i.e. the analogue to digital conversion, and d) the resolution of the stored signal, i.e. the sampling rate. The previous are only the challenges to capture and store a signal on a digital device. Other challenges relate to the type of preprocessing and feature extraction that fits the nature of the data captured, and finally, the pattern recognition algorithm, which is almost never hyperparameter-free, i.e. finding the optimum hyperparameters of a model that fits the data being considered is an exploration mission.

Hand-crafting the features extracted from signals such as sounds or images or videos requires tremendous effort. Recently, deep architectures of neural networks have managed to achieve remarkable results for image and video recognition feature extraction. The success of these deep architectures induced applying them to the sound recognition problem aiming to automate the feature extraction stage.

A multi-layer perceptron (MLP) [7] can be considered as a basic feedforward neural network used for pattern classification. In such a model, the network is introduced to a feature vector of a specific length matching the input of the network. The input of a temporal signal can be the raw sound signal or the frames of a spectrogram transformation of the signal or even a complex set of perfectly finetuned hand-crafted combination of features. The problem with such a classification scheme is that it ignores the sequential nature of a temporal signal as it treats each frame as an isolated entity. Recurrent Neural Networks (RNN) [8] introduce a feedback loop from the neurons' previous state to the current input, which allowed RNN models to be adapted to sequential signals. Long Short-Term Memory (LSTM) [9] is a descendant of the RNN that was introduced to tackle the some of the problems of the vanilla RNN. LSTMs achieved remarkable results in handwritten text recognition [10], speech [11] and images [12]. The remarkable performance of the RNN is accounted for by its ability to capture long-term dependences especially with their variants such as the LSTM by embedding a memory within the network. Convolutional Neural Networks (CNN) achieved wide success as well in image processing [13, 14], which extended its application to spectrograms of sound [15-18]. They were even considered as a replacement for the Gaussian Mixture Model (GMM) [19] widely used in combination with the Hidden Markov Model (HMM) [20-22]. Despite the successful attempts in using neural networks for feature extraction using deep neural networks, the models are usually adopted to sound after they gain wide acceptance in other domains especially image recognition, which may not optimally harness the nature of the sound signal.

In this thesis, we present a neural network architecture that is designed to harness the time-frequency domain representation of the sound signal. The model takes into consideration the temporal nature of a sound signal in addition to the spectral bands of a spectrogram. The models we propose are general enough to be applied for any multi-channel temporal signal representation.

An additional problem special to environmental sound recognition, compared to speech and music research, and that has hindered advances in this field is the scarcity of large annotated datasets. There are recent attempts [23, 24] to collate large datasets of environmental sounds, which we used for benchmarking the models proposed in this work. Additionally, we extended the dataset introduced in [24] by manually collecting sound samples with more emphasis on vehicle sounds (road and rail traffic). We also extended the evaluation of the models to widely adopted music datasets, since music has similar properties, to a certain extent, to environmental sounds in terms of the overlapping sound sources that have a longer duration than phonemes of speech.

1.3 Organization

The thesis is structured as follows:

> *Chapter 2* explores the history of sound recognition highlighting important landmark attempts used to tackle the problem.
>
> *Chapter 3* explores signal representation techniques that have been used across the years for sound recognition.
>
> *Chapter 4* highlights machine learning models that have been used for temporal signal modelling.

Chapter 5 provides a review of several neural network models that are widely adopted for pattern recognition especially sound.

Chapter 6 introduces the Conditional Neural Networks and its masked variant the Masked Conditional Neural Networks, which are the main contribution of this work.

Chapter 7 benchmarks the performance of the models proposed in this work through a set of extensive experiments using literature wide adopted sound datasets.

Chapter 8 provides an in-depth analysis of the model proposed in this work and an unbiased comparison to the state-of-the-art Convolutional Neural Networks.

Chapter 9 summarizes the contributions of this thesis and highlights future work.

2

History of Sound Recognition

SOUND recognition is a field that has always captured the attention of the research community and end users. The problem is intermingled between the features or the intermediate representation and the recognition system used. In this chapter, we will examine the history of sound recognition as a problem approached either from the side of the signal analysis and feature crafting or the perspective of the machine learning models adapted to the problem. Our review will avoid categorizing the methods at this early stage. We will rather attempt to highlight the methods, and through later chapters this form of categorization should be handled.

2.1 Speech Recognition

Speech recognition can be considered as the primary driver for the sound recognition problem, where later interest appeared in environmental sound recognition and several music information retrieval tasks such as music genre classification. 1952 marks the first speech recognizer by Davis et al. [25], AUDREY. It had large analogue circuitry to recognize spoken digits using the energy of a spectrogram split into two bands as a signal representation. Of course, AUDREY was proceeded with different artworks in signal analysis and recognition and more appeared later. One of such techniques used for signal analysis that appeared in 1965, was the work of Cooley and Tukey [26]. They devised the Fast Fourier Transform (FFT), an efficient method to calculate the Discrete Fourier Transform, which is still the primary signal transformation from the time to the frequency domain, at the time of writing. In the year 1966, the next year to devising the FFT, the Hidden Markov Model (HMM) was formulated by Baum and Petrie [20], and later the Viterbi algorithm by Forney [21] was introduced in 1973 to formalize the possible state transitions in an HMM. Though the HMM and FFT advances may seem unrelated at that time, later they joined forces to create efficient speech recognizers together with the Gaussian Mixture Models (GMM) [27] trained with the Expectation Maximization algorithm introduced by Dempster et al. [19] in 1977. The combination [22] between the GMM and the HMM was dominant for some time (at least until deep architectures started to gain much attention). This assembly was used extensively in speech recognition [28-30], where a GMM can model a phoneme distribution and a HMM models the temporal sequence relation between the frames of a phoneme.

2.2 Music Genre Classification

The interest in the sound recognition problem arose from a range of applications that appeared across the years beside speech recognition as a driver, primarily applications in Music Information Retrieval (MIR). MIR recently gained increasing attention from music industry leaders and businesses with the growing use of digital music content shared over the web. MIR involves several sub-areas according to the specificity of the task as discussed by Casey et al. [31] ranging from music identification, copyright monitoring, and melody detection to recommendation and genre recognition. The problem involves the ability to categorize music files to facilitate their retrieval based on the instruments played, the author, the type of music and other tags that usually have to be labelled manually to a music file and possibly influenced by the annotator's decision. These manual annotations are further used by other subsystems in the field of MIR, where the recommendation systems are the most obvious applications. These systems are built around the core challenge of fetching a music file that may appeal to the listener based on some piece of music that is played inside the actual content of the file, but practically it has to be through the "subjective" genre tags accompanying that file. Therefore, based on the listener's taste of one music genre and probably with the collaborative opinions of other listeners, the system can recommend a list of songs. Music genre classification involves challenges related to the large number of variations of musical instruments, musical notes and the introduction of a mix between two or more genres in the same music piece. Most of the time, a musical piece could also involve the presence of a human voice. A number of variables have to be considered for a classification decision, and automatic classification of different genres based on the audible music contents is required, at least to overcome the labour that goes into manual categorization.

Early attempts of music classification were in the mid-90s [32, 33] and possibly earlier. Back then most of the methods used were dependent on handcrafting the most prominent acoustic features for distinguishing between different music files and using either a distance measure clustering algorithm or simple neural network architectures [33] and Support Vector Machines (SVM) [34] in others [35]. An attempt of content based classification of audio signals was in the work by Wold et al. [32]. Their work used handcrafted features based on sound perception properties like loudness, pitch, brightness, bandwidth, and harmonicity. In their system, sounds were classified by calculating a distance measure between a new audio segment and the already categorized database of sounds. A similar attempt using hand-designed features and a Gaussian classifier was by Tzanetakis et al. in [36], and they later extended the work in [37] using a feature vector comprising timbral texture, rhythmic content and pitch content features classified using a GMM. In [38] Bergstra et al. achieved noticeable results on the music genre task using AdaBoost [39] as a classifier applied to several features. They also studied the effect of feature aggregation over a texture window on the classification accuracy. The work in [40] used a variation of the Self-Organizing Map (SOM) [41], introduced in 1981, for the classification of musical recordings of a clarinet based on features extracted from both the time and frequency domains.

Other approaches have also been considered, rather than the mix and match methods of choosing features that are widely adopted [42]. An example of these was the work by Holzapfel et al. [43], where Non-Negative Matrix Factorization (NMF) was used to extract the basis vectors from a spectrogram, acting as a dimensionality reduction technique as well, and using a Bag-of-Frames (BoF) [44], a GMM was constructed to represent each music genre. Similar approaches were considered in [45, 46]. Andén et al. in [47] used the scattering transform, achieving distinguishable results compared to other time-frequency representations applied for the music genre classification task, accounting to the frequency and time-wrapping invariant properties of the scattering transform.

Another development was the work by Henaff et al. [48]. They adapted the Predictive Sparse Decomposition (PSD) [49] to generate sparse representations of the input signal that are further classified using an SVM.

The work by Soltau et al. [33] can be considered as one of the initial references to the use of neural network architectures for feature extraction rather than for direct classification. Soltau used a three-layered architecture, where the first layer was a 10-node neural network dedicated to extracting audio events in a music file. The output layer of the network was dropped, and the activations of the hidden layer were used as an abstract low dimensional representation of the features for succeeding layers. The second layer in the proposed system is a statistical analysis layer to capture details about the events collected from the previous layer network's activations and finally a recognition layer with a neural network for the final classification. A similar advanced method was approached through the use of Restricted Boltzmann Machines (RBM) [50] by Hamel et al. [51]. They used three RBM layers, forming a Deep Belief Net (DBN) architecture, trained generatively on music spectrograms for feature extraction. The extracted features were further classified using an RBF-SVM. They showed in their work how each layer of the RBM captured an abstract representation of the data introduced to it. Convolutional DBN, a variant of DBN, was investigated by Lee et al. [52], which they used for unsupervised extraction of speech and music representations. In a different attempt aiming to bypass the need to transform the sound signal to an intermediate representation like a spectrogram, Dieleman et al. [53] applied a Convolutional Neural Network (CNN) [54] directly to the raw signal for the tagging problem of music files. Their results show that CNN was capable of tagging the music files, but still the spectrogram transformation prevails. CNN has also been studied for different music tasks in [55, 56]

2.3 Environmental Sound Recognition

One other field that captured the attention relates to the problem of Environmental Sound Recognition (ESR). Environmental sounds are very informative when it comes to specifying the soundscape of a region as either rural or urban, indoor or outdoor. The tonal characteristics and loudness of specific sound categories are also an indication of the hazardous noise levels that can cause long-term health problems [2-4] including anxiety, high blood pressure and cardiac diseases. Noise monitoring is a concern of legislators. For example, the Environmental Noise Directive (END) [57], specifies the types of sounds that require monitoring. These are mainly sounds of low tonal components [5] generated from air, rail and road traffic in addition to industrial site activities. These sound sources are indicative of the level of noise pollution and can pinpoint other linked pollution sources such as carbons and NOx components generated from engines. The monitoring and measuring standards may not be sufficient though. Current monitoring procedures involve the deployment of microphones or sound pressure level meters in specific locations, presumably locations expected to report high noise levels to measure the dB level generated, but these do not take account of the particular qualities and properties of the sounds that make them especially hazardous.

The environmental sound scene is made up of several sound components that may not be as hazardous to the human health and could show similar dB levels to birds singing or the sound of a waterfall. Therefore, more efficient noise monitoring should involve the recognition of the sound sources. ESR can be considered a more challenging recognition task compared to speech and music due to the absence of a clear structure for the sound compared to the use of phonemes in speech recognition and the perceptual properties of music, e.g. timbre, rhythm. Additionally, there is a wide pool of sounds for all events occurring in nature for which the unavailability of labelled data can hinder considering all categories in the recognition task. The problem is not only confined to the scope of automatic noise recognition [58, 59]. Information about the surroundings in robotic platforms that are dependent on computer vision and image processing can be leveraged from the additional cue the ESR can provide especially when these vision sensors are hindered by low lighting conditions. Similar settings apply for surveillance applications [60]. Another application for ESR involves the use of sound in the non-invasive detection of underground burrowing animals and insects in woodlands [61]. This application extends to quality monitoring of imported wood that can cause nationwide damage to woodlands if the imports embed an infestation. ESR has also found its way to be incorporated into smart homes and assisting the elderly [62, 63]. Additionally, as a visual-hearing aid for hearing impairments, where the environmental sound scene can be described on a mobile device, ESR can provide information about the user's surroundings visually and provide an alert in critical situations.

Efforts have tried to tackle the environmental sound recognition problem using a diversity of machine learning methods, mainly statistical attempts using Hidden Markov

Model (HMM) and Gaussian Mixture Model (GMM). One of the 1990s attempts was the work by Goldhor [64] using the likelihood measure and cepstral coefficients as features. He used a limited dataset in terms of size compared to other datasets appearing later in the literature, but still, his work marks the early interest of the research community in this problem. A similar early effort was by Gaunard et al. in [65], where a HMM was used for modelling five classes of sound, mostly sound categories overlapping with Environmental Noise Directive (END) [57] (i.e. rail, road, air traffic). They used Linear Predictive Coding (LPC) [66] for feature extraction and Vector Quantization (VQ) to implement a codebook for the extracted features and used a 5-state HMM for classification. Zhang et al. [67] used a HMM as well, but here they were aiming towards audio recording retrieval by developing a three stage approach incorporating a HMM as a method to classify environmental sounds. A notable appearance of a larger dataset compared to earlier works was in [58], where Dufaux et al. used a database of around 800 impulsive sounds, such as glass breaks, human screams, etc. which fits well in surveillance applications. They compared the performance of a Gaussian Mixture Model (GMM) and HMM at different Signal to Noise Ratios (SNR).

Recognition targeted for particular types of environmental sounds has also been considered in [68] for helicopter sound detection. Similarly, in [59], the authors studied recognizing nuisance sounds of scooters and horns. A specific example was in the work by Chesmore in [69], where he used simple time domain analysis to identify features for a neural network to classify sounds of 25 species of animals. Cowling et al. [70] aimed to review the widely used methods of speech analysis and adapt them for environmental sound.

In [71], Eronen et al. attempted to devise a scheme for context recognition. They evaluated the performance of several hand-designed time and frequency domain based features including Mel-Frequency Cepstral Coefficients (MFCC) on a dataset of 225 sound files from various environmental settings, e.g. street, restaurant, railway station, etc. aiming to find the best performing combination of features. They also considered Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Linear Discriminant Analysis (LDA) as dimensionality reduction techniques for the extracted features and compared the classification results of a HMM and a K-Nearest Neighbor (KNN). A similar attempt was in the work of Su et al. [72], but with the use of

Local Discriminant Bases (LDB) [73] for dimensionality reduction. Context recognition was also studied by Heittola et al. [74] using a GMM with a three-state HMM. Dictionary-based features were considered in the work of Chu et al. [75] using Matching Pursuits (MP) [76] and MFCC to classify among 14 classes of urban sounds, comparing the performance of a Gaussian Mixture Model (GMM) and K-Nearest Neighbors (KNN) as classifiers. A dictionary based approach was also proposed in [77], and an unsupervised feature learning scheme was considered in [78] using a mel-scaled spectrogram as an intermediate representation of the signal, PCA as dimensionality reduction and Spherical K-Means [79] for classification. Similarly, the same authors in [80] investigated the use of the scattering transform [47]. Wichern et al. [81] proposed a system targeted for environmental sound segmentation and retrieval in a yet another attempt using a HMM to model the sound category in a query-by-example paradigm. Their dataset can be considered limited, but their work focused more on the segmentation problem than classification.

The feature extraction stage has a substantial effect on the recognition systems' accuracy. A review was done by Chachada et al. [82] for ESR with more emphasis on the features. The classifier complements the extraction stage of a recognition system. Based on the works referenced earlier and others [83, 84], classifiers are mostly dependent on the GMM-HMM statistical combination. Neural-based attempts have been considered as well, but there is more interest appearing recently especially in applying deep architectures. Cakir et al. [85] made a notable effort to tackle the nature of the auditory environmental scene, in which several sounds usually overlap the same temporal instance. In their work, they used a deep neural architecture for a multi-labelling problem with a post-processing stage to alleviate the recognition confusion between overlapping events. The Convolutional Neural Network (CNN) [54] having extensive usage in image classification [13, 14] and speech [86], was considered by Piczak in [87] for ESR. In his work, Piczak used an architecture formed of two convolutional layers interleaved with two max-pooling layers applied on a logarithmic mel-scaled spectrogram. A similar but deeper structure was proposed in the work by Salamon [16]. Both works also considered the application of an augmentation stage to the dataset through several combinations of tweaks to the sound signal before introducing it to the network, which enhanced the classification accuracy to a certain extent. Another deep architecture was studied in the work by Hertel et al. [17], but they attempted to investigate the possibility of using the raw sound files for classification using a CNN to eliminate the need for the time-frequency representation of the spectrogram. Their work demonstrated the supremacy of frequency analysis experimentally as an intermediate representation.

Efforts using deep learning are being made in an attempt to automate the feature extraction stage and hopefully surpass the performance of hand-designed features. Nevertheless, hand-designed features [88-94] are still superior in most contexts compared to features extracted by other methods like deep neural architectures but the gap is getting smaller with deep learning evolving as a rival to minimize and hopefully eliminate the need to hand design the features required for classification.

2.4 Summary

In this chapter, a general overview of sound recognition as a problem studied for speech recognition, music genre classification and environmental sounds classification has been provided. Efforts were targeted to either handcrafting the most optimized features or introducing pattern recognition models that can exploit such features. We explored a wide range of attempts in each of these fields with an intention to highlight the challenges facing the research community in tackling the sound recognition problem. The following two chapters will approach the two folds of the sound recognition problem, i.e. the signal representation and the recognition models, in more detail.

3

Signal Representation

FEATURE extraction has a significant influence on the quality of the signal's intermediate representation introduced to a recognition model. Researchers are always seeking to engineer the most optimum features that can help in solving the recognition problem. A raw temporal signal has a lot of information, which is exploited using time-domain feature extraction methods, but there are far more details within the signal that are inaccessible within the time-domain and are attainable through the frequency domain. Eventually, it depends on the application and the available computational resources to choose either of them. Throughout the literature, a range of features have been proposed including both straightforward techniques and heavily engineered methods.

In this chapter, we will discuss examples of time-domain analysis and emphasize the discussion on frequency domain methods adopted widely for intermediate representation of the raw signal to a format comprehensible enough to allow recognition models to elicit distinctive properties for classification.

3.1 Time domain

The Zero Crossings rate can be considered as one of the most straightforward features that can be extracted from a raw signal in the time domain. The rate of the crossing of a signal to the horizontal axis is used in several speech recognition systems and sound analysis [95, 96]. Looking further into the zero crossing and how it is generated based on the analysis of the signals; in the absence of any pressure exerted on the diaphragm of a microphone, Gaussian noise will appear in the signal fluctuating across the horizontal axis. On applying a pressure on the diaphragm, the voltage starts to mimic the diaphragm movements. The frequency of vibration of the diaphragm while changing its shape from being concave to convex and the other way around, forming the zero crossings. Therefore, the frequency of occurrence of the zero-crossings across the buffer under consideration can be used as a distinction between different categories of sound. It has been used for Automatic Speech Recognition (ASR) [97] and sound classification [69]. Zero Crossings (ZC) is a simple time-domain measure of the number of times a signal crosses the horizontal axis in a specific duration.

Figure 3.1 shows the rate of ZC across 900 sound samples of human voice and a similar plot is shown for an air-conditioning unit in Figure 3.2. A 1971 study of the ZC for speech was in the work of Ito et al. [95], where they investigated the relationship between the ZC rate and the corresponding spectral representation of phonemes.

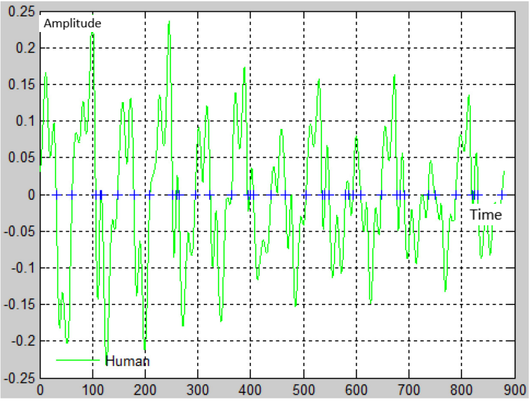


Figure 3.1 Zero-crossings of a Human voice

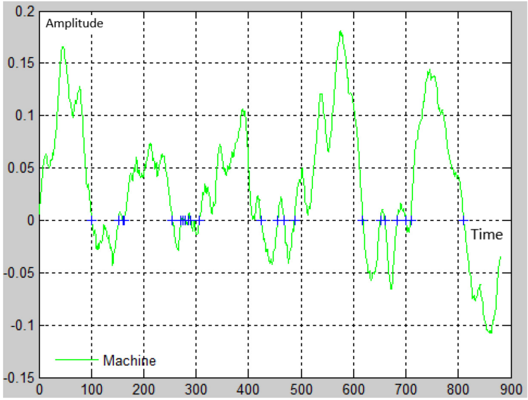


Figure 3.2 Zero-crossings of an Air-conditioning unit

The second derivative, another time-domain based feature, is a measure of the rate of change of the first derivative, which is yet a measure of the rate of change of the signal itself. It was used as a feature extraction for sound [98] and as a smoothness measure [99]. A signal possesses one global minimum – maximum and several local minima – maxima. Chesmore [69] used the number of occurrences of local minima – maxima between two zero crossings together with the number of samples between these crossings to generate Duration-Shape pairs that are further mapped through a codebook to a specific code. The method was used as a feature extraction method in sound and signal analysis for classifying animal sounds. Statistical measures are more general, and can be applied in either the time-domain or frequency-domain. The fourth moment of a signal also known as kurtosis [100], is a measure of the degree of flatness of peakedness of a probability distribution compared to the normal distribution. Kurtosis has been applied in bioacoustics for sound detection in the wild in [101]. The work in [102] used skewness in addition to other statistical measures like kurtosis to measure the statistical properties of the accumulated magnitudes generated from 18 mel-scaled channels applied in the frequency domain over a window of frames.

3.2 Spectrograms

The Discrete Fourier Transform (DFT) is used to decompose a signal into its fundamental components of sinusoids. The process involves using several cosine and sine signals of different frequencies as biases. The frequencies depend on the integer number of cycles of the biases per the number of samples of the signal under consideration. The sinusoidal biases are correlated with the function in the time domain to decompose the signal to its sinusoidal components. The magnitude of the real (cosine correlation) and imaginary (sine correlation) components results in the magnitude spectrum of the signal at different frequency bins.

The DFT has a computational complexity of $O(n^2)$. The Fast Fourier Transform (FFT) [26] was introduced as an efficient algorithm to calculate the DFT for a discrete signal, having a complexity of $O(n \log n)$. Calculating the FFT for a long signal is impractical, in terms of computation. Moreover, it assumes a signal to be stationary. Accordingly, it is even not suitable for non-stationary signals. Therefore, a modification was introduced to solve this issue in Short Time Fourier Transform (STFT) that involves splitting a signal into chunks (small enough to assume it is stationary). The FFT is applied on these short chunks of a temporal signal to generate the magnitude of the energy at each frequency bin per chunk. The concatenation of the consecutively generated STFT frames provides the change of the energy across the bins of a spectrogram as the signal progresses through time. A smoothing window [103] (e.g. Hanning, Hamming) is applied on each chunk to smooth the signal near the endings of the fragments to prevent a high-frequency response when applying the Fourier transform. Several other parameters control the resolution of a spectrogram such as the number of samples in an FFT window to calculate the DFT (usually 2^n to make use of the efficient FFT calculation, e.g. 64, 128, 256, 512, 1024).

The overlap between the successive windows of samples on which the FFT is calculated, 50% overlap is the most common overlap distance. Figure 3.3 shows a spectrogram representation. Each timestep in the temporal direction of a spectrogram represents the magnitude of the coefficients across each frequency bin for each window being analyzed. The frequency coefficients generated through the FFT has been widely used as features for signal classification [104].

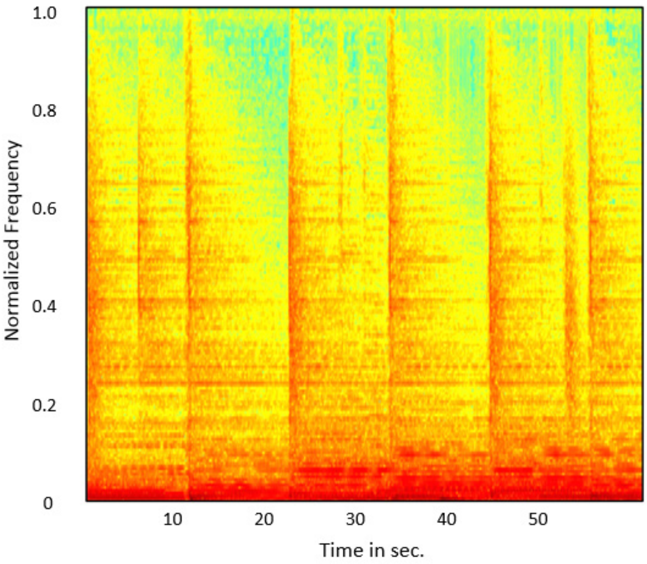


Figure 3.3 Spectrogram of road traffic

The FFT bins play a dominant role in controlling the resolution and the level of details a spectrogram can provide, especially the details required to enhance the recognition accuracy. An impairment of the spectrograms, when used in conjunction with recognition systems, is the frequency shifts, i.e. the energy of one frequency bin can move to a nearby frequency bin for signals generated from the same source. This occurs possibly due to uncontrolled circumstances affecting the signal propagation. For close signal analysis, this sensitivity is helpful, but for recognition systems, it is not. Recognition systems deal with feature vectors. Therefore, it is essential that a particular feature within a vector holds a consistent trend, i.e. a representation that holds a frequency shift-invariance property is required. Thus, instead of dealing with the frequency bins, Filterbanks [105] are used to represent the signal in bands (groupings of frequency bins). The spacing in-between the centre frequency of each of the band-pass filters of a filterbank is controlled by a specific scale. The mel-scale is widely adopted to control this spacing as it mimics the human auditory system, which responds non-linearly to the tones perceived. The human ear behaviour differs according to the sound pitches (frequency perceived by the human ear) reaching the eardrum. Stevens, Volkman and Newman (1937) studied the relation between the actual frequency and the pitches perceived by the listener, and they formulated this relationship by what is known as the "Mel-Scale" [106] (Mel from Melody) as shown in Figure 3.4. The mel-scale studies this behaviour by mapping the sound signal to the perceived tone, which is linear for frequencies less than 1 kHz and logarithmic for higher frequencies.

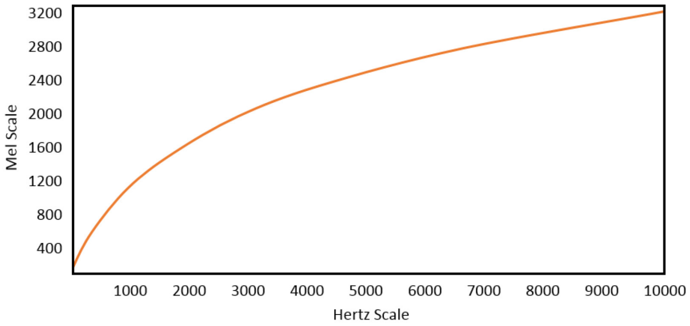


Figure 3.4 The Mel-scale

The concepts of the band and the melody scale are the basis of the Mel-Frequency Cepstral Coefficients (MFCC) and the mel-scaled spectrogram. Mel-Frequency Cepstrum is a power density (calculated from a Periodogram) of the pitches mapped to the mel-scale. The mapping process is applied through the use of a bank of mel-scaled filters as in Figure 3.5.

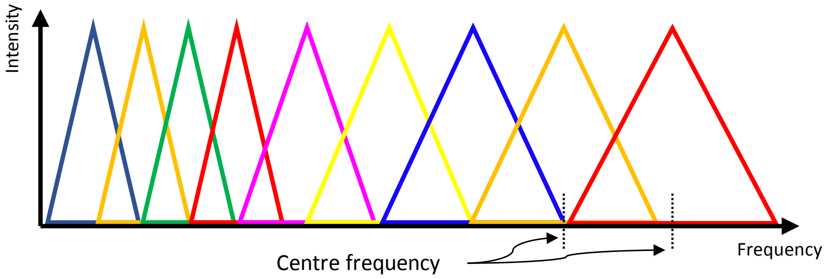


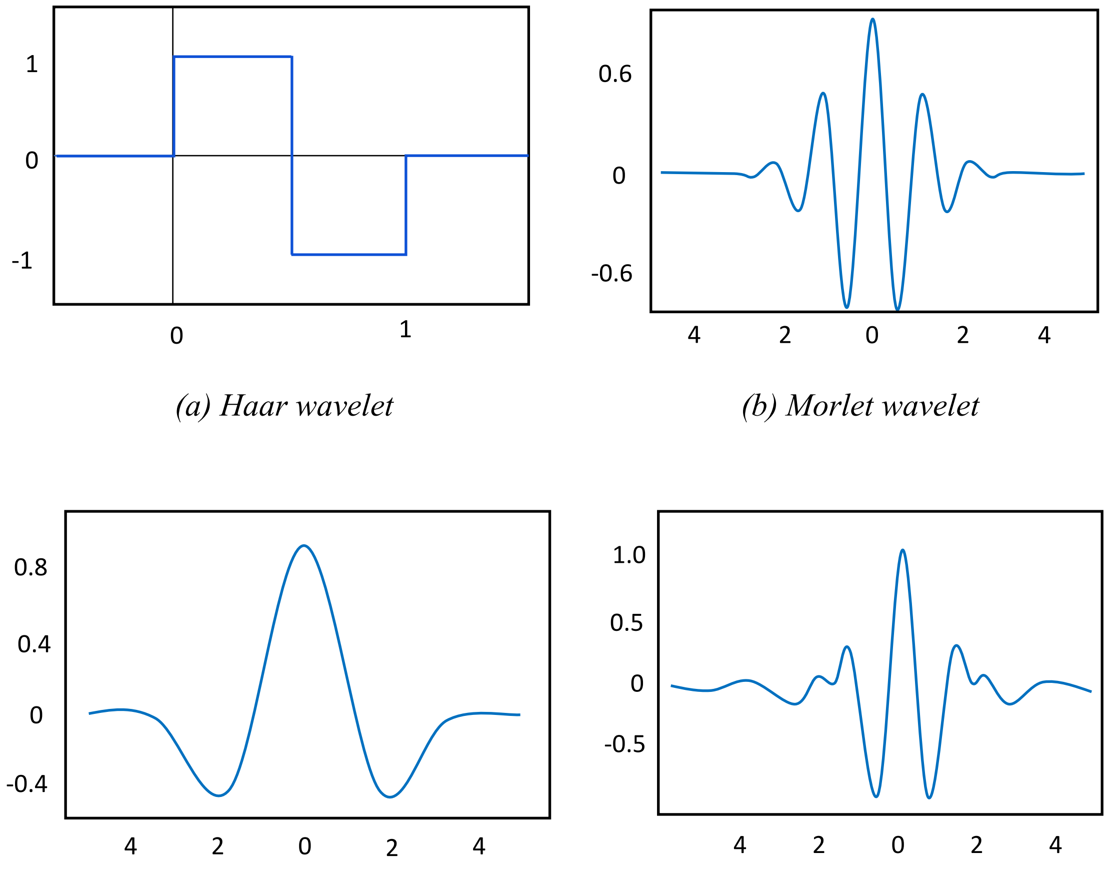
Figure 3.5 Filterbank

On applying the mel-scaled filterbank on the power density generated from a periodogram, the output is the power estimate assigned for each filter of the filterbank for each range of frequency bins. At this stage, we want to extract the values of the power that are most effective to the power density of the signal; this is where the Discrete Cosine Transform (DCT) comes into play. The DCT is very similar to the DFT. However, the DCT operates on real values only instead of complex ones, which eliminates half the computational complexity of the DFT at the expense of losing some of the high frequencies preserved in the imaginary components. This decreases the precision to an affordable extent that allows the DCT to be used in compression for images and sound; this is attributed for the ability of the DCT in extracting the most effective components and the less effective ones get to be represented with near zero values, which can be ignored. Applying DCT on the logarithmic power of the filterbank output generates the required coefficients that could be used for classification. MFCC is a widely known method for speech recognition and signal analysis [107, 108]. Several other coefficients were used in literature; Linear Predictive Cepstral Coefficients LPCC [109], Perceptual Linear Predictive Cepstral Coefficients PLPCC based on Perceptual Linear Predictive Analysis (PLP) [110], Relative Spectral PLP or RASTA-PLP [111] and Human Factor Cepstral Coefficients (HFCC) [112] to name a few. Though they may provide extra information according to the application, they are still very much similar in concept to MFCC, and better performance can be achieved combining the strength of each of them [113].

3.3 Scaleograms

Similar to transforming a signal to the frequency domain using an FFT, wavelet transformation is yet another method to examine the signal from a different point of view. The FFT transforms the signal to a number of sinusoids with different frequencies, on the other hand, wavelets perform the same role of a sinusoid but for wavelet transform. The wavelet transformation solves a clear drawback in Fourier transformation regarding the relation between the time and the frequency domain. At a certain point in the time domain, there is no possibility of specifying the exact frequency. Similarly, on approaching a signal in the frequency domain, the temporal properties are eliminated, mainly because transforming the signal to the frequency domain using the STFT assumes the signal is stationary over a window. For non-stationary signals, the window size is constant disregarding any change in the signal geometry, which causes a loss in the resolution. The uncertainty principle explains the drawback in Fourier transform [114]. The principle was studied in particle physics by Heisenberg in 1927, and it states that the direction of a particle or its speed can be determined but not both.

The concept still applies to wavelets, but with the flexibility of using a dynamic resolution based on the frequency range we are analyzing the signal for, very similar to having a dynamic window size in FFT, but for wavelets, we no longer use the term frequency, and we use the term *scale* instead. Therefore, after choosing a wavelet to be applied to the signal (samples of mother wavelets in Figure 3.6), variations of this wavelet at different scales are correlated with the signal. Low scale (high frequency) versions are used to capture the high frequencies of the signal with high resolution and the low frequencies with low resolutions. On the contrary, high scale (low frequency) versions are used to capture low frequencies with high resolution and high frequencies with low resolution. In addition to the increased resolution of both the low and high scales (analogous to the low and high frequencies), it is possible to track the temporal resolution with the knowledge of the wavelet position.



(a) Haar wavelet

(b) Morlet wavelet

(c) Mexican hat wavelet

(d) Meyer wavelet

Figure 3.6 Mother wavelets.

Despite the fact that normal wavelet transforms provided good resolution with affordable computations, there was still some data within the signal that were not captured. The Complex Wavelet Transform [115] was introduced to solve this drawback, where the phase of the signal was also taken into consideration. As an example, the Morlet wavelet, mistakenly known in the literature as the Gabor wavelet but it is not related [116] to the work of Dennis Gabor (1900 – 1979), is a widely used wavelet that is based on a Gaussian distribution modulated with a sine wave carrier as shown in Figure 3.7. A variation of it is the Complex Morlet wavelet that is capable of capturing the phase details is shown in Figure 3.8.

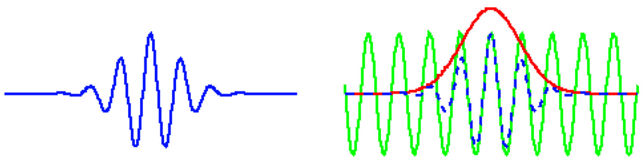


Figure 3.7 Morlet wavelet formation

(http://paos.colorado.edu/research/wavelets/wavelet2.html)

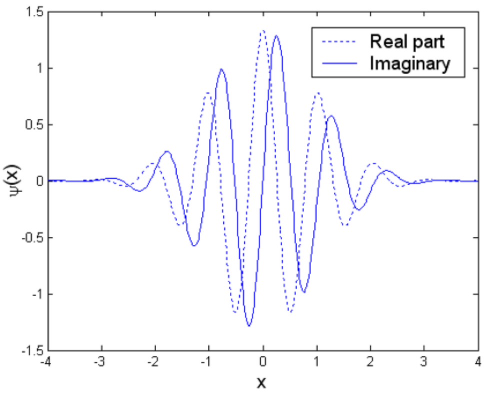


Figure 3.8 Complex Morlet Wavelet

(Adapted from [117])

A Continuous Wavelet Transform (CWT) involves scanning the signal with different scales of the mother wavelet. The Discrete Wavelet Transform (DWT) is the CWT counterpart that depends on sampling the CWT. The Fast Wavelet Transform [118] (FWT), is a method used for efficient calculation of DWT that iteratively discard half the signal required to extract the DWT coefficients. Wavelets have been used in audio recordings and music classification in [119, 120].

3.4 Pre-Processing

The features collected from the sound signals through the feature extraction stage may contain significant variations in terms of magnitude between one feature and another or even some of the features may be irrelevant to the classification process. Feeding these data directly to a classifier may degrade the performance of the classification stage. Therefore, some transformations are required as discussed in this section.

3.4.1 Rescaling

Features may have entirely different ranges of magnitudes, i.e. one feature could be a number in the interval between 1 and 4 and another could be in the range of 500 to 1000 or the mean of one feature is different from another. Accordingly, comparing the Euclidian distance will be useless since each feature has a different reference point. A feature vector having this type of variation affects the learning performance of a classifier and can increase the time complexity of the learning phase. Normalization and standardization are used to solve these issues with the data.

In normalization, it is required to fix the values to reside between a common range, generally in the interval [0 – 1] by scaling the values using (3.1).

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.1}$$

Standardization, on the other hand, is concerned with shifting the mean of the distribution to zero and controlling the variations to a unit variance as shown in (3.2).

$$x_{stand} = \frac{x - \eta}{\sigma} \tag{3.2}$$

where η is the mean and σ is the standard deviation.

3.4.2 Principal Component Analysis

Feature vectors used in classification usually have several features, where each of them is considered a dimension. Some of these features, or dimensions, may be useless or their effect on the classification process is minor. Principal Component Analysis (PCA) [121], formulated in 1901 by Pearson [122], can eliminate these features through a process called “Dimensionality Reduction”. To measure which of the dimensions is more relevant than the others, we need to calculate the degree of variation of the data across a particular dimension with respect to another. This is calculated by the covariance. The covariance is a measure of the degree of tightness or looseness of a distribution of the data points to each other. The covariance between two random variables measures the proportionality of their change with respect to each other. The covariance is the generalization of the variance, i.e. the covariance of a random variable with itself is the variance. On the basis that feature vectors usually have more than one dimension, a covariance matrix is used to hold the covariance relation of each dimension with every other dimension under consideration. In addition to the covariance of the dimension with itself across the diagonal, which is the variance.

In the context of linear transformations, a characteristic vector is a vector in space that when multiplied by a transformation matrix its length is changed but not its direction. This vector is also known as the Eigenvector, and its length is known as the eigenvalue. In other words, multiplying the square transformation matrix by the eigenvector results in another eigenvector that is multiples of the original eigenvector and these multiples are the eigenvalues. Therefore, provided that a specific transformation matrix is available of size $n \times n$, this matrix has n eigenvectors that are orthogonal to each other, and each eigenvector has a corresponding eigenvalue.

On treating the covariance matrix as the transformation matrix, the eigenvector and eigenvalue pairs are extracted. The eigenvector having the maximum eigenvalue is the principal dimension or in other words the *principal component* of the covariance matrix and consequently the data. Dimensions with small eigenvalues can be ignored, which helps in reducing the amount of data, i.e. it reduces the dimensionality. The work in [123, 124] analyzed the use of several dimensionality reduction algorithms in studying source localization of environmental sounds.

3.4.3 Linear Discriminant Analysis

PCA is only concerned with the covariance across the data, which does not consider the class of the features when projecting them on the new axis. Linear Discriminant Analysis (LDA) behaves in a very similar way to PCA, it is also used in classification besides dimensionality reduction, but it tries to find the dimension that maximizes the distance across the means of the classes under consideration. Figure 3.9 shows the difference between PCA and LDA, where PCA tries to find the dimension in the direction of the maximum variance, while LDA seeks the dimension in the direction that maximizes the distance across the mean of the distributions.

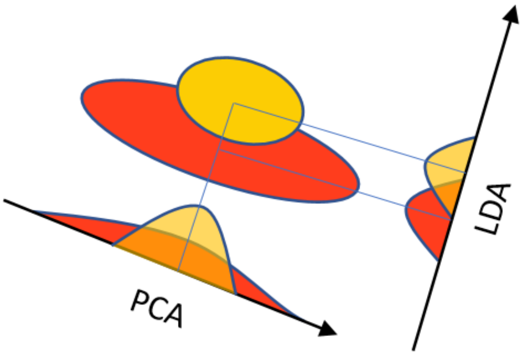


Figure 3.9 Linear Discriminant Analysis vs. Principal Component Analysis

$$J(w) = \frac{(\mu_1 - \mu_2)^2}{S_1 + S_2} \tag{3.3}$$

LDA aims at maximizing the objective function in (3.3), where µ1, µ2 are the means of features of class 1 and class 2 respectively and S1, S2 are the covariance (also known as the scatter matrix) matrices of each class. The objective function can be reformulated in the form of two matrices: the *between-class* matrix $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$ and the *within-class* matrix $S_w = S_1 + S_2$. The objective function becomes $S_w^{-1} S_b w = \lambda w$, where λ and w are the Eigen pair to be calculated attempting to find the optimal projection dimension which correspond to the maximum Eigen value. LDA has been used to enhance the performance of speech recognition systems in [125].

3.4.4 Independent Component Analysis

Independent Component Analysis (ICA) [126] is a statistical method for separating components based on the assumption that the components are non-Gaussian and linearly separable. It is an algorithm that belongs to more general set of techniques used in Blind Source Separation. Accordingly, ICA tries to minimize the Gaussian distribution of a signal to extract the independent components, even if the components are dependent, it seeks to maximize the independence. FastICA [127] was introduced in 2000 by Hyvärinen et al. to calculate ICA efficiently, and it is one of the widely used methods in the literature.

Though ICA can tackle situations where other similar algorithms like PCA fails, it still has an inherent drawback in that it requires prior knowledge of the number of components in the mixture, which is not available for the environmental sounds or music. However, a previous work [128] has used ICA for environmental sound classification.

3.4.5 Non-Negative Matrix Factorization

Belonging to Blind Source Separation similar to ICA, the Non-Negative Matrix Factorization (NMF) was introduced by Paatero & Tapper [129] in 1994 under the name "positive" matrix factorization, but was computationally intensive. In 2000, Lee & Seung introduced an efficient algorithm for NMF calculations, which revived its use in different fields. NMF theorem states that a non-negative matrix V of size $m \times n$ can be decomposed into two non-negative matrices; W of size $m \times k$ and H of size $k \times n$, where k is less than the rank (max number of linearly independent vectors which form the basis vectors) of the matrix. The basis matrix W holds the linearly independent vectors, which can be used to represent the rest of the vectors in the matrix V. The matrix H holds the coefficients used to generate the dependent vectors using the linearly independent ones in W. Therefore, multiplying W×H regenerates V. Consequently, if it is required to store V, it can be decomposed to the much smaller sized matrices, W and H, to be stored instead of V and when V is required, it can be reconstructed again from these two. Another point that links to the compression of the data storage is that NMF is sparse decomposition method, where the decomposed matrices are characterized by their sparsity, which is having mostly zeros or near zero values. Figure 3.10 shows NMF applied on images.

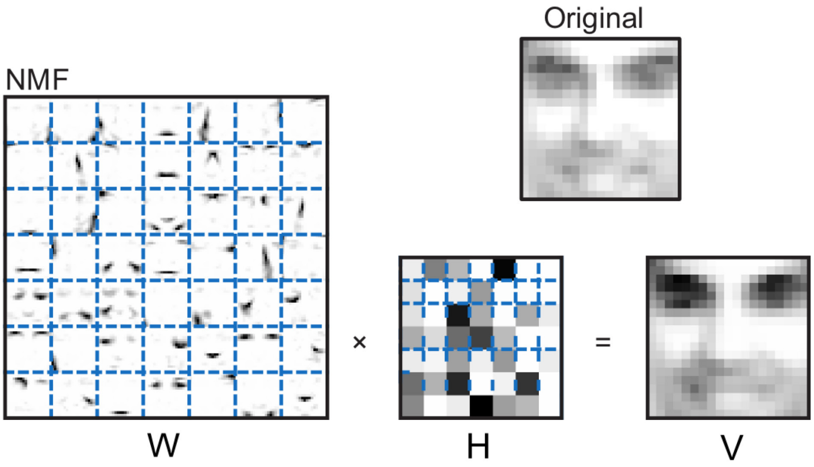


Figure 3.10 Non-Negative Matrix Factorization in image processing

(Adapted from Lee & Seung [130])

NMF is constrained by non-negative values, which made it applicable to images. It is capable of learning small parts of the images represented in the basis matrix W. Both W and H are calculated iteratively. Accordingly, no unique solution encompasses the approximations to the details of the image V.

Adopting NMF to a time domain signal is not applicable because of the negative components, but a time-frequency spectrogram fits well with the non-negative restriction. Decomposing the spectrogram to the basis matrix W and the coefficients matrix H, where W represents the features of the spectrograms and H holds the timing at which those features are appearing as shown in Figure 3.11 was investigated by Wang & Plumbley [131]. They used NMF to separate the sounds of different musical instruments within a sound mixture. Similar work was done by Virtanen in [132]

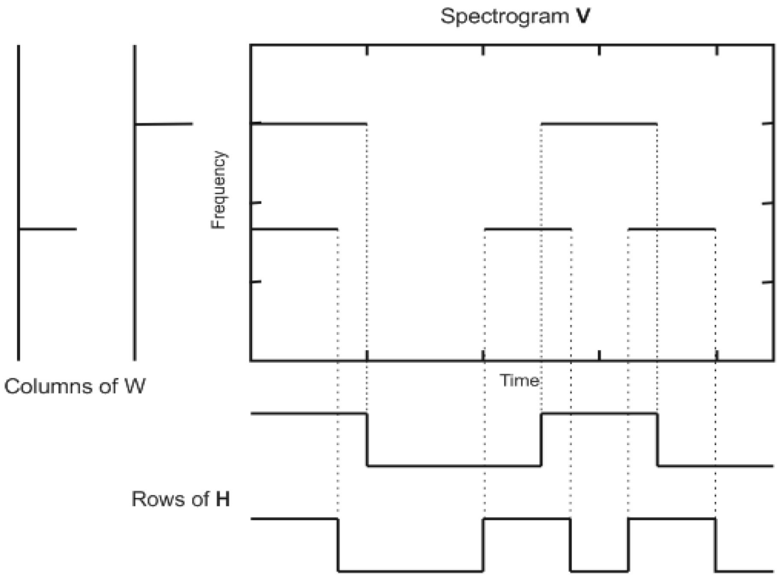


Figure 3.11 Decomposing a spectrogram into basis and weight matrices

(Redrawn based on Wang and Plumbley [131])

A practical feature in favour of NMF is the absence of a need to have prior knowledge of the number of components sharing the mixture, which is the case in environmental sound and music.

3.5 Summary

In this chapter, we reviewed the signal intermediate representations together with several pre-processing techniques applied through this work or competing attempts in sound recognition that will be discussed further in the experiments chapter. The next chapter will focus on pattern recognition models as the second part of the sound recognition problem.

4

Pattern Classification

RECOGNITION models can be broadly categorized into: *statistical*, *syntactic* and *neural networks* [133]. The training method can be supervised or unsupervised. In this chapter, we will walk through some of the widely adopted models for pattern recognition, and classification especially models adopted to temporal signals such as sound. Our discussion will avoid neural networks to which we will dedicate the next chapter.

4.1 Supervised Learning

In models undergoing supervised learning, they are trained using labelled data. The performance of supervised models generally surpasses the unsupervised ones, but one downside is that signals surrounding us are not annotated. Supervised learning requires a considerable amount of labelled training and test samples to create a well-trained model. Having such models can be hindered by the unavailability of enough labelled samples.

4.1.1 Bayes Classifier

Before moving forward with Bayes classifier [134], we will discuss Bayes Theorem (also known as Bayes rule) developed by Thomas Bayes (1701–1761). It is an important rule in the field of statistics and a fundamental one for pattern recognition systems that depend on Bayes' theory.

The joint probability of observing a certain class and a specific feature in a two-class classification problem (classes ω_1 and ω_2) is given by $P(x, \omega_j) = P(x \cap \omega_j) = P(x \mid \omega_j) P(\omega_j)$, where x is the feature, $P(x \mid \omega_j)$ is the class conditional probability of observing feature x given that the class is ω_j and $P(\omega_j)$ is the prior probability of observing class ω_j, knowing that for our two-class problem $P(\omega_1) + P(\omega_2) = 1$. The joint probability can be formulated the other way around, i.e. the probability of observing class ω_j given a value of feature x can be formulated as $P(\omega_j \mid x)P(x)$. Equating both joint distributions formulates Bayes rule in (4.1).

$$P(\omega_j \mid x) = \frac{P(x \mid \omega_j)P(\omega_j)}{P(x)} \tag{4.1}$$

which states that that posterior probability $P(\omega_j \mid x)$ is equal to the likelihood $P(x \mid \omega_j)$ × the prior probability $P(\omega_j)$ given the evidence $P(x)$, where $P(x)$ is mainly for normalization, and the other two terms are more important for the classification decision. This defines a statistical dependent classifier that can be adapted to our two-class classification problem, where if $P(\omega_1 \mid x) > P(\omega_2 \mid x)$ the decision is class ω_1 else the class is ω_2. This conditional relation can be represented in the form of a graphical representation using Bayesian networks referred to in section 4.2.2.

The problem of parameter estimation is a crucial problem for statistical methods, either supervised or unsupervised, e.g. the Gaussian Mixture Model (section 4.2.1). The problem relates to the scarcity of information, in the real world, of the prior probability distribution $P(\mathrm{A})$ and the likelihood probability $P(x \mid \omega_j)$. This problem can be approached by considering each likelihood probability as a distribution on its own with mean μ_j and a covariance matrix $\sum_j$ (discussed in section 3.4.2) that capture the relation across the likelihoods of each category. Maximum-likelihood estimation algorithms can be used for estimating the distribution parameters ($\mu_j, \sum_j$). The principle depends on dividing the sample space of the training data based on the class categories into sub-sets $D_1 \ldots D_c$, where the data in D_j follows the distribution $P(x \mid \omega_j)$ having parameters ($\mu_j, \sum_j$) represented by θ_j. Accordingly, the problem becomes a separate estimation problem for each dataset D to estimate θ that maximizes the likelihood $P\ (D \mid \theta)$, where D contains $x_1, x_2, \ldots, x_n$ samples. So the probability of observing each sample is equal to the multiplication of probabilities of observing each one of them on its own given by Likelihood probability in (4.2).

$$P(D|\theta) = \prod_{k=1}^{n} P(x_k|\theta) \tag{4.2}$$

To get the value that maximizes the *P(D|θ),* The equation can be differentiated with respect to θ and equated to zero, where the value that maximizes the vector θ is extracted. It is worth mentioning that it is easier to differentiate a sum then a multiplication. Therefore, the log-likelihood is used instead of just the likelihood.

In some situations, not all the combinations of features *x* exist especially during the training phase. In such a state, a Naïve Bayes classifier, a variation of Bayes classifier, deals with this situation by ignoring any relationship between features and it assumes independence between features of a class. The work in [135] used Bayesian inference for sound source separation and musical instrument detection.

4.1.2 Conditional Random Fields

The Hidden Markov Model (HMM), section 4.2.2, is a generative data modelling approach, where the classification problem is approached in an attempt to find the maximum likelihood of having a certain data point generated using a given class model. Conditional Random Field (CRF) [136] on the other hand, the discriminative counterpart of HMM, is used in prediction and classification problems to assign labels to a feature vector discriminatively. Linear-Chain CRF, a type of CRF, is graphically modelled very similar to HMM model and is based on the same concept, except that CRF is modelled using Markov Network referred to in section 4.2.2 that possesses undirected edges between nodes in the model.

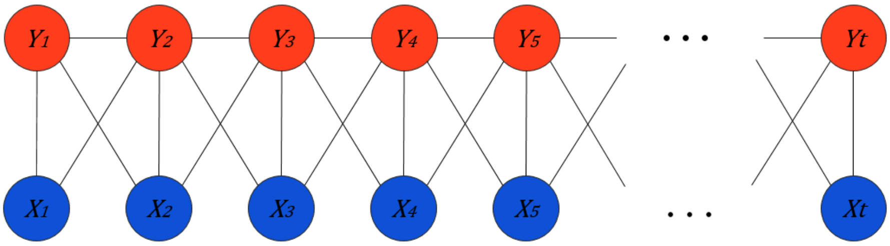


Figure 4.1 Linear-Chain Conditional Random Fields represented using Markov Network

Figure 4.1 shows a linear-chain CRF represented by a Markov Network. The Y_1,…, Y_T represents the random variables or the labels within the sequence and X_1,…, X_T represents the number of input vectors containing the features, i.e. the sequence of input vectors to be considered. The diagonal and the horizontal edges capture the influence of each node on the ones it is connected to. CRF can be considered as several neural networks next to each other, where a single neural network is influenced by the input and the output of the two, or more, networks on its sides.

CRF has been used by Yuxuan et al. [137] in analyzing the cocktail party problem, which is the main focus of the field of sound source separation, especially Computational Auditory Scene Analysis (CASA) discussed in [138]. An attempt to use CRF for the problem of speech recognition was considered by Hifny et al. in [139].

4.1.3 Support Vector Machine

Introduced by Vapnik and Lerner [140] in 1963, Support Vector Machine (SVM) was concerned with devising a linear classifier, which is not applicable to non-linear data representation. For example, two classes of data taking the shape of concentric circles. The linear SVM will fail to solve such a classification problem. The work by Bernard et al. in [141] extended the SVM to the classification problem of non-linear data boundaries. Through finding a hyperplane for higher dimensions using the dual space transformation with Lagrangian multipliers [142]. The main idea depends on using a kernel function to map all the data points to a new space and finding a hyperplane that maximizes the margin between the support vectors, then projecting the hyperplane back to the original space.

SVM has been used widely for pattern classification in general and sound classification specifically, where it is used to classify the spectrogram frames through some of the work discussed in succeeding chapters. The downsides of SVM is their inability to scale with large amounts of data due to its computation intensive requirements. In [143], the SVM efficiency regarding power consumption when implemented on a smartphone for sound event detection monitoring using different kernel functions was investigated.

4.2 Unsupervised learning

Unsupervised methods try to cluster the data into clusters without any prior information even in the absence of labelled data. A crucial input for some of these algorithms though is the number of clusters that the data needs to be split among.

4.2.1 Gaussian Mixture Model

The idea of the Gaussian Mixture Model (GMM) [27] is based on finding a combination of Gaussian distributions that can fit the data to be modelled, where we are still dealing with a clustering problem, but here we are using a generative model represented by the probability distribution of each cluster.

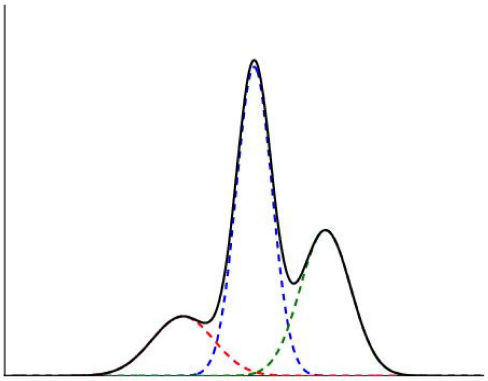


Figure 4.2 Three component GMM.

(www.robots.ox.ac.uk)

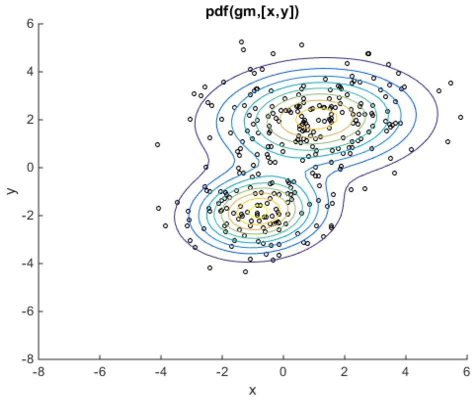


Figure 4.3 Two component GMM

(uk.mathworks.com)

Figure 4.2 shows an example of three normal distributions represented by the dashed lines and a GMM model with three components trying to fit them. The definition of a component is used in GMM to represent a class. Therefore, the three-component GMM will classify the distributions shown into three classes. Figure 4.3 shows two random variables having two distributions clustered using a two-component GMM.

Maximum Likelihood Estimation (MLE) [19] is an approach to estimate the parameters of a statistical model that increase the likelihood of the data to be generated from it, where the model has some unknown parameters (the mean and the variance of the distribution) that need to be estimated. Accordingly, the parameters that increase the likelihood of the dataset are the maximum likelihood estimates. This can be considered in other words as the learning process in GMM.

On having a clustered dataset, it is easy to calculate the mean and the variance of each distribution. On the other hand, if the dataset is unlabelled, it is only with the presence of the mean and variance of each distribution, there will be a possibility of assigning each data point to its respective distribution using the Gaussian density function in (4.3).

$$f(x) = \frac{1}{\sigma\sqrt{2\,\pi}\;\, e^{\frac{1}{2}(\frac{x-\mu}{\sigma})^2}} \tag{4.3}$$

where σ and μ are the standard deviation and the mean, respectively. But in a real-life scenario of trying to fit a GMM to the dataset, there is no prior knowledge of each distribution's parameters nor is the dataset labelled.

Expectation Maximization (EM) [19] introduced by Dempster et al. in 1977, is an iterative algorithm to solve the problem of having no prior knowledge of neither the distribution parameters nor the data assignment in order to find the best distribution to fit the data. The algorithm operates iteratively through probability instead of the Euclidian distance used in simple clustering algorithms such as k-Means. EM starts with a randomly established distribution based on a random mean and variance for each distribution. Then for all the points in the dataset, it will calculate the probability of a point belonging to each of the randomly generated distributions using Equation (4.3) and each distribution will update its mean and variance accordingly based on the distribution of the newly assigned data points. The operation continues until convergence.

GMM is one of the most widely used techniques in speech recognition [144, 145] and sound classification systems [67, 120, 146] and it has always been used in combination with the Hidden Markov Model to construct efficient speech recognizers.

4.2.2 Hidden Markov Model

Graph theory [147] has been around since it was introduced by Leonhard Euler in 1736. It is an approach of using nodes and edges to describe a mathematical relation. Graphical modelling, based on graph theory, describes the statistical relations between random variables using graphs.

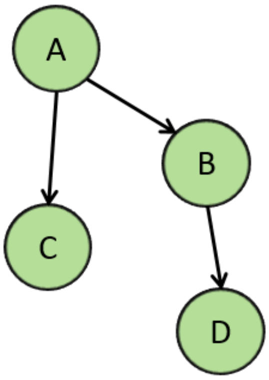


Figure 4.4 Bayesian Network

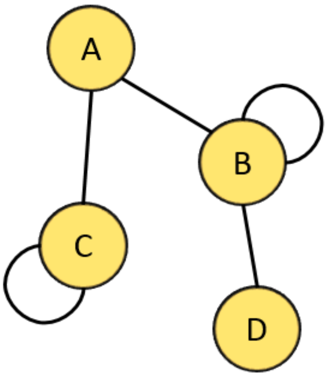


Figure 4.5 Markov Network

Two of the primary categories of graphs are Bayesian Networks [148] such as the one shown in Figure 4.4 that are characterized by having directed acyclic edges to describe the relation between the random variables (Bayesian Networks have been used in environmental sound segmentation in [81]) and Markov Networks (also known as Markov Random Fields). A Markov Network, as the one shown in Figure 4.5, is an undirected cyclic type of graph with the Markov property, which states that a future state of a random variable depends only on a specified number of previous states. For example, in a 1st order Markov property, the next state of a random variable depends only on the current state, in a 2nd order case the next state of a random variable depends on the current and the previous state.

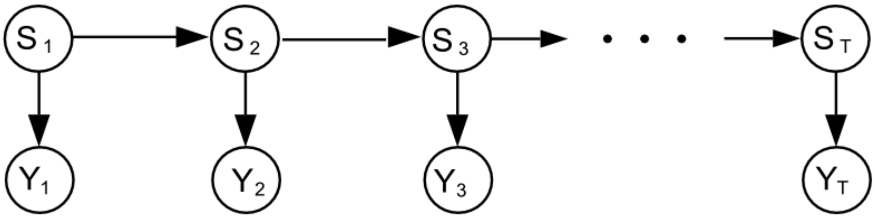


Figure 4.6 Hidden Markov Model represented as a Bayesian network

(Adapted from [148])

A Markov chain is a Bayesian network that captures the interrelation between the states of a variable across time-based on the Markov property. The HMM [22] is a Markov chain that hides the state of the random variables and it is the observations related to the state of a random variable that are available, where these observations can be used to infer the state of a random variable. Figure 4.6 shows an HMM represented using a Bayesian network. $S_1, \ldots, S_T$ are the states of a random variable across time, $Y_1, \ldots, Y_T$ represent the observations related to the random variables. The directed arrows show the dependencies among the states besides the observations and their corresponding states. The joint distribution of the states and observations is summed in (4.4).

$$P(S_{1:T}, Y_{1:T}) = P(S_1)P(Y_1 \mid S_1) \prod_{t=2}^{T} P(S_t \mid S_{t-1}) P(Y_t \mid S_t) \tag{4.4}$$

The prior distribution of the random variable that represents the initial state S_1 is $P(S_1)$, the term $P(S_t \mid S_{t-1})$ represents the transition probability between a state at time *T* and another at *T+1*. If the state space S is of size *K,* the combination of the probability of transitions between each state and the rest of the states is kept in a $K \times K$ size transition matrix A, and $P(Y_t \mid S_t)$ is the probability of an observation given a certain state. The HMM parameters are the transition matrix A and the emission probability related to the observations, which are induced from the training stage of the model.

Tracking the sequence of observations of an HMM over time can provide an insight of the most probable sequence of states the model has gone through to generate that observed sequence. Obviously, there could be several combinations of states capable of generating the same observations. The Viterbi algorithm is used in this regard, where it tries to maximize the probability of each transition within a sequence, and consequently it extracts the sequence that maximizes the probability of a certain path.

HMMs have been widely used in speech recognition systems [28, 149]. The work by Gaunard et al. in [65] used the HMM for the classification of environmental sounds in the time domain. Similar work with HMM is considered in [72].

4.3 Summary

The chapter explored examples of supervised and unsupervised classification models, aiming to provide an overview of recognition models that have been adopted for temporal signals such as sounds. These are models that exploit the temporal correlation between consecutive frames of a temporal signal. The chapter highlights conventional classifiers as well such as the Support Vector Machine, which is widely adopted as a classifier for handcrafted features or even features automatically extracted from neural networks. The chapter avoided referencing neural networks for pattern recognition, which will be handled in the next chapter.

5

Deep Neural Networks for Abstraction

NEURAL networks have been used extensively in several areas of pattern recognition. A wide range of neural network variants have been introduced throughout the literature, especially recently after a long dominance of traditional multi-layer perceptrons. The interest in introducing these models emerged from the need to automate the feature extraction stage. Feature extraction is a laborious process that involves handcrafting the optimum combination of features to enhance the accuracy of the recognition model. A recent attempt [13] has shown the success of a deep neural network architecture to extract features from raw images automatically. The attempt attracted the attention of the research community to investigate various deep architectures in nearly every possible application of pattern recognition.

In this chapter, we will dissect the structure of a neural network and walk through examples of the lately introduced neural network models. Our exploration will start with Autoencoders as a simple advancement to traditional neural networks, then Restricted Boltzmann Machines and its variant the Conditional Restricted Boltzmann Machine, which this work extends. The chapter will also highlight Convolutional Neural Networks and Recurrent Neural Networks as two state-of-the-art techniques applied to image recognition and temporal data. Finally, we will wrap up the chapter with a discussion of the referenced models and attempts to adapt them to the sound recognition problem.

It is worth mentioning that a valuable percentage of the advances the research community has achieved, is owing to Graphical Processing Units or GPUs. GPUs were initially used in the 1970s for games and in arcade systems. With the evolution of the gaming industry, the need is never ending for the most vivid and high quality rendered graphics. GPUs fulfil this need providing an intensive computing platform. In parallel, advances were being made into the field of deep learning (the name seems synonymous to deep neural networks though this is not always the case with the emergent of other types of models that still adopt the deep architecture but are not neural networks [150]). Over time, the data was expanding considerably, models were becoming larger and more complex, demanding high-performance computing. A major breakthrough was of Krizhevsky et al. [13] in training a massive CNN model on millions of images, which induced further interest. Since then, the expansion of the deep learning paradigm has been explored in a wide range of applications, and with GPUs becoming faster, they provided a platform to be adapted for machine intelligence with the immense amount of data and the complicated models with millions of free parameters to be tuned in training.

5.1 Neural Network Building Blocks

The Multi-Layer Perceptron (MLP), is the simplest form of a Feed Forward Neural Network [7]. We will use it in this section as a case study to discuss the different building blocks that make up a neural network.

The MLP, shown in Figure 5.1, is formed of interconnected nodes, where each node fires a constrained response based on the collective values of the input at this node and the constraining function implemented at the neuron level. Each connection in the figure refers to a trainable gate known as the weight. The collection of weights control how much of the input should pass to the neuron. These weights are tuned using an optimizer to minimize the error between the model's prediction and the actual label of the training sample. The constraining function is known as the transfer or activation function, which squashes the neuron's output to a mathematically plausible range. The number of layers and the number of nodes in each layer are two of the tuneable hyperparameters that depend on the nature of the data.

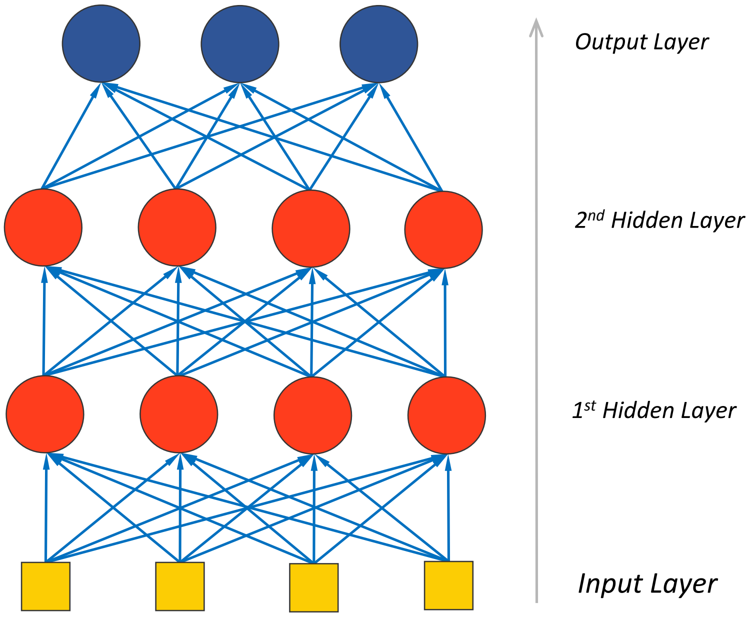


Figure 5.1 Multi-Layer Perceptron architecture

The training of the network proceeds by providing the neural network with a dataset of labelled data, where the network learns iteratively through backpropagating the errors. The error backpropagation involves updating the weights by comparing the output of the network given a labelled feature vector with the ground truth represented by the provided label. The network tries to lessen the error until convergence. At the neuron level, a transfer function receives a summation of the inputs and generate an output following (5.1).

$$y_j = f(\sum_{i=1}^{n} w_{i,j} x_i + b_j) \tag{5.1}$$

where $w_{i,j}$ is the weight between the input i and the hidden node j, x_i is the i^{th} feature of the input feature vector of length n and b_j is the bias at the hidden node. *f(...)* is the transfer function.

5.1.1 Error Function

The error (also cost or loss or objective) function is a measure of the level of deflection of the model from representing the data distribution. It is used in either a regression or a classification problem. In a classification problem, the difference between the predicted label and the target one is a measure of the performance of the model. The Mean Square Error (MSE), in (5.2), and the Cross-Entropy (CE), in (5.3), (where *y* is the target and *a* is the prediction of the network) are two of the widely adopted error functions. There is no preference of one over another as it depends on the model being trained, but generally, CE has a better convergence due to the smooth derivative of it compared to the MSE. There are models where the predictions do not depend on a logistic output, e.g. using the output for regression in the absence of a softmax function. Accordingly the values of the output are real numbers and not a probability distribution. For example, in an autoencoder structure, the input feature vector can have real values that exceed one. Accordingly, using a CE will generate an undefined value. The MSE is the candidate error function for such models.

$$MSE = \frac{(y-a)^2}{2} \tag{5.2}$$

$$CE = -\frac{1}{n}\sum_{i=1}^{n}\left[y\ln a + (1-y)\ln(1-a)\right] \tag{5.3}$$

5.1.2 Optimizer

Training the neural network involves fitting the weights of the model to the general distribution of the data. The process is initiated by predicting a label for the input and comparing it with the target label using an error function. The difference between the true and predicted value of the label is the error to be propagated back [151, 152] down the network using the optimizer. The role of the optimizer is to tune each weight in the model with a delta step towards the global minima of the error function and hopefully not getting stuck in one of the local minima. The error is propagated using the chain rule similar to (5.4).

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w}\frac{\partial w}{\partial x} \tag{5.4}$$

where for a single-layered network, the partial derivative of the error ∂E with respect to the input ∂x is given by the partial derivative of the error with respect to the activation layer output ∂a multiplied by the partial derivative of the activation ∂a with respect to the activation function input ∂z and similarly for the partial derivatives of the weights ∂w and to the input ∂x. The $\partial a/\partial z$ and the $\partial z/\partial w$ are repeated as many times as the number of layers in the network. The calculated gradient along with the weights are fed to the optimization function to minimize the loss function by updating the network's weights.

In 1847, Cauchy [153] devised the basis of what is now known as the Gradient Descent (GD). One of the noticeable revisits to Cauchy's method was in 1988 through the work of Barzilai and Borwein [154]. GD is used to minimize a function, in our case aiming to reach the global minimum of the error function. GD is an iterative method that moves in small steps towards the minimum value of the error function that consequently means the model has reached a possible optimum arrangement of weight values that maximizes the fitting of the model to the data points it is trained on. The Conjugate Gradient [155] method by Hestenes et al. in 1952 provided an accelerated convergence method compared to GD. Stochastic Gradient Descent (SGD) [156] is a variant of the GD tackling the slow operation of GD. GD moves one step towards minimizing a function after calculating the gradients across all the data points, which is not practical for large datasets. SGD, on the other hand, can apply a single update using a single data point, but practically, the parameters are updated using a random minibatch of data points. Several other optimization variants have been introduced such as AdaGrad [157], AdaDelta [158] and ADAM [159].

5.1.3 Transfer function

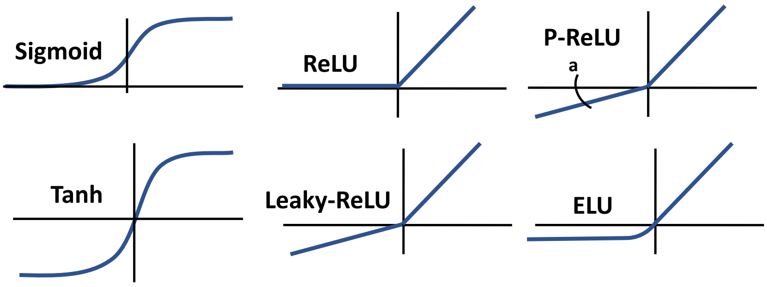


Figure 5.2 Examples of transfer functions

The S-shaped sigmoid and its cousin the tanh allow for a continuous gradient calculation, which is helpful in backpropagating the error values during optimization. Also, limiting the output to [0, 1] in the sigmoid case provides a probability value that acts as a confidence level of the activated neuron. The Tanh pretty much applies a similar non-linearity to that of the sigmoid but transforms the input to a range between [-1, 1].

The non-linear logistic family functions (Sigmoid and Tanh) were dominant until recently when new units appeared such as the Rectifier Linear Units (Relu) [160] to tackle drawbacks in such a family. The problem relates to the slow convergence of the sigmoid due to the vanishing gradient problem that causes very slow propagation of the error signal to the front layers near the beginning of the sequence of layers in the network. Several variants of the Relu appeared, e.g. Leaky Relu [161], Parametric Relu (PRelu) [162] and Exponential Linear Unit (ELU) [163] as depicted in Figure 5.2.

The softmax function is a special transfer function that is used at the output layer of the neural network for classification purposes. The softmax scales the values of the vector generated from the last layer to add up to 1, which provides a probability distribution over the classes in the output vector.

5.1.4 Regularization

Overfitting is a problem that occurs when the neural network, and generally in machine learning algorithms, learns to memorize the training data and fails to generalize to the test data because the model is overtrained. This phenomenon could occur due to the presence of a model having far more trainable free parameters compared to the amount of training data available. It could also happen if the network is trained indefinitely.

Regularization is a counter attempt to overcome the overfitting problem which simply involves suppressing the effect of some weights (by bringing them more towards 0), and consequently disabling the features associated with them, that are actually not helping in learning the general distribution of the data but rather distracting the overall generalization of the model. The L1 and L2 norms weight decay are two of the most common techniques that are added to the loss function to perform the regularization, formalized in (5.5) and (5.6), respectively.

$$L1 = \frac{\lambda}{n} \sum_{i=1}^{n} |w| \tag{5.5}$$

$$L2 = \frac{\lambda}{2n} \sum_{i=1}^{n} w_i^2 \tag{5.6}$$

where *w* is for the network weights. Dropout [164] is another strong regularization technique. Averaging the prediction of multiple networks with different weight settings enhances the overall performance. Dropout involves averaging several “thinned-networks” by disabling a fixed percentage of the activations (or input) that randomly changes from one epoch to another during training stage and enabling all the nodes during the test stage.

Batch Normalization [165], introduced recently tackles the slow training of the weights, especially when a nonlinear transfer function saturates. They also behave as a regularization technique. The method allows the normalization of the input to be applied within the layers of the model, where the normalization occurs per minibatch.

5.2 Neural Network Models

In this section, we will refer to some of the most widely used neural network models that are relevant to this work with an emphasis on the Restricted Boltzmann Machine (RBM) and the Conditional Restricted Boltzmann machine (CRBM), which the work in this thesis is based on. We will start the discussion with Autoencoders as a simple extension to MLP; then we will extend the discussion to more advanced models such as the RBM and the CRBM, and later we will highlight the CNN with the notion of weight sharing; finally, we will discuss the RNN designed for temporal signals. Most of these models were initially used for image recognition problems and later adapted to sound except, for example, RNNs that were initially introduced for sequence modelling but still not for sound.

5.2.1 Autoencoders

The Autoencoder architecture, initially studied in 1986 by Rumelhart, Hinton and Williams in [166], was an attempt to use a neural network in an unsupervised way to represent data, i.e. encode data. It is based on a neural network with one or more hidden layers in addition to the usual input and output layer trained using backpropagation. It only differs in two primary points: first, the size of the output layer must be equal to the size of the input layer. Second, the error function is not calculated against the ground truth anymore, like in a normal MLP, instead it is calculated against the input.

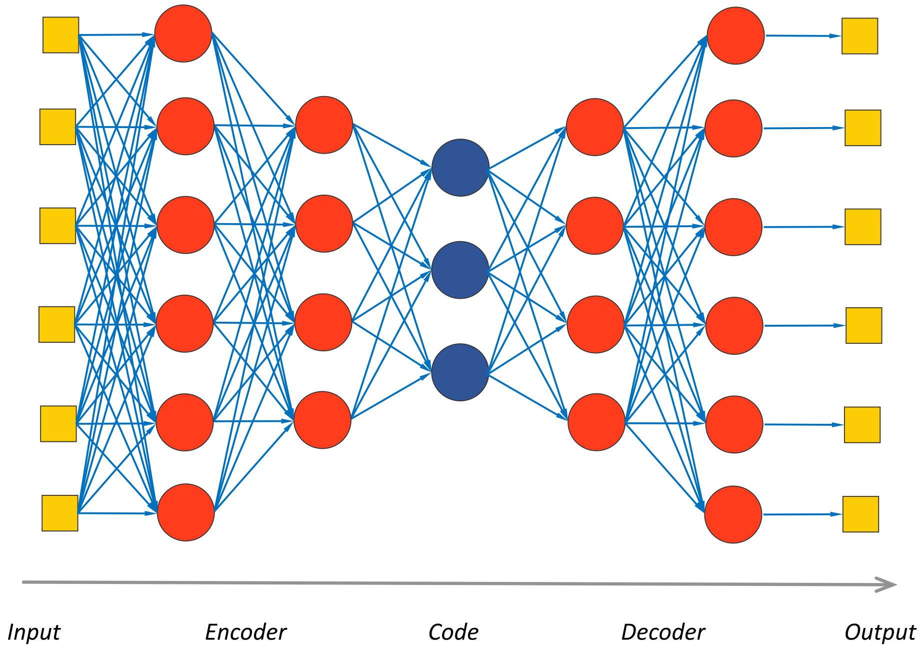


Figure 5.3 Autoencoder architecture

Figure 5.3 shows the architecture of an autoencoder, where it learns by adjusting its weight aiming to match the output of the network to the input vector at the input layer. When the hidden layer has fewer nodes, the network provides a compressed representation of the dataset. The hidden layer can then be used in accordance with a softmax layer to work as a classifier. A variant of autoencoders, used for feature extraction, was investigated through the denoising autoencoder [167]. The work in [168] used autoencoders for sound separation.

5.2.2 Restricted Boltzmann Machines

Ludwig Boltzmann (1844–1906) [169] studied mechanical systems defining their state through statistical mechanics. A field which merges probability theory with theoretical physics aiming to provide mathematical definitions to systems behaviour. Temperature and pressure are examples of macrostates that define a system on the other hand microstates go down to the particles' state defined by quantities such as the kinetic energy and velocity. The Boltzmann distribution defines the probability that a system is in a certain microstate in relation to the system's energy ε_i and temperature T following (5.7).

$$p_i = \frac{e^{-\varepsilon_i/k_B T}}{Z} \tag{5.7}$$

where k_B is Boltzmann constant and Z is a normalization constant summed over the system's possible states, Z = $\sum_j e^{-\varepsilon j/k_B T}$.

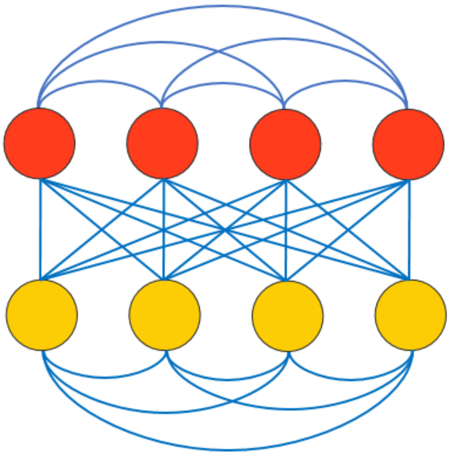


Figure 5.4 Boltzmann Machine

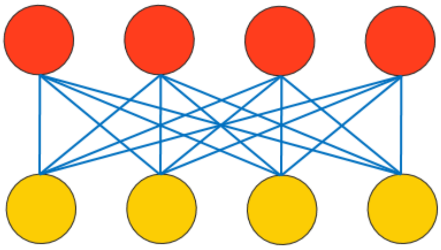


Figure 5.5 Restricted Boltzmann Machine

In 1983, Scott Fahlman, Geoffrey Hinton and Terrence Sejnowski introduced the Boltzmann Machine [170] architecture (BM) based on the Boltzmann Distribution. BM is a neural network composed of a visible and a hidden layer of neurons, where each neuron is connected to all the other neurons in the machine including neurons in the same layer as shown in Figure 5.4. Later in 1986, Smolensky [171] introduced the Restricted Boltzmann Machine (RBM), a variant of the BM. The RBM [172, 173] restricted the connections of a BM to the connections across layers as shown in Figure 5.5. In the early 2000s, Contrastive Divergence (CD) [174] was introduced as a simple method to train an RBM, which led to a breakthrough in using the RBM by Hinton et al. in 2006 [50] for dimensionality reduction. They used a stack of RBMs to form a "*deep*" architectural structure of a Deep Belief Net (DBN). The attempt is not considered a breakthrough confined only in using the neural network for dimensionality reduction, but mainly for reviving the *deep* notion of neural networks that has been around for years before 2006. This encouraged later models to use the deep architecture of neural networks beyond the single layer of neurons that was popular in recognition models earlier to the DBN. The DBN exploits the capacity of the features abstraction using the RBM as a building block, where each layer in a deep architecture extracts a higher level abstract representation of the data from the layer below it. In an unsupervised training scheme, each RBM is trained generatively and separately on the input data introduced to it. For example, training two RBMs stacked on top of each other involves training the first layer until convergence then the training of the first layer is seized and all the data samples generated from its hidden layer are used to train the second layer as if they are a new raw representation of the original input data for the second layer.

Using Contrastive Divergence [174], the training process in a single RBM involves two phases: a positive and negative cycle. In the positive cycle, the input feature vector is introduced to the visible layer, and the corresponding hidden layer activations are sampled, and in the negative cycle, an attempt in the opposite direction is to reconstruct a feature vector at the visible layer based on the sampled hidden layer activations and the weights between the hidden and the visible layer. The training process referred to with Gibbs sampling continues back and forth between the visible and the hidden layer many times for a single training sample, but typically a single iteration is used per sample, aiming to minimize the energy entrapped between the two layers defined in (5.8).

$$E(\hat{v}\,,\hat{h}) = -\sum_{i,j} W_{i,j} v_i h_j - \sum_{i} a_i v_i - \sum_{j} b_j h_j \tag{5.8}$$

where the v_i and h_j are the states of the visible and the hidden layers respectively, a_i and b_i are their biases and W_{ij} is the weight between them. The joint probability of observing a certain configuration between the hidden and visible layers is formulated in (5.9)

$$p(\hat{v}\,,\hat{h}) = \frac{e^{-E(\hat{v}\,,\hat{h})}}{Z} \tag{5.9}$$

where Z, a normalization constant, is defined as the partition function, which involves the summation of the energy of all the possible configurations of both $\hat{v}$ and $\hat{h}$. This operation is difficult to determine, but it could be estimated. Since the connections are restricted in an RBM, the distributions in either the hidden and the visible layer can be deduced, while clamping the visible layer to a training sample. Accordingly, the probability of observing an *on* neuron of the hidden layer can be given in (5.10).

$$p(\ h_j = 1 |\ \hat{v}\ ) = \sigma\left(b_j + \sum_{i=1}^{n} W_{i,j} v_i\right) \tag{5.10}$$

where σ denotes a sigmoid transfer function, b_j is the bias at the hidden neuron, v_i is the i^{th} visible input of a vector of length *n* and $W_{i,j}$ is the weight between the hidden and the visible node. Conversely, observing an *on* visible neuron is given in (5.11).

$$p(\ v_i = 1|\ \hat{h}\ ) = \sigma\left(a_i + \sum_{j=1}^{m} W_{i,j} h_j\right) \tag{5.11}$$

where a_i is the bias at the visible neuron, h_j is the j^{th} hidden activation of a hidden layer of length m and $W_{i,j}$ is the weight between the hidden and the visible node. The gradient of the weights using Contrastive Divergence is given in (5.12).

$$\Delta W_{ij} = \ \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon} \tag{5.12}$$

where $\langle v_i h_j \rangle$ is for the probability that both the visible and the hidden states are active (assigned a binary 1) together for either the input data or the reconstructed version from the negative cycle at the visible nodes.

The weights can be further finetuned with the unsupervised training of an Autoencoder [50] structure. The RBM was initially introduced with Bernoulli (binary) visible and hidden layers, and further modification adopted the use of real-valued Gaussian visible nodes with Bernoulli hidden layers, which allows an RBM to be trained on real-valued values outside the [0, 1] range. Weights pre-training was found to be one of the useful applications of the RBMs, where using an RBM for initializing the weights in an unsupervised manner provided better performance compared to randomly initialized weights. Later methods of smart weight initialization [162], [175] eliminated the need for pre-training the network. Three layers of an RBM were used in the work of Hamel et al. [51] for automating the feature extraction process in music clips for a music genre classification problem. Their work showed the accuracy achieved using the features extracted at each layer of the stacked RBM layers, where the extracted features were classified using an SVM.

5.2.3 Conditional Restricted Boltzmann Machines

The Restricted Boltzmann Machine, discussed earlier, lacks the ability to model the sequential correlation between the samples of a temporal signal. Temporal signals possess a relational property across the successional samples, which if considered can enhance the capability of a model.

Taylor et al. extended the RBM through the Conditional Restricted Boltzmann Machine (CRBM) [176] to the temporal dimension. The CRBM is similar to an RBM; a generative model trained using Contrastive Divergence [174]. They used the CRBM in modelling the human motion by tracking the movement across the joints. They also introduced variants of the CRBM like the Factored CRBM [177] for the same task. RBMs can track the observations of one frame of motion but do not have the ability to capture the correlation between successive frames. The CRBM incorporates an additional type of directed links (forming the Conditional relation) from the past samples to both the activations of an RBM at the hidden layer and the vector being reconstructed at the visible layer.

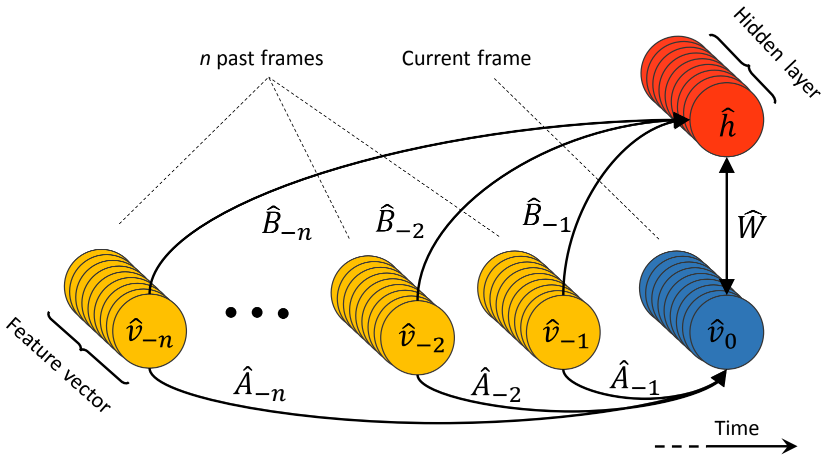


Figure 5.6 Conditional Restricted Boltzmann Machine

Figure 5.6 shows the architecture of a CRBM. The RBM's bidirectional links are depicted by the matrix $\widehat{W}$ in the figure and the conditional links from the previous time steps are depicted by the $\hat{B}$ and $\hat{A}$ tensors. The 3-dimensional tensor $\hat{B}$ preserves the influence of the past n frames on the hidden layer and the 3-dimensional tensor $\hat{A}$ holds the autoregressive relation between the past n frames and the current frame at the visible layer. The model is trained through a forward and backward pass. In the forward pass, the probability of observing a certain activation by the hidden nodes conditioned on the visible frames is given in (5.13).

$$\begin{aligned} p(\ h_{j,t} = 1 |\ \hat{v}_t, \hat{v}_{t-1}, \hat{v}_{t-2}, \dots, \hat{v}_{t-u}\ ) \\ = \sigma\left(b_j + \sum_{i=1}^{n} W_{i,j} v_{i,t} + \sum_{k=t-1}^{t-u} \sum_{i=1}^{n} B_{i,j,k}\ v_{i,k}\right) \end{aligned} \tag{5.13}$$

where σ is the sigmoid transfer function, b_j is the bias at the j^{th} node, $W_{i,j}$ is the weight between the i^{th} input and the j^{th} hidden node, and $v_{i,t}$ is the i^{th} visible element at the current temporal instance *t*. The three terms match the RBM equation in (5.10). The conditional relation from the previous visible nodes is considered in the double summation of the remaining terms $B_{i,j,k}$ and $v_{i,k}$, where the tensor *B* has *u* matrices corresponding to the *u* previous visible vectors *v*. On the other hand, the probability of observing the reconstructed pattern generated from the visible nodes given the hidden one and the auto-regressive links through the backward pass is given by (5.14).

$$\begin{aligned} p\left(v_{i,t} = 1 \mid \hat{h}_t, \hat{v}_{t-1}, \hat{v}_{t-2}, \dots, \hat{v}_{t-u}\right) \\ = N\left(a_i + \sum_{j=1}^{m} W_{i,j}\, h_{j,t} + \sum_{k=t-1}^{t-u} \sum_{j=1}^{n} A_{i,j,k}\, v_{j,k}\right) \end{aligned} \tag{5.14}$$

where the activation of the visible neuron $p(v_{i,t} = 1)$ is conditioned on the previous *n* visible states $v_{t-1}\dots v_{t-u}$ and the current hidden $\hat{h}_t$. The bias at the visible neurons is a_i, $W_{i,j}$ is the weight between the hidden neuron *j* and the i^{th} visible vector and $h_{j,t}$ is the j^{th} hidden neuron. The terms *a*, *W* and *h* overlap with the RBM original equation in (5.11). The A and *v* terms refer to the autoregressive relation between the previous visible states and the current visible one.

The Interpolating CRBM (ICRBM) [178], a CRBM extension, was used for phoneme classification in speech recognition. The ICRBM outperformed the CRBM as it considers the influence of the future frames in addition to past ones. The CRBM was also used for drum sound analysis in [179].

5.2.4 Convolutional Neural Networks

Th multi-layer perceptron has a manageable computational complexity for feature vectors of practical sizes, but for images, the number of free parameters that require training explodes in terms of count. For example, an image of a size equal to a page of this thesis (A4 page size) at printing resolution (300 dpi) has dimensions 2480 × 3508 pixels. Let us assume that we will only take a quarter of this image discarding the rest of it, which leaves us with an image of size 1240 × 1754 pixels. To feed this image to an MLP of a single hidden layer of 100 nodes, for example, we have to flatten it to a feature vector that will possess a size of 1240 × 1754 = 2,174,960 features. Accordingly, the number of free parameters in a single layer is 2,174,960 × 100 = 217,496,000. Having 217 million parameters in a single layer of a neural network does not seem practical and apparently will not scale efficiently with images of larger sizes or models having more layers. Additionally, images by nature have no restrictions on the spatial position of entities appearing in them, i.e. similar objects (or features) could reside at different locations across two different images. Accordingly, fixing the weight location is not efficient.

The Convolutional Neural Network (CNN) introduced in the work of LeCun et al. [54] in 1998, was an attempt to adapt the MLP to the nature of images. The CNN exploits the weight sharing paradigm, where weights are not tightly coupled with the spatial location of the features detected, but rather the same weights can be shared across all the neurons of a hidden layer, which decreases the number of neurons compared to a normal MLP. In this scheme, the weights behave as filters or edge detectors irrespective of the location of the features within the image, making the CNN less susceptible to orientation, variations and noise within an image.

In a CNN, a set of weight matrices also known as filters or kernels convolve the input image as in Figure 5.7. The input image may be composed of more than one channel (e.g. 3 channels for red, green and blue channels of an image). A single filter has dimensions: [length, width, depth], where the depth matches the number of channels. The convolution operation involves a weighted sum between the region of the image being scanned by the filter multiplied by the weights in the corresponding locations within the filter as shown in Figure 5.7.

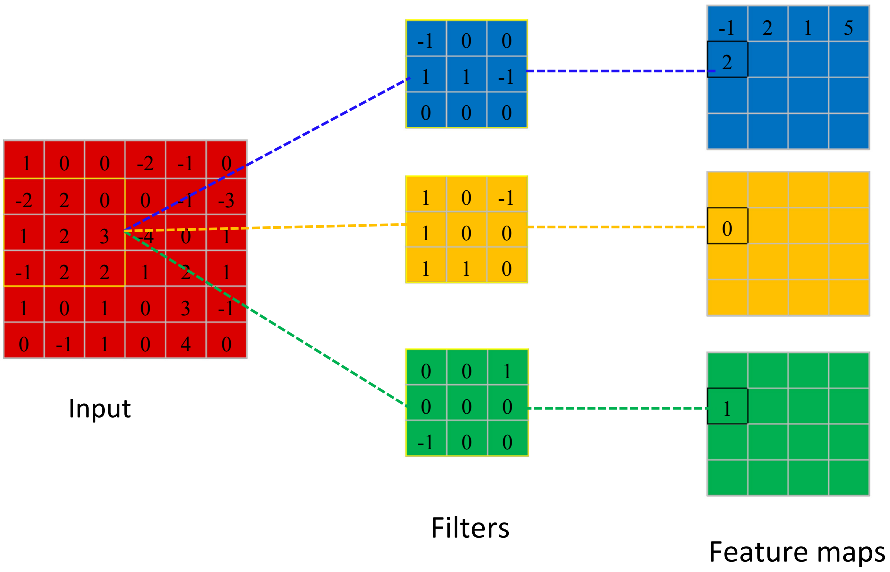


Figure 5.7 The convolution operation

Each filter generates a new representation known as feature map (a term borrowed from the image processing field) Accordingly, the number of generated feature maps matches the number of filters scanning the image. It is worth mentioning that the dimension of the generated feature map is smaller than the original input, i.e. feature map dimension [input length – filter length +1, input width – filter width +1, 1]. The output from each convolution step further goes through a linear or a non-linear activation function. The value at each location of the feature map is summarized in (5.15).

$$C_{i,j} = \sigma\left(b + \sum_{a=0}^{m-1}\sum_{b=0}^{n-1} x_{i+a,j+b}W_{a,b}\right) \tag{5.15}$$

where the neuron activation of the convolution output $C_{i,j}$ at position *i,j* is given by the transfer function σ applied over the double sum of the element-wise multiplication of each element w in the shared filter of size $m \times n$ and its corresponding element x in the input image. b is a shared bias across all the neurons of the hidden layer. A shared filter and the accompanying bias detect a specific feature e.g. horizontal or vertical edge. Sharing the same filter across all the neurons used in generating a single feature map allows the network to be translation invariant i.e. a detected prediction is not tightly coupled to the spatial location of the feature within the input since the same feature can be located elsewhere in the image. The interleaved operations of convolution and pooling are depicted in Figure 5.8.

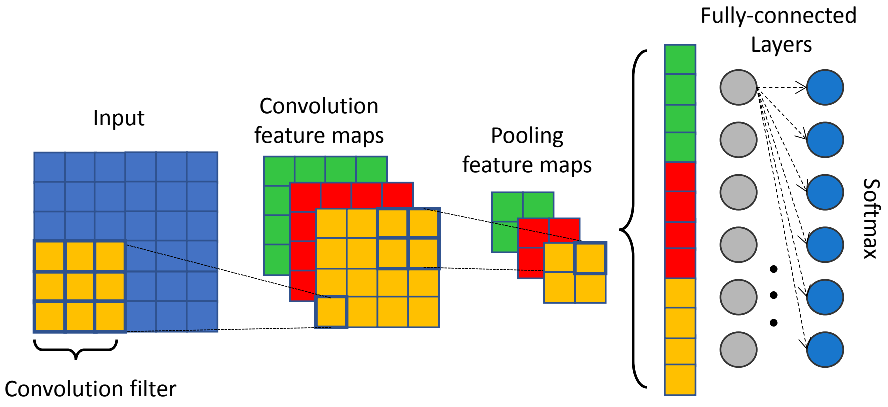


Figure 5.8 Convolutional Neural Networks

The pooling layer follows the convolutional one to generate a lower resolution feature map using a sliding window of a specified size to apply a mean or max pooling operation across the pixels in the window. For example, a window of 2×2 will subsample the a feature map to half its original dimensions in both the height and width.

CNNs are used extensively for processing images [13, 14], and several attempts exploited the convolutional operation in an effort to adapt the weight sharing property to other models. For example, Convolutional Autoencoders in [180] was an attempt to merge the autoencoder structure with a CNN. Convolutional Restricted Boltzmann Machine (ConvRBM) by Lee et al. [181] was another attempt to include the convolutional behaviour with the unsupervised training of the RBM. The ConvRBM adapted terminologies of the CNN to the RBM to allow scaling the RBM unsupervised training to images by sharing the weights and by implementing a probabilistic pooling layer. The ConvRBM is formed of a binary visible layer and groups of hidden binary units, where each group of the hidden binary units is linked with a shared weight filter across the nodes of the group in addition to a shared bias for each group. A probabilistic max-pooling layer is introduced in their work that follows the convolution layer. Lee et al. also extended applying the Convolutional RBM to the sound problem in [52]. A range of CNN variants has been applied to the sound problem will be discussed throughout the experiments.

5.2.5 Recurrent Neural Networks

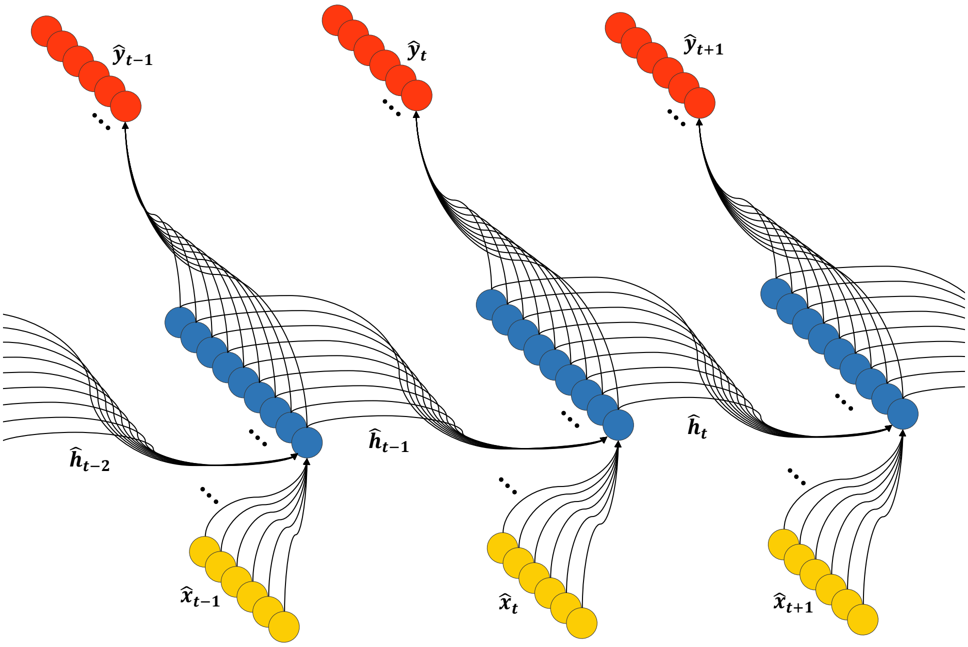


Figure 5.9 Recurrent Neural Networks

Recurrent neural networks preserve the sequence relation by involving the effect of the neuron's previous activation state in current activation through the use of a directed cycle between the output of the hidden layer and its input.

Figure 5.9 shows an unfolded recurrent neural network, for visualization purposes, where the input vector $\hat{x}_t$ of the RNN at time *t* is considered together with the previous state activations $\hat{h}_t$. The network is trained on a sequence of *n* vectors, which specifies the number of iterations the network will run, following the steps below:

for t =1 *until n*

$$\hat{h}_t \,:\, g(\widehat{W}_{hx}.\,\hat{x}_t + \widehat{W}_{hh}.\,\hat{h}_{t-1} + \hat{b}_h) \tag{5.16}$$

$$\hat{y}_t \,:\, f(\widehat{W}_{yh}.\,\hat{h}_t + \hat{b}_y) \tag{5.17}$$

where the $\widehat{W}_{hx}$ is the weight matrix between the input and the hidden layer, $\widehat{W}_{hh}$ is the weight matrix between the previous hidden state vector and the hidden layer, and $\widehat{W}_{yh}$is the weight between the hidden layer and the output. $\hat{x}_t$ is the input vector at index *t* of the input sequence of vectors, $\hat{h}$ is the hidden state vector and $\hat{y}_t$ is the corresponding output. *g* and *f* are the transfer functions used to add the non-linear transformation to the generated vectors, where *g* is usually a *tanh* function and *f* is a softmax. It can be inferred from the above equation that the folded form of an RNN in Figure 5.9 is a normal MLP with a feedback loop from the hidden layer's output back to its input.

An RNN is trained using Back Propagation Through Time (BPTT) [8, 182], which behaves similar to normal Back Propagation [151] using the chain rule, but for RNN it is back-propagating the error from each generated sequence to the sequence before it. Theoretically, a RNN can process long sequences, e.g. 1000-long sequence, which resemble a very deep network architecture having a number of layers matching the sequence count that are recurrently fed through the network. This introduces the vanishing and exploding gradient problem that was studied by Hochreiter in [183]. Both problems occur due to the multiplication of several large value gradients (exploding) or small ones (vanishing). In the vanishing gradient problem, small gradient values of the weights are multiplied recurrently using the chain rule. The multiplication between such small values results in even smaller gradients, which makes the network less sensitive to inputs that are further down the sequence as it diminishes the error signal propagated and consequently prevents the RNN from learning long-term dependencies in addition to slowing down the learning speed. The vanishing gradient problem has also been noticed in other deep multilayer neural network models, e.g. CNN, when they use the sigmoid as discussed earlier in the transfer function section.

Long Short-Term Memory (LSTM) was introduced in 1997 by Hochreiter and Schmidhuber [9] to address the RNN problems, especially the exploding and vanishing gradient, using memory to preserve the hidden state of a RNN. The LSTM memory module replaces the conventional hidden layer of neurons used in the RNN, behaving as a computer memory with read, write and reset controllers or "gates". Graves et al. [11] used a deep LSTM RNN architecture for phoneme recognition in speech. An attempt to use the LSTM to exploit the long-term dependencies across the frames in combination with the features extracted by a CNN was investigated by the Convolutional RNN (CRNN) in the work of Choi et al. [184] for music classification. Other LSTM variants appeared such as the Gated RNN [185], and different LSTM architectures were proposed in the Bidirectional LSTM [186] and the Multidimensional LSTM [12].

5.3 Sound Recognition with Neural Networks

The neural network models explained earlier, broadly categorized into convolutional, recurrent and multilayer perceptron neural networks, are some of the widely used models that have been adapted to a range of applications including sound. For example, after the DBN success in digit recognition in 2006 through [50], it was considered as a feature extraction method for music by Hamel et al. [51] in 2010. In their work, they explored the use of a DBN in extracting features from spectrograms, aiming to avoid the need to hand-craft the required features for classification. They trained three RBM layers on music spectrograms, where the extracted features were further classified using an SVM. The frame level classification accuracy was 77.0% using their three RBM layers applied on 513 frequency bins generated from a DFT of music files. The performance of their extracted features surpassed that of the MFCC, which achieved an accuracy of 63%. Lee et al. [187] applied a similar attempt for environmental sounds. They compared a PCA-whitened logarithmically mel-scaled 128 bins spectrogram processed with an RBM to extract features that are classified using an SVM, to a 13 bin MFCC classified using a GMM. They achieved an accuracy 72% compared to the MFCC counterpart achieving 65.5% using the Bag-of-Frames (BoF) modelling. The BoF method was adopted from text retrieval, where a text document is modelled using a Bag-of-Word vector representing the words in a text document and their corresponding frequency of occurrence irrespective of the syntax structure. A similar analogy was adapted to sound [44, 51] representation, where the long-term statistical distribution of a signal is represented by the distribution of its short-term features represented in the frames of the STFT, which are usually modelled using a GMM. This incur treating each frame as an isolated entity while ignoring the neighboring frames correlation, which embodies the context of a frame in the temporal progression of a signal.

The multilayer perceptron and their advanced cousins of DBNs and similarly the BoF methods ignore the neighboring frames temporal correlation, which if considered would enhance the performance of the recognition. This encouraged other models that can preserve the temporal correlation to be adopted for the problem. Weight sharing is one of the principal advantages of using CNN and convolutional based architectures, as it eliminates the need to have a corresponding connection between each pixel in the input image and the hidden layer nodes. This is motivated by the inherent property of images that a feature detected in a certain region in the image has a high probability of being detected elsewhere. This allows the network to be translation-invariant to the different variations and orientation of an object in an image. A CNN seems like a prominent candidate model with its convolutional operation. The process is efficient and effective for images, but not for spectrogram representations. Contrary to images, the two dimensions of a spectrogram have completely different meanings. Moreover, the amplitude of a frequency bin at a certain temporal instance is composed of the sum of energies generated from overlapping sound events. And whereas objects in images tend to be spatially contiguous, the energies of frequencies of a sound event in a spectrogram are distributed about the spectrum. The fundamental frequency, harmonics and overtone frequencies of a sound event will reside at different spatial locations across the frequency bins of a spectrogram, yet all of them contribute to the energy of the same source. These considerations pose the possibility that CNNs are not the optimum solution for spectrogram recognition, which induced several attempts to tailor them to the problem. In 2014, the work by Abdel-Hamid et al. [86] used a CNN for speech recognition. They proposed what they refer to in their work as a CNN with "*limited weight sharing*", which involves using different sets of filters for different bands. Pons et al. [56] proposed different CNN architectures tackling music-related properties using different set of single-dimension filters to scan the temporal and the frequency axis of a spectrogram separately, where they achieved an outperformance using these pretrained sets of weights combined into the same model compared to using square-shaped filters. Kereliuk et al. explored a comparable attempt in [15]. Another tailored deep architecture that considered a set of filters dedicated to music and another set for speech within the same model with a merging stage for the features extracted from both types of filters was in the work by Barros et al. [18]. Wyse [188] investigated using a CNN channel for each frequency bin in a spectrogram. It is evident from these attempts [18, 56, 86, 189] that weight sharing across the frequency and time axes, is not optimum for preserving the location (the specific coordinates of a feature in the two dimensional representation) of the energy of the frequency bins. Especially if the filters are learning about features in any location in the image (spectrogram in the case of sound) rather than features in a specific location. The position of a frequency bin is crucial for time-frequency representations since the location of the learned features across the frequency bins complemented by the modality of the energy through time act as distinctive properties between sounds.

Recurrent Neural Networks are plausible for temporal signals due to the nature of their design. Long Short-Term Memory solved the inability of the standard RNN to extend for long sequences due to the vanishing gradient problem. Graves et al. [11], in 2013, used a deep LSTM RNN architecture for phoneme recognition in speech. An attempt to use the LSTM to exploit the long-term dependencies across the frames in combination with the features extracted by a CNN was investigated the Convolutional RNN (CRNN) in the work of Choi et al. [184] for music classification. Despite the LSTM success in handwritten text recognition and speech, the model and its variants such as Gated Recurrent Neural Networks use memory modules, which complicate the model and consequently the training.

Most of the referenced attempts, in this chapter, are state-of-the-art methods that have achieved wide success in a range of applications. They were adopted to the sound problem after they had gained such success. However, it is clear from the above discussion that these methods may not be optimized to harness the time-frequency representation of a sound signal.

5.4 Summary

Handcrafting an optimized set of features from a raw signal is a time-consuming process. Neural Networks are currently being considered as a method to automate the feature extraction stage. Through this chapter, we explored the internals of neural networks and referenced examples of popular neural network architectures used for temporal signals that are relevant to this work. The next chapter will explain the main contribution of this thesis, and it will discuss the relevant connection between some of the models in this chapter and their limitations when it comes to recognizing sound.

6

Masked Conditional Neural Networks

THE temporal correlation among the consecutive frames of a temporal signal is influential for sound recognition, and several models have been proposed to exploit this nature [190]. In addition to the temporal relation between frames, the energy of a frequency bin at a specific spatial location within a spectrogram is distinctive to the sound category. Accordingly, the frequency and temporal locality of the features detected is crucial and can significantly affect the performance of a model.

Most of the neural network models used for the sound problem are adapted after they gain wide acceptance in other domains especially image recognition. This is evident through some of the widely used neural network architectures, discussed in the previous chapter, attempting to fit these models to the nature of the sound signal, which may not optimally harness time-frequency representations.

In this chapter, we introduce the Conditional Neural Network (CLNN) that is designed for the nature of the temporal signal. Most importantly, the Conditional Neural Network preserves the frequency and temporal locality of the learned features and act as the main skeleton for the Masked Conditional Neural Networks (MCLNN). The Masked Conditional Neural Network exploits properties of the filterbank used in spectral transformation by enforcing a systematic sparseness that follows a band-like pattern over the network's connections. The models we introduce in this thesis are designed for multichannel (a channel represent a single feature across time) temporal signal representations. The models consider the temporal succession of frames and the positions of frequency bins within spectrograms. Meanwhile preserving the generalization to be adapted for any multichannel temporal signal.

For notation purposes:

- Uppercase symbols with the hat operator are used for matrices, e.g. ($\widehat{W}$) and with a subscript refers to a matrix in a 3D tensor at an index, *e.g.* ($\widehat{W_u}$).
- Lowercase symbols with the hat operator are used for vectors, e.g. ($\hat{x}$).
- The absence of the hat operator refers to a single element within the matrix or a vector i.e. $W_{i,j}$ is the element of a matrix at location [i, j] and x_i is the i^{th} element of the vector $\hat{x}$.
- The dot operator ($\cdot$) is used for vector-matrix multiplication.
- Element-wise multiplication between two vectors or two matrices of the same dimensions uses ($\circ$).
- The absence of any operators or the use of a multiplication symbol (×) refers to normal element multiplication, i.e. ($x_i\ W_{i,j}$) or ($l \times e$).

6.1 Conditional Neural Networks

A sound signal possesses a temporal correlation between its consecutive frames. Accordingly, extending the network structure to embed a windowing behaviour enhances the model's decision, where a window of frames rather than a single feature vector, as in bag-of-frames classification, is projected in the prediction of the network. The Conditional Neural Networks (CLNN), we introduce in this work, similar to other temporal models, observes a window of frames. The CLNN implements this behaviour by including conditional links that span a window. The CLNN is a discriminative model that extends from the generative Conditional Restricted Boltzmann Machine, discussed in the previous chapter. The main overlap is the adaption of the conditional links from the previous temporal inputs to the hidden layer. The conditional links have been extended in the Interpolating Conditional Restricted Boltzmann Machine [178] to the future frames in addition to the past ones, which we adopt as well for the CLNN.

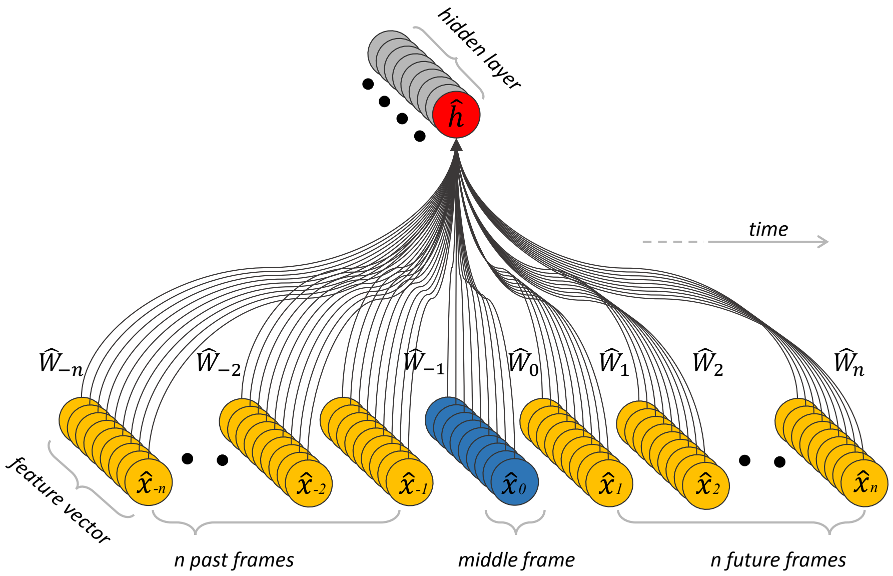


Figure 6.1 ConditionaL Neural Network layer

Figure 6.1 depicts the connections of a single neuron of a CLNN layer. The figure shows a number of feature vectors ($\hat{x}_{-n}, \dots, x_{-2}, x_{-1}, x_0, \hat{x}_1, \hat{x}_2, \dots, \hat{x}_n$) representing the window of frames for the CLNN to process. Each feature vector is fully-connected with the hidden layer through a dedicated weight matrix $\widehat{W}_u$, where *u* is the index of the matrix in the weight tensor having *d* weight matrices of indices in the interval [-*n, n*]. The input to a hidden layer is a set of vectors each of *l* features of count *d* following (6.1).

$$d = 2n + 1 \quad , n \geq 1 \tag{6.1}$$

where the window of *d* frames has a width that depends on the order *n* in addition to the window's middle frame. The order specifies the number of frames in a single temporal direction. Twice the order is used to account for both future, and past frames. Accordingly, the predicted activations, at the single-dimensional hidden layer of *e* neurons, are conditioned on the window's central frame in addition to the *n* frames on either of its sides.

The output of a CLNN step has 2*n* fewer frames than its input, where the window's central frame is summed with the 2*n* off-centre frames. In order to account for the consumed frames in a deep CLNN architecture, segments of the spectrogram are extracted each of size [*l, q*], where *l* is the feature vector length (number of frequency bins), and *q* follows (6.2).

$$q = (2n)m + k \quad , n, m \text{ } and \text{ } k \geq 1 \tag{6.2}$$

where the width of the segment q is dependent on the order n (2 for the past and future frames), the number of layers m and the extra frames k. These extra frames, remaining beyond the CLNN layers, can be either flattened to a single feature vector or pooled across using mean or max pooling as discussed in [191] for images, but for time-frequency representations, it will be a single dimension temporal pooling through the features. The relative sizes between the CLNN processing window d, the segment q and the actual spectrogram is depicted in Figure 6.2.

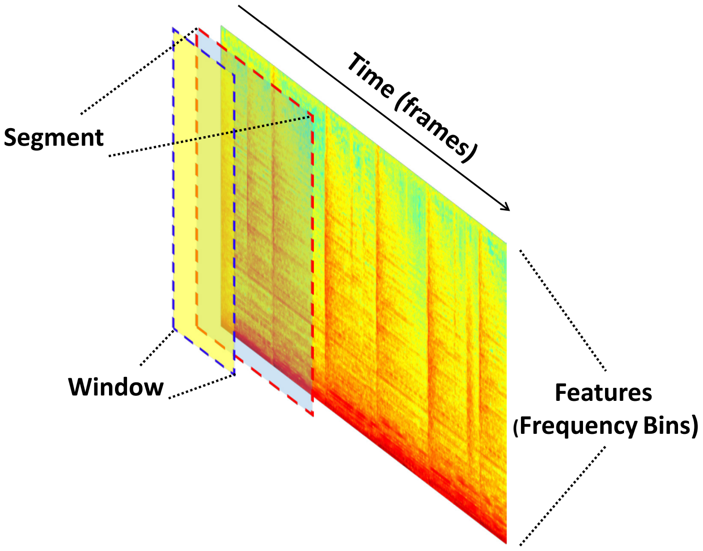


Figure 6.2 The relative size of the window d compared to the segment q and the spectrogram.

The output of a single neuron of the hidden layer is formulated in (6.3).

$$y_{j,t} = f\left(b_j + \sum_{u=-n}^{n} \sum_{i=1}^{l} x_{i,u+t} \, W_{i,j,u}\right) \tag{6.3}$$

where $y_{j,t}$ is the output of neuron j of the hidden layer. The index t is for the position of the frame within the segment, which is also the middle frame of the window. The activation function at the neuron is f and the bias is b_j. The term $x_{i,u+t}$ is for feature i of the vector $\hat{x}_{u+t}$, where each element in the vector of length l is multiplied by its corresponding weight element $W_{i,j,u}$, where the indices i *and* j refer to the connection between the i^{th} feature in the feature vector and j^{th} hidden node. The index u in both $x_{i,u+t}$ and $W_{i,j,u}$ refers to the index within a window of width d.

The output at the hidden layer formulated in a vector form is given in (6.4).

$$\hat{y}_t = f\left(\hat{b} + \sum_{u=-n}^{n} \hat{x}_{u+t} \cdot \widehat{W}_u\right) \tag{6.4}$$

where $\hat{y}_t$ is the activation vector observed at the output of a CLNN conditioned on the window's middle frame $\hat{x}_t$, and the $[\hat{x}_{-n+t}, \ldots \hat{x}_{-1+t}]$ and $[\hat{x}_{1+t}, \ldots \hat{x}_{n+t}]$ neighbouring frames. The transfer function at the neuron is *f* and the bias vector at the hidden layer is $\hat{b}$. The vector at index *u* in a window is $\hat{x}_{u+t}$. The index *t* is for the window's middle frame, which matches the index of the frame in the segment. The weight matrix at index *u* within the weight tensor is $\widehat{W}_u$ (the tensor have *d* matrices) of size [*l, e*], where *l* is the length of the feature vector and *e* is the hidden layer width. For each index *u,* a vector-matrix multiplication is applied between the frame at index *u* within the window *d* and its corresponding weight matrix at the same index. The vector-matrix multiplication generates *d* frames each of *e*-dimensions. The resulting vectors are summed feature-wise across the temporal direction to generate a single vector that undergoes a non-linear transformation using the transfer function at the hidden layer. The conditional distribution of the inferred activation vector at the hidden layer conditioned on the middle frame of the window and the the 2*n* neighboring frames can be captured in $p(\hat{y}_t | \hat{x}_{-n+t}, \ldots \hat{x}_{-1+t}, \hat{x}_t, \hat{x}_{1+t}, \ldots \hat{x}_{n+t}) = \sigma(\ldots)$, where σ is the sigmoid activation or the final softmax output. Figure 6.3 depicts the *d* weight matrices of a CLNN scanning a segment extracted from a spectrogram.

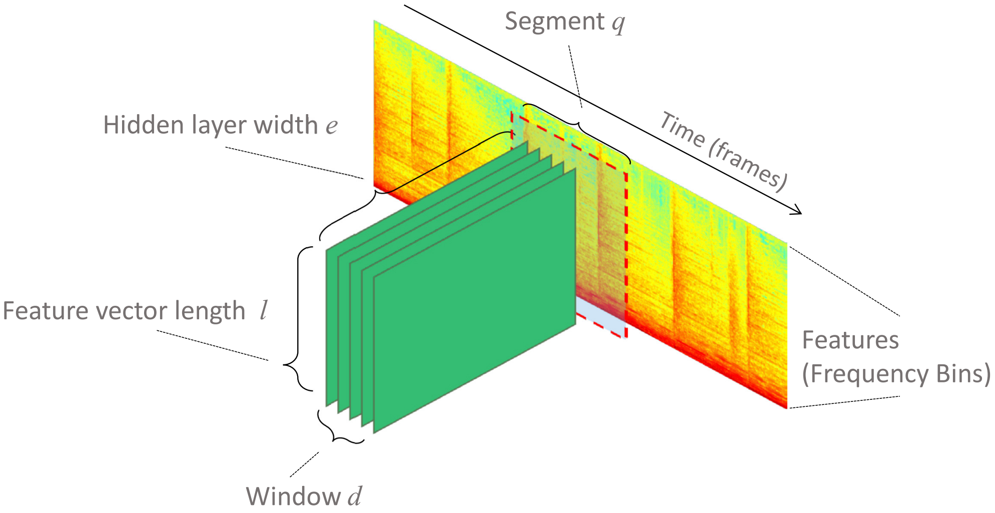


Figure 6.3 The CLNN scanning a segment extracted from a spectrogram.

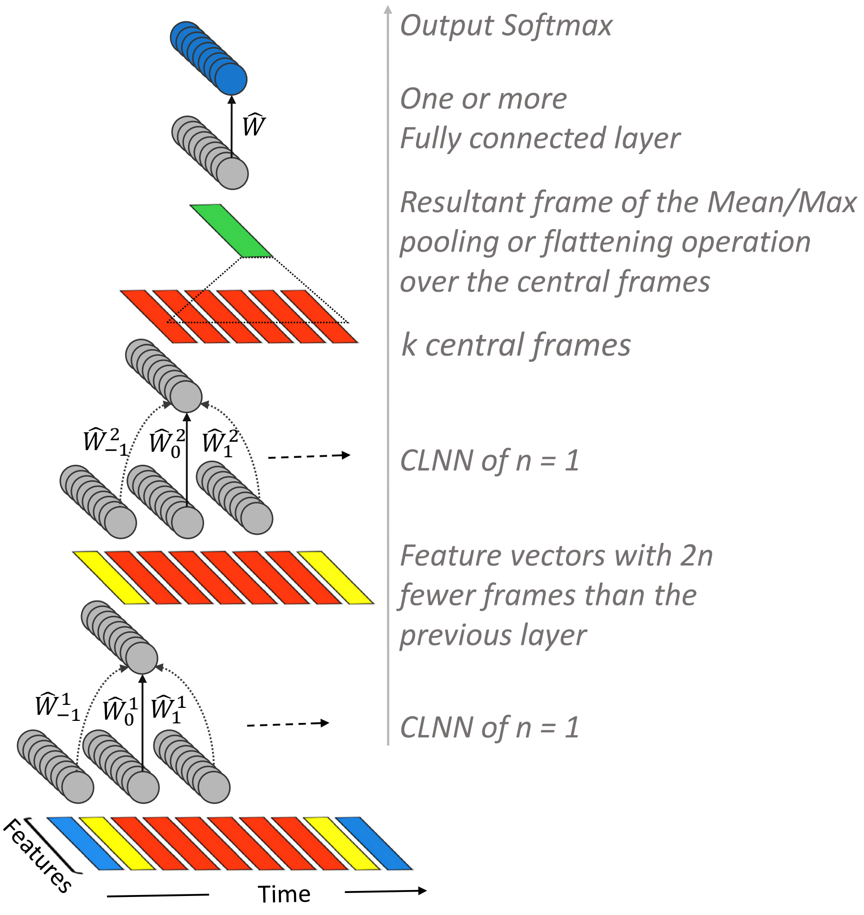


Figure 6.4 A two-layer CLNN model with $n = 1$

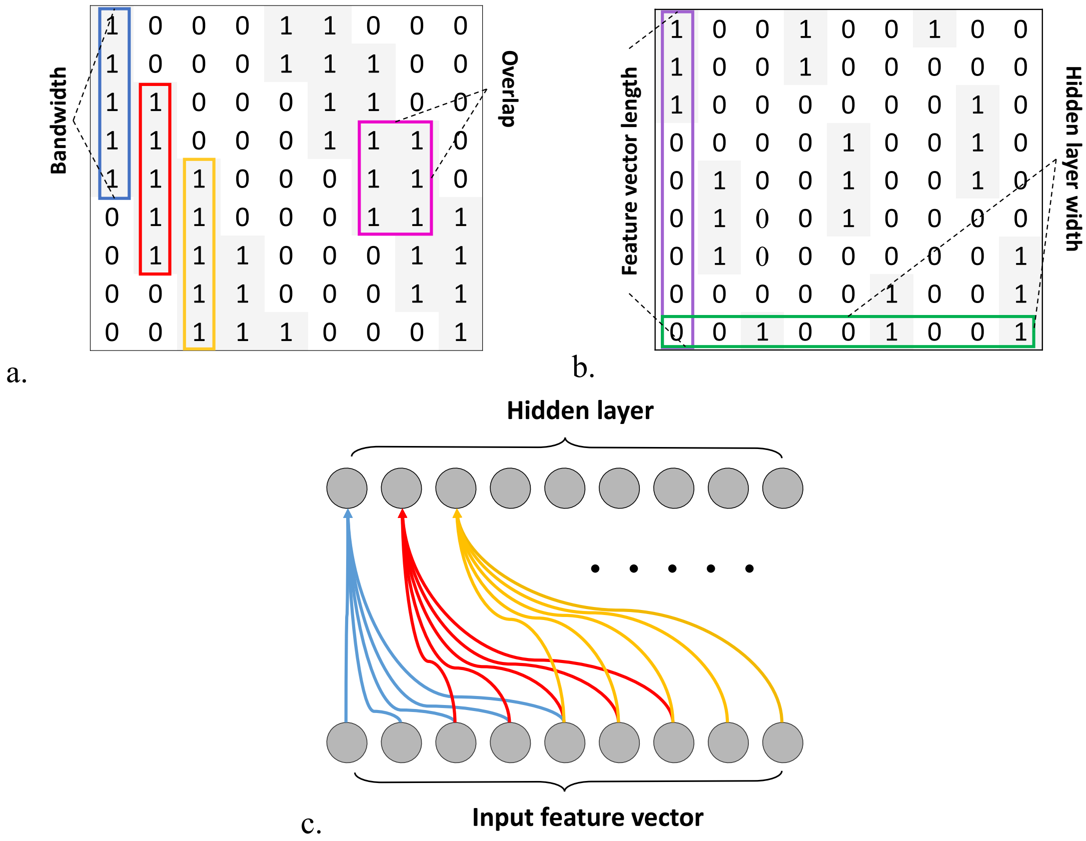
Figure 6.4 shows the architecture of a two-layered CLNN ($m = 2$) with an order $n = 1$. Each layer holds a 3-dimensional weight tensor $\widehat{W}^b$, where $b = 1, 2, \ldots, m$. For an order $n = 1$, the depth of the weight tensor = 3. Therefore, for each of frames within a window (3 frames at n=1) there is a dedicated weight matrix having the same index to process it through a vector-matrix multiplication. Accordingly, the weight matrix $\widehat{W}^b_0$ is for the window's middle frame at $u = 0$, $\widehat{W}^b_{-1}$ and $\widehat{W}^b_1$ for the off-center frames at $u = -1$ and $u = 1$, respectively. Similarly, for $n = 2$, the weight tensor is composed of five weight matrices. As shown in the figure, the first CLNN layer feeds $q - 2n$ frames to the second CLNN layer, which in turn performs in a similar manner to generate another representation for succeeding layers. The final output for these two layers scheme is one or more (based on the k extra frames) representative frames at the output of the second CLNN, which can be flattened or pooled across then fed to a densely connected network before the final softmax layer for classification. For example, at $n = 6$, $m = 3$ and $k = 10$, the input at the first layer is (2 × 6) × 3 + 10 = 46 frames. The output of the first layer is 46 – 2n = 46 – (2 × 6) = 34 frames. Similarly, the output of the second and the third layers is 22 and 10, respectively. The 10 frames at the output of the third layer represent the k extra frames that can be flattened or globally pooled feature-wise to create a single output vector per input segment to be used for classification. The temporal pooling behaves as aggregation over a texture window, which was studied in [38] for music. The extracted segments from the spectrogram can overlap with a maximum of q – 1 frames and a minimum of zero.

6.2 Masked Conditional Neural Networks

This section elaborates on how the Masked ConditionaL Neural Network extends upon the structure of the CLNN to account for the spectrogram frequency band properties.

The frequency components of a time-frequency representation at a temporal instance can be combined using filterbanks [192], discussed earlier in the signal representation chapter. Filterbanks are formed of a group of bandpass filters each suppressing a range of frequencies while allowing others. Filterbanks may have different shapes to provide different scaling factors over the frequencies under consideration. They provide a weighted sum to aggregate the energies across the frequencies residing within the bandwidth of each bandpass filter. A filterbank is designed based on the number of filters and their shape together with both the center frequency and bandwidth of each, which consequently affects the overlapping distance between the filters.

The MCLNN mimics a filterbank-like behaviour through a systematic sparseness enforced over the connections between the input and the hidden layer within the network through a binary mask. The mask follows the structural pattern of the frequency bands in a spectrogram as shown in Figure 6.5. The figure depicts two examples of a binary mask, where the columns match the number of hidden nodes and the rows are equal to the number of features of the input. The mask design is controlled through two tunable hyper-parameters namely: the Bandwidth and the Overlap. The Bandwidth controls the number of features to be considered in the same band (similar to a filter in a filterbank), and the Overlap controls the superposition distance between successive bands (mimicking the overlap between filters). For example, Figure 6.5.a. shows an example of a binary masking pattern of a Bandwidth = 5, this is represented by the consecutive ones positioned in a single column. The same masking pattern has an Overlap = 3; this is represented by the superposition of the binary patterns across the consecutive columns. The Overlap can be assigned negative values that refer to the non-overlapping distance across the columns as shown in Figure 6.5.b. The sparseness enforced by the mask enables certain regions of the weight matrix and disable others as depicted in Figure 6.5.c. showing the active connections following the mask in Figure 6.5.a.



1	0	0	0	1	1	0	0	0
1	0	0	0	1	1	1	0	0
1	1	0	0	0	1	1	0	0
1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	0
0	1	1	0	0	0	1	1	1
0	1	1	1	0	0	0	1	1
0	0	1	1	0	0	0	1	1
0	0	1	1	1	0	0	0	1

1	0	0	1	0	0	1	0	0
1	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	1	0
0	1	0	0	1	0	0	1	0
0	1	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1

a) Bandwidth of 5 with an overlap of 3, b) Bandwidth of 3 and an overlap of -1, and c) The allowed connections matching the mask in (a).

Figure 6.5 Examples of the mask patterns.

The overlap *ov* and bandwidth *bw* controls the linear spacing of the 1's positions within a mask following (6.5).

$$lx = \ a + (g - 1)\ (l + (bw - ov)) \tag{6.5}$$

where *lx* is the linear index having an upper bound of $l \times e$, *bw* is the bandwidth, *ov* is the overlap and *l* is the length of the feature vector (number of frequency bins). The values of a are within the interval [1, bw] and the values of g are within the interval $[1, \lceil (l \times e)/(l + (bw - ov)) \rceil\ ]$.

The sparseness enforced with this band-like scheme ensures that each hidden neuron of a certain layer exhibits an interest in a certain band of frequencies by focusing on a localized region of the feature vector. Meanwhile, the spatial locality, across the frequency dimension, of the learned features is preserved as the locations of the active weights are fixed. The systematic sparseness allows the connections within a certain band of inputs (as if they are frequency bins) to contribute to the hidden node's activation.

Hand-crafting features does not only involve finding the best individual features, but also finding the optimum combination of features. The mask automates this process by embedding shifted versions of the filterbank-like pattern. This allows each neuron to learn differently about different regions of the feature vector. For example, in Figure 6.5.a, ignoring the temporal dimension for the sake of explanation, (with the columns mapping to neurons) the first neuron in a hidden layer (i.e. the first column in the mask) will learn about the 1st five features, meanwhile, the 5th neuron will learn about the 1st two features and the last feature in the feature vector. Similarly, in Figure 6.5.b the input to the first neuron is the 1st three features, the fourth neuron (i.e. the fourth column in the mask) will learn about the 1st two features in the feature vector, and the seventh neuron will learn about a single feature. Accordingly, different feature combinations are considered concurrently.

The masking operation is applied through an elementwise multiplication between the binary mask and each matrix in the set of *d* matrices. This is formulated in (6.6).

$$\hat{Z}_u = \widehat{W}_u \circ \widehat{M} \tag{6.6}$$

where $\widehat{W}_u$ is the original weight matrix, $\widehat{M}$ is the masking pattern having the same dimensions as $\widehat{W}_u$ and $\hat{Z}_u$ is the new weight matrix after the element-wise multiplication by the mask to substitute $\widehat{W}_u$ in (6.4).

Figure 6.6 shows a single step of an MCLNN of order *n*. Accordingly, a window of frames of size *2n*+1 is being processed with a weight tensor of a similar depth having matrices of a count *2n*+1. Each frame in the window at an index *u* is processed with its corresponding matrix at the same index. The highlighted cells in the figure depict the active connections. The output of a step of an MCLNN over a window of frames is a single resultant frame.

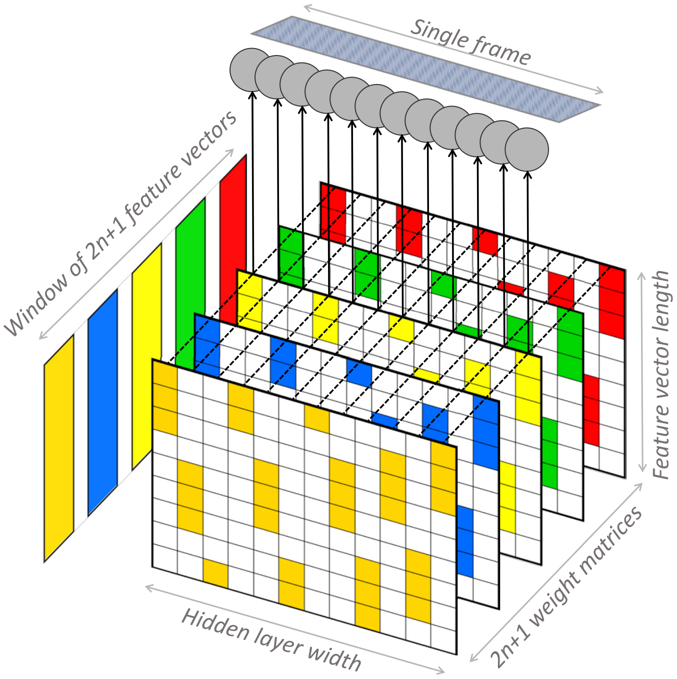


Figure 6.6 A single step of MCLNN

6.3 Summary

In this chapter, we have introduced the ConditionaL Neural Networks (CLNN) and its extension the Masked ConditionaL Neural Networks (MCLNN). The models presented in this chapter are the core contribution of this thesis. The CLNN exploit the temporal correlation across the frames of a sound signal by considering a window of frames. And since the location of the detected energy across the frequency bins is crucial for interpreting the frequency axis and consequently the sound category, the CLNN preserves the spatial locality of the learned features across the frequency bins through the use of fixed connections between the input and the hidden layer. The MCLNN uses the CLNN as the main framework to subdivide the features into bands by embedding a filterbank-like behaviour through an enforced controlled sparseness across the connections of the neural network. The MCLNN also automates the task of considering different feature combinations concurrently during training, which is usually a manual exploration mission to hand-craft the optimum combination of features. In the next two chapters, we will evaluate the performance of the proposed models through an extensive range of experiments using several publicly available datasets of environmental sounds and music.

7

Experiments

THIS chapter is composed of an extensive set of experiments to evaluate the performance of the models introduced in this thesis. The evaluation has been applied using several environmental sound (ESC-10, ESC-50, Urbansound8k and YorNoise) and music (Ballroom, GTZAN, Homburg, and ISMIR2004) datasets widely utilized in the literature. In this chapter, we also present the YorNoise dataset, which we manually collected and incorporated in the evaluation of our models. We have dedicated a special section for each dataset. We used the ISMIR2004 and the GTZAN datasets in a particular type of experiments to measure the sustainability of the MCLNN performance against the data split influence compared to the other reported works, since they are two of the oldest datasets, and they have been used in a range of attempts using different experimental settings. The following summarises the datasets used and highlights the experiments carried out with further details about each dataset postponed until their relevant section.

Ballroom: The dataset, released in 2004, is composed of 698 music files. The samples are unbalanced in distribution across the 8 music genres of the dataset. The dataset is accompanied by tempo annotations showing the BPM (Beats Per Minute) for each music file. We have used this dataset to evaluate the performance of the MCLNN compared to other attempts especially the handcrafted ones and methods that exploit the tempo nature of this dataset.

Homburg: A dataset of 9 music genres, released in 2005, of 1886 files. The dataset has a low recognition accuracy in the literature that has not surpassed 65%. We have used the Homburg dataset to evaluate the MCLNN performance against handcrafted attempts that involved multistage processing.

GTZAN: Released in 2002, it is one of the most widely adopted datasets for music genre recognition tasks. The dataset has 10 music genres with 1000 music files equally distributed among the dataset categories. Despite its popularity, the dataset suffers a range of faults, e.g. repetitions, distortion, …etc., which has been studied in [193]. Since GTZAN has been used in a number of attempts, there are various experimental settings that have been explored in the literature. We have used the GTZAN to investigate the effect of the data split on the reported accuracies in the literature including this work.

ISMIR2004: The dataset was released within the ISMIR contest in 2004. The dataset is composed of unbalanced 6 music genres with a total number of 1458 files. The experiments on this dataset exploit the wide usage of the ISMIR2004 throughout the literature with different experimental settings and data splits as in the GTZAN.

ESC-10: The dataset was collected from the Freesound project [194] (www.freesound.org) for 10 environmental sounds. Some sound samples are difficult to recognize with possible overlapping sounds in the same clip. Despite the challenging task of having a high human recognition level, it is still manageable compared to its parent dataset the ESC-50. The dataset avoids the data split influence being released in 5-folds. We have used this dataset to compare the performance of the MCLNN to Convolutional Neural Network attempts. Additionally, we have explored the effect of data augmentation, which involves applying a controlled deformation to the sound signal such as pitch shifting and temporal stretching.

ESC-50: The dataset contains 50 environmental sounds out of which 10 are used in the ESC-10 dataset. The dataset is also released in 5-folds to standardize reported accuracies. The human level accuracy varied across different sounds in the dataset as discussed in [23].

Urbansound8k: Can be considered as the largest environmental dataset with almost 9000 sound files for 10 sound categories. Also collected from the Freesound project as the ESC-10/50. The sound files are very difficult to distinguish with human hearing with the presence of background overlapping sounds in the same recording. The dataset is released into 10-folds to facilitate reporting the accuracies.

YorNoise: The dataset is manually collected in the scope of the work presented in this thesis. It focuses on two main urban sounds: rail and road traffic. The dataset is used to investigate the confusion between sounds possessing common tonal components. The experiments have used the YorNoise dataset as an extension to the Urbansound8k classes. More details on the collection process and preparation of YorNoise are provided in the relevant experiments section dedicated for the dataset.

Before we go through the experiments, we will elaborate on the common preprocessing and the models used in the experiments, and we will refer to the hardware and software environment.

We used a mel-scaled spectrogram as an intermediate signal representation for all the datasets. The transformation applied to the music datasets (Ballroom, GTZAN, Homburg and ISMIR2004) involved a logarithmically Mel-scaled spectrogram of 256 bins with an FFT window of 2048 and a 50% overlap. A similar transformation was applied to the environmental sound datasets (ESC-10, ESC-50, Urbansound8K, and YorNoise), but with 60 bins at an FFT window of 1024 and a 50% overlap. Extra spectral features are calculated for the environmental sound datasets by extracting the delta (1^{st} derivative) across the frames of the 60 bin FFT. The two spectrograms are concatenated column-wise to generate a 120-dimension feature vector. Segments of the spectrogram are extracted following (6.2) with a running step of 1, i.e. the number of overlapping frames between consecutive segments is $q - 1$ frames.

We adopted two MCLNN models for the music and the environmental sounds to cope with the different spectrogram transformation, discussed earlier, applied to the two categories. Parametric Rectified Linear Units (PRelu) [162] with zero initializations were used as activation functions. Dropout [164] was used for regularization. Models were trained to minimize the categorical cross-entropy between the predicted labels of each segment and the actual one using ADAM [159] at a learning rate of 10^{-4} using the default values published in their work, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. More details about the complexity of the models is provided in the relevant section of each experiment.

Table 7.1 and Table 7.2 list the hyperparameters used for music and environmental sounds, respectively.

Table 7.1 MCLNN Hyper-parameters for the MUSIC datasets

Layer	Type	NODES	Mask Bandwidth	Mask Overlap	Dropout
1	MCLNN	220	40	-10	1%
2	MCLNN	200	10	3	35%

Table 7.2 MCLNN Hyper-parameters for the Environmental Sound datasets

Layer	Type	NODES	Mask Bandwidth	Mask Overlap	Order *n*	Dropout
1	MCLNN	300	20	-5	15	1%
2	MCLNN	200	5	3	15	50%

The MCLNN layers are followed by a global single dimensional mean pooling [191] layer. The final output layer is a softmax layer with 10% dropout for all datasets. The Gaussian weight initialization proposed by He et al. in [162] was used for the MCLNN layers and the uniform initialization proposed in [175] for the dense layers. The default hyperparameters are listed in the tables while deferring any differences in the order *n*, the extra frames *k* and the densely-connected layers to the dataset relevant section.

All experiments used a 10-fold cross validation unless otherwise stated, with the mean and standard deviation across the folds reported. Standardization is applied to the training data using the z-score. The extracted parameters (mean and variance) are used to standardize the validation and testing data. The validation set is used for early stopping of the model training after 50 epochs, i.e. if the validation accuracy does not get better after 50 epochs, the weights stored at the *final epoch index* – 50 are used for the model. The labels assigned to the segments follows a Multiple Instance Learning paradigm [195] used in most of the attempts referenced in the literature, where the original tag of the sound file is used to label each segment in isolation and the final decision for the clip follows a voting mechanism across the predicted labels of the input segments. The final decision of a clip's category is decided based on a probability voting across the predicted vectors of the clip following (7.1).

$$Category = argmax_{j=1\ldots c}\left(\sum_{i=1}^{r} o_{ji}\right) \tag{7.1}$$

where *r* is the number of predicted output vectors following the number of total segments extracted from a clip's spectrogram. Each output vector $\hat{o}$ has a length *c*, where *c* is the number of classes predicted by the softmax. The clip's category is decided by summing the predicted distributions across all the *r* predictions per class and choosing the maximum sum.

The development machine is equipped with an Intel Xeon CPU (E5-2640v3, 2.6 GHz), 128 GB of ram and Nvidia Geforce GTX Titan X (driver version 10.18.13.5921). The implementation of the model was carried out with python (2.7.11 64-bit) using Theano [196] (version 0.8.2) and Keras [197] (version 1.0.8) with the exploit of Nvidia CUDA (version 7.5.17). FFMPEG [198] (version N-81489 built with GCC 5.4.0) was used for the sound files format conversion, and LibROSA [199] (version 0.4.0) a python signal processing library was used for the signal transformation. Muda [200] was used for sound files augmentation.

7.1 Ballroom

The Ballroom [201] music dataset is summarized in Table 7.3. The parameters used for the model, and the training complexity statistics are captured in Table 7.4. We used a deep and a shallow architecture, each is followed by two fully-connected layers of neurons of 50 and 10 nodes having a dropout of 35% and 10%, respectively, before the final 8-way softmax output. We used a mini-batch size of 600 samples.

Table 7.5 lists the accuracies achieved using the MCLNN in addition to other attempts on the Ballroom dataset. The deep MCLNN architecture (described in the common section) achieved a mean accuracy of 90.4% over the 10-folds with a standard deviation of 2.57%. The architecture is formed of two MCLNN layers and an order *n* = 15 with pooling across *k* = 10. We applied another architecture of a shallow MCLNN composed of a single layer of order *n* = 20 and extra frames *k* = 55. MCLNN achieved an accuracy of 92.12% with a standard deviation of 2.9%.

Table 7.3 Ballroom dataset

Release date	ISMIR tempo contest, 2004
Total #	698
Files format	.wav
Classes #	8
Classes (instances)	ChaCha (111) Jive (60) Quickstep (82) Rumba (98) Samba (86) Tango (86) Viennese Waltz (65) Slow Waltz (110)
Files arrangement	N/A
Sampling Rate	44100 Hz
File Duration	30 seconds

Table 7.4 Training complexity for the Ballroom dataset

Model	Number of MCLNN layers (*e* = 220)	1
	Number of Dense layers	2
	Window size (frames at *n* = 20)	41
	MCLNN trainable parameters (per layer)	2,309,340
	Total trainable parameters	2,333,368
Dataset split	Frames count per clip	600
	Features count per frame	256
	Training count (clips)	558
	Validation count (clips)	70
	Testing count (clips)	70
Input	Segment size (frames at *m* = 1, *k* = 55)	96
	Training count (segments)	281,790
	Validation count (segments)	35,350
	Testing count (segments)	35,350
	Batch size (segments)	600
Execution statistics	Average training time per fold (hours)	14
	Total training time (hours)	168
	Total memory (GB)	33

Table 7.5 MCLNN performance on the Ballroom dataset compared to other attempts in the literature

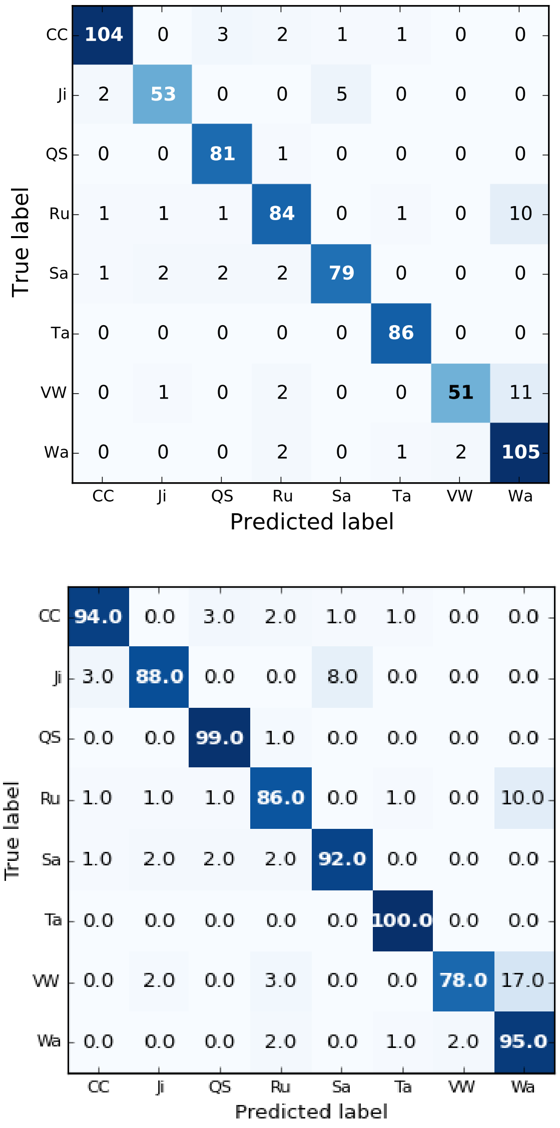
Classifier and Features	Acc. % ± Std.
SVM + 28 feature with Tempo [94]	96.13
KNN + Modulation Scale Spectrum [202]	93.12
Manhattan Distance + Block-Level features [203]	92.44
MCLNN (Shallow, *n* = 20, *k* = 55) + Mel-Spec. (this work)	**92.12 ± 2.94**
MCLNN (Deep, *n* = 15, *k* = 10) + Mel-Spec. (this work)	**90.40 ± 2.57**
SVM + Rhyth., Hist., Statist., Onset, Symb. [93]	90.40
KNN + 15 MFCC-like descriptors + Tempo [204]	90.10
KNN + Rhythm and Timbre [205]	89.20
SVM + 28 features without Tempo [94]	88.00
CNN + Mel-Scaled Spectrogram [56]	87.68 ± 4.40
SVM + Rhyth. + Hist. + Statist. features [90]	84.20
KNN + Tempo [204]	82.30

The MCLNN achieved the listed accuracies without using any musical perceptual properties or hand-crafted features that are exploited in attempts that achieved higher accuracies. For example, Peeters in [94] achieved 96.13% using the tempo annotation released with the dataset. When Peeters reapplied his method without the tempo annotations, the accuracy declined to 88%. Another work that explored the effect of the tempo annotations was in the work of Gouyon et al. [204]. They proposed several hand-crafted features, which achieved 90.1% in combination with the tempo annotations and 82.3% using tempo annotations alone. Other comparable attempts in [202] and [203] achieving 93.12% and 92.44% used a heavily engineered set of features.

In comparison to CNN attempts, the work of Pons et al. [56] tackled the translation invariant properties of the CNN, which is not optimal for spectrogram representations. In their work, they investigated the use of a shallow CNN with dedicated filters for the temporal domain and another set for the frequency domain. They also applied rectangular filters to scan the spectrogram irrespective of the temporal or frequency dimensions, in addition to filters of size [1, n] and [n, 1] to scan both the temporal and frequency domain separately. They also designed the frequency filters to fit a specific Beat Per Minute (BPM). They reported the highest accuracy of 87.68% on using a combination of the spectral and the temporal pre-trained filters as listed in Table 7.6. On the other hand, the MCLNN achieved 92.12% without any special design to exploit musical properties and using a shallow architecture as of Pons.

Table 7.6 MCLNN compared to the CNN in [56]

Classifier and Features	Acc. % ± Std.
MCLNN (Shallow, *n* = 20, *k* = 55) + Mel-Spec. (this work)	**92.12 ± 2.94**
Time-Frequency pre-trained CNN	87.68 ± 4.44
Black-box CNN	87.25 ± 3.39
Time-Frequency CNN	86.54 ± 4.29
Time CNN	81.79 ± 4.72
Frequency CNN	59.59 ± 5.82



	CC	Ji	QS	Ru	Sa	Ta	VW	Wa
CC	104	0	3	2	1	1	0	0
Ji	2	53	0	0	5	0	0	0
QS	0	0	81	1	0	0	0	0
Ru	1	1	1	84	0	1	0	10
Sa	1	2	2	2	79	0	0	0
Ta	0	0	0	0	0	86	0	0
VW	0	1	0	2	0	0	51	11
Wa	0	0	0	2	0	1	2	105

a.

	CC	Ji	QS	Ru	Sa	Ta	VW	Wa
CC	94.0	0.0	3.0	2.0	1.0	1.0	0.0	0.0
Ji	3.0	88.0	0.0	0.0	8.0	0.0	0.0	0.0
QS	0.0	0.0	99.0	1.0	0.0	0.0	0.0	0.0
Ru	1.0	1.0	1.0	86.0	0.0	1.0	0.0	10.0
Sa	1.0	2.0	2.0	2.0	92.0	0.0	0.0	0.0
Ta	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
VW	0.0	2.0	0.0	3.0	0.0	0.0	78.0	17.0
Wa	0.0	0.0	0.0	2.0	0.0	1.0	2.0	95.0

b.

Classes: ChaCha(CC), Jiva(Ji), QuickStep(QS), Rumba(Ru), Samba(Sa), Tango(Ta), Viennese Waltz(VW) and Slow Waltz(Wa). a) actual count and b) normalized count in percentage

Figure 7.1 Ballroom confusion using MCLNN.

Figure 7.1 shows the confusion across the music genres of the Ballroom dataset. The highest confusion is between the Slow Waltz and each of the Viennese Waltz and the Rumba genres, which matches findings in [206].

7.2 Homburg

A summary of the Homburg [207] dataset is listed in Table 7.7. The parameters used for the model, and the training complexity statistics are captured in

Table 7.8. We used the MCLNN architecture in the common section for music datasets with an order *n = 5* and extra frames *k = 2* for the single dimension global pooling. The pooling is followed by two densely-connected layers of 50 and 10 nodes with a dropout of 35% and 10% respectively before the 9-way softmax output. The mini-batch used for training was composed of 800 samples.

Table 7.7 Homburg dataset

Release date	2005
Total #	1886
Files format	.mp3
Classes #	9
Classes (instances)	Alternative (145) Blues (120) Electronic (113) Folkcountry (222) Funksoulrnb (47) Jazz (319) Pop (116) Raphiphop (300) Rock (504)
Files arrangement	N/A
Sampling Rate	44100 Hz
File Duration	10 seconds

Table 7.8 Training complexity for the Homburg dataset

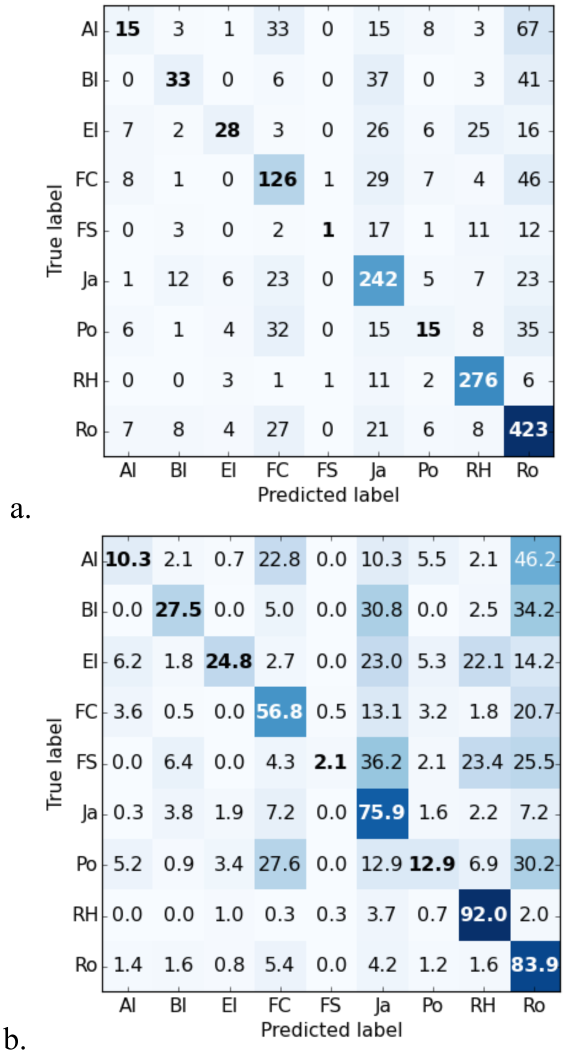
Model	Number of MCLNN layers (e = 220, e = 200)	2
	Number of Dense layers	2
	Window size (frames at n = 5)	11
	MCLNN trainable parameters (1st layer)	619,740
	MCLNN trainable parameters (2nd layer)	484,200
	Total trainable parameters	1,117,699
Dataset split	Frames count per clip	200

	Features count per frame	256
	Training count (clips)	1,508
	Validation count (clips)	189
	Testing count (clips)	189
Input	Segment size (frames at m = 2, k = 1)	22
	Training count (segments)	269,932
	Validation count (segments)	33,831
	Testing count (segments)	33,831
	Batch size (segments)	800
Execution statistics	Average training time per fold (hours)	1
	Total training time (hours)	12
	Total memory (GB)	8

Table 7.9 lists the accuracy achieved using MCLNN in addition to other attempts. The mean accuracy across 10-folds using MCLNN is 61.45% with 1.4% standard deviation. The accuracy achieved surpasses several hand-crafted attempts and is comparable to others. The highest attempts [208] on the dataset employed a set of handcrafted features using an auditory cortical representation, which models the cochlea using Constant Q-Transform followed by a wavelet transformation to extract a 4D cortical representation. This is combined with a set of MFCC and chroma features. The extracted features were used in combination with a specially designed classifier to exploit sparseness properties within the features. The MCLNN performed only slightly below these complicated handcrafted methods without any special handling. The closest neural network based attempt achieved an accuracy of 55.3% in [209] using the mean-covariance RBM (mcRBM) a variant of the RBM.

Table 7.9 MCLNN performance on the Homburg dataset
compared with attempts in the literature

Classifier and Features	Acc. % ± Std.
JSLRR + Cortical, MFCC, Chroma [208]	63.46 ± 2.49
LRSM + Cortical, MFCC, Chroma [210]	62.40 ± 3.65
MCLNN + Mel-Spectrogram (this work)	**61.45** ± 1.40
KNN + LFP,VDSP,CP,SCP [211]	61.20
SVM + ESA-MFCC [212]	57.81
KNN + Rhythm and Timbre [205]	57.00
KNN + mcRBM, PCA, MVG-MFCC [209]	55.30
SVM + Marsyas features ([37]) [213]	55.00
KNN + Multiple features [207]	53.23
SVN + Novelty Functions [206]	51.10
KNN + mcRBM, PCA, Mel-Spectrogram [214]	45.50



	Al	Bl	El	FC	FS	Ja	Po	RH	Ro
Al	**15**	3	1	33	0	15	8	3	67
Bl	0	**33**	0	6	0	37	0	3	41
El	7	2	**28**	3	0	26	6	25	16
FC	8	1	0	**126**	1	29	7	4	46
FS	0	3	0	2	**1**	17	1	11	12
Ja	1	12	6	23	0	**242**	5	7	23
Po	6	1	4	32	0	15	**15**	8	35
RH	0	0	3	1	1	11	2	**276**	6
Ro	7	8	4	27	0	21	6	8	**423**

	Al	Bl	El	FC	FS	Ja	Po	RH	Ro
Al	**10.3**	2.1	0.7	22.8	0.0	10.3	5.5	2.1	46.2
Bl	0.0	**27.5**	0.0	5.0	0.0	30.8	0.0	2.5	34.2
El	6.2	1.8	**24.8**	2.7	0.0	23.0	5.3	22.1	14.2
FC	3.6	0.5	0.0	**56.8**	0.5	13.1	3.2	1.8	20.7
FS	0.0	6.4	0.0	4.3	**2.1**	36.2	2.1	23.4	25.5
Ja	0.3	3.8	1.9	7.2	0.0	**75.9**	1.6	2.2	7.2
Po	5.2	0.9	3.4	27.6	0.0	12.9	**12.9**	6.9	30.2
RH	0.0	0.0	1.0	0.3	0.3	3.7	0.7	**92.0**	2.0
Ro	1.4	1.6	0.8	5.4	0.0	4.2	1.2	1.6	**83.9**

Classes: Alternative(Al), Blues(Bl), Electronic(El), FolkCountry(FC),
FunkSoulRnb(FS), Jazz(Ja), Pop(Po), RapHiphop(RH) and Rock(Ro).
a) actual count and b) normalized count in percentage

Figure 7.2 Homburg confusion using MCLNN.

Figure 7.2 shows the confusion of the Homburg dataset using the MCLNN. There is a proportional relation between the accuracy per genre and the number of samples available for that genre. The MCLNN achieved lower confusion across the classes than the one reported in [207].

7.3 GTZAN

Throughout the experiments of the GTZAN and the ISMIR2004 (in the next section), we are evaluating the sustainability of the MCLNN performance against the data split influence and limiting the dataset size. We used these datasets for this type of evaluation as they are two of the oldest datasets in the literature that have been used in several works using different experimental settings especially with respect to the data split. Accordingly, we will cover as many splits as possible to provide an unbiased comparison to the work reported in the literature.

The GTZAN dataset was introduced by Tzanetakis et al. in [37]. Table 7.10 lists the GTZAN properties. The parameters used for the model, and the training complexity statistics are captured in Table 7.11.

Table 7.10 GTZAN dataset

Release date	2002
Total #	1000
Files format	.ac
Classes #	10
Classes (instances)	Blues (100) Classical (100) Country (100) Disco (100) Hip-hop (100) Jazz (100) Metal (100) Pop (100) Reggae (100) Rock (100)
Files arrangement	N/A
Sampling Rate	22050 Hz
File Duration	30 seconds

Table 7.11 Training complexity for the GTZAN dataset

Model	Number of MCLNN layers (e = 220, e = 200)	2
	Number of Dense layers	1
	Window size (frames at n = 4)	9
	MCLNN trainable parameters (1st layer)	507,100
	MCLNN trainable parameters (2nd layer)	396,200
	Total trainable parameters	920,290
Dataset split	Frames count per clip	600
	Features count per frame	256
	Training count (clips)	800
	Validation count (clips)	100
	Testing count (clips)	100
Input	Segment size (frames at m = 2, k = 10)	27
	Training count (segments)	459,200
	Validation count (segments)	57,400
	Testing count (segments)	57,400

	Batch size (segments)	600
Execution statistics	Average training time per fold (hours)	7
	Total training time (hours)	70
	Total memory (GB)	16

We adopted a two-layer MCLNN as described in the common section for music datasets with an order $n = 4$ followed by a single fully-connected layer of 50 neurons with an input dropout of 35%. The extra frames $k = 10$, in addition to the window's middle frame, and the minibatch size is 600 samples. The results reported in this section used majority voting across the frames for the clip's category to be comparable to other attempts in the literature. The dataset has no specific distribution across the train, test and validation splits. We experimented using a 10-fold cross-validation process. To evaluate the sustainability of the accuracy against the data split, we randomly generated 10 trials of the 10-fold experiments using the system clock as a seed for the random number generator across the 10 trials. MCLNN achieved a mean accuracy of 85.1% across the 10-folds of the cross-validation and 84.1% across the 100 runs (10 trials × 10-folds) on the GTZAN as listed in Table 7.12. The highest accuracy of 92.7% in [215] proposed using a set of 10 features designed to exploit long-time and short- time acoustic features for genre classification, e.g. octave-based spectral contrast, octave-based modulation spectral contrast, modulation spectral flatness measure, to name a few, in addition to MFCC, spectral flux and others. They also used a specially designed classifier based on Compressive Sampling. A comparable accuracy of 92.4% was also achieved in [46] with a similar complicated system using an auditory cortical representation that is dimensionally reduced using non-negative matrix factorization and finally classified using a sparse based classifier. The work in [47] achieved 91.4% using a specially designed signal transform that aims to provide a frequency shift-invariant representation of the signal. The accuracy reported in this method is also tightly coupled with the fine-tuning of the grid-search for the optimum RBF-SVM parameters. The MCLNN surpassed several state-of-the-art methods that are dependent on hand-crafted features or neural networks.

Figure 7.3 shows the confusion across the GTZAN genres. The highest accuracy achieved was for the classical music of 99%, and the lowest was for the Rock genre, which overlaps with the confusion rates reported in [193].

Table 7.12 MCLNN performance on the GTZAN dataset compared with other attempts in the literature

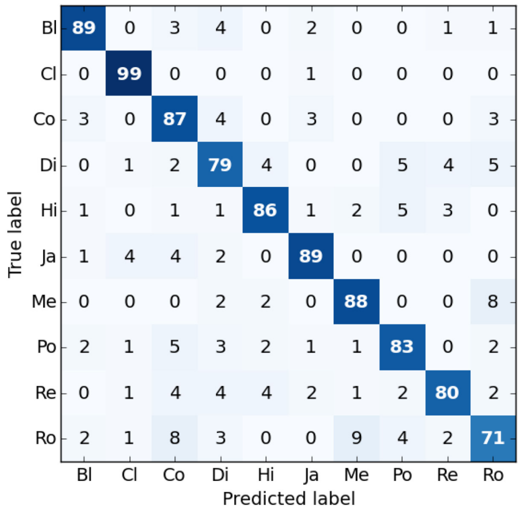
Classifier and Features	Acc.% ± Std.
Compressive Sampling + Multiple feature sets [215]^{2}	92.7
SRC + LPNTF + Auditory Cortical features [46]^{2}	92.4
RBF-SVM + Scattering Transform [47]^{2}	91.4 ± 2.2
MCLNN + Mel-Scaled Spectrogram (this work)^{2}	**85.1 ± 3.3**
RBF-SVM + Spectrogram – DBN [51]^{4}	84.3
MCLNN + Mel-Scaled Spectrogram (this work)^{3}	**84.1 ± 4.0**
Linear SVM + PSD on Octaves [48]^{3}	83.4 ± 3.1
Random Forest + Spectrogram – DBN [216]	83.0 ± 1.1
AdaBoost + Several features [38]^{1}	83.0
RBF SVM + Spectral Covariance [217]^{2}	81.0
Linear SVM + PSD on Frames [48]^{3}	79.4 ± 2.8
SVM + DWCH [88]^{2}	78.5 ± 4.1
Logstic Regression + Spectral Covariance [217]^{2}	77.0
LDA + MFCC, FFT, Beat and Pitch [89]^{2}	71.0
GMM +MFCC, FFT, Rhythm and Pitch [37]^{2}	61.0 ± 4.0

^{1}5-fold cross-validation

^{2}10-fold cross-validation

^{3}10×10-fold cross-validation

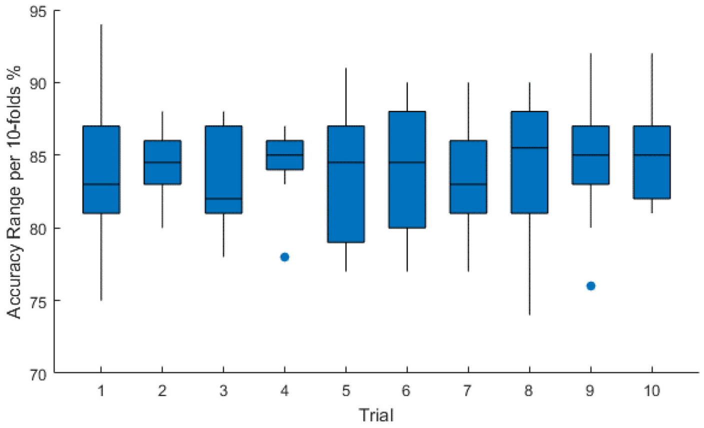
^{4}50% training, 20% validation and 30% testing



	Bl	Cl	Co	Di	Hi	Ja	Me	Po	Re	Ro
Bl	89	0	3	4	0	2	0	0	1	1
Cl	0	99	0	0	0	1	0	0	0	0
Co	3	0	87	4	0	3	0	0	0	3
Di	0	1	2	79	4	0	0	5	4	5
Hi	1	0	1	1	86	1	2	5	3	0
Ja	1	4	4	2	0	89	0	0	0	0
Me	0	0	0	2	2	0	88	0	0	8
Po	2	1	5	3	2	1	1	83	0	2
Re	0	1	4	4	4	2	1	2	80	2
Ro	2	1	8	3	0	0	9	4	2	71

Classes: Blues(Bl), Classical(Cl), Country(Co), Disco(Di), Hip-hop(Hi), Jazz(Ja), Metal(Me), Pop(Po), Reggae(Re) and Rock(Ro)

Figure 7.3 Confusion matrix for the GTZAN dataset



A plot of 10 attempts of 10-folds cross-validation using the MCLNN

Figure 7.4 Boxplot for 10 trials on the GTZAN dataset

The 10 trials reported varying accuracies accounting to the difference of the data distributions generated in-between the trials. Figure 7.4 reports a boxplot for the accuracies reported for each trial of the 10-folds cross-validation we applied on the GTZAN dataset using the MCLNN, where the accuracy ranged from 74% to 94% over the 100 training runs. Though the GTZAN has analytical problems as studied by Strum [193], the problem is not confined to the GTZAN dataset as we will elaborate in the ISMIR2004 section. The fluctuation of the accuracies across the trials reveals the influence of the data split on reported accuracies in the literature that do not consider the data split. For example, in the work of Hamel et al. [51], they used a DBN architecture to extract features from spectrograms, and the extracted abstract features of the DBN were further fed to an SVM for classification. In their experiments, they used a fixed data split (50% training, 20% validation and 30% testing) and they did not use cross-validation for choosing the SVM parameters. Kereliuk et al. [15] used the same architecture proposed by Hamel et al. and they achieved 81.5 %. The lower accuracy reported by Kereliuk et al. compared to Hamel et al. could be greatly accounted to the data split influence. The split influence was considered by Sigtia et al. [216]. They applied a fixed (50% training, 25% validation and 25% testing) split for 4 times and reported a mean accuracy of 83%. In a similar context to experimental settings, Henaff et al. [48] used the whole dataset to train their unsupervised method, where they achieved an accuracy of 83.4%. In this setting, the model is aware of the test samples, which affects the reported accuracy as discussed by Henaff. When they repeated the experiment on the training set only, they achieved an accuracy of 80%. In their work, they showed variability in the accuracies across 10 attempts of cross-validation that ranged between 77% to 87%.

Table 7.13 GTZAN Random and Fault-Filtered accuracy using the splits by Kereliuk et al. [15]

	Random Acc. %	Filtered Acc. %
MCLNN (this work)	**84.4**	**65.8**
DNN [15]	81.2	49.0

For further evaluation of the MCLNN against the split influence, we adopted the publicly available split released in the work of Kereliuk et al. [15]. In their work, they released a randomly stratified split (50% training, 25% validation and 25% testing) of the GTZAN dataset and another fault-filtered split. The fault-filtered split removed the repeated and distorted files as discussed by Sturm [193]. The MCLNN achieved an accuracy of 84.4% and 65.8% for the random and fault-filtered splits, respectively, which outperforms the DNN attempt by Kereliuk et al. [15] as listed in Table 7.13. With regard to the dataset size, the works in [15, 38, 48, 51, 217] used an FFT window of 1024 samples. The MCLNN used a window of 2048, which decreases the number of samples by 50% and consequently the training complexity.

7.4 ISMIR2004

The ISMIR2004 dataset was released within the scope of the ISMIR conference in 2004. Table 7.14 lists the dataset properties. The parameters used for the model, and the training complexity statistics are captured in Table 7.15. We extracted 30 seconds from each file following [218]. Further preprocessing followed the one described in the common section for the music datasets. We adapted the MCLNN architecture used for the GTZAN dataset and applied it in the experiments of the ISMIR2004 without any changes to the hyperparameters, except for the minibatch size, to evaluate the generalization of the models to datasets of different distributions.

Table 7.14 ISMIR2004 dataset

Release date	2004
Total #	1458
Files format	.mp3
Classes #	6
Classes (instances)	Classical (640) Electronic (229) Jazz-Blues (52) Metal-Punk (90) Rock-Pop (203) World (244)

Files arrangement	729 train / 729 test
Sampling Rate	44100 Hz
File Duration	Different durations

Table 7.15 Training complexity for the ISMIR2004 dataset

Model	Number of MCLNN layers (e = 220, e = 200)	2
	Number of Dense layers	1
	Window size (frames at n = 4)	9
	MCLNN trainable parameters (1st layer)	507,100
	MCLNN trainable parameters (2nd layer)	396,200
	Total trainable parameters	920,086
Dataset split	Frames count per clip	600
	Features count per frame	256
	Training count (clips)	1,170
	Validation count (clips)	144
	Testing count (clips)	144
Input	Segment size (frames at m = 2, k = 10)	27
	Training count (segments)	670,263
	Validation count (segments)	82,656
	Testing count (segments)	82,656
	Batch size (segments)	1000
Execution statistics	Average training time per fold (hours)	9
	Total training time (hours)	90
	Total memory (GB)	23

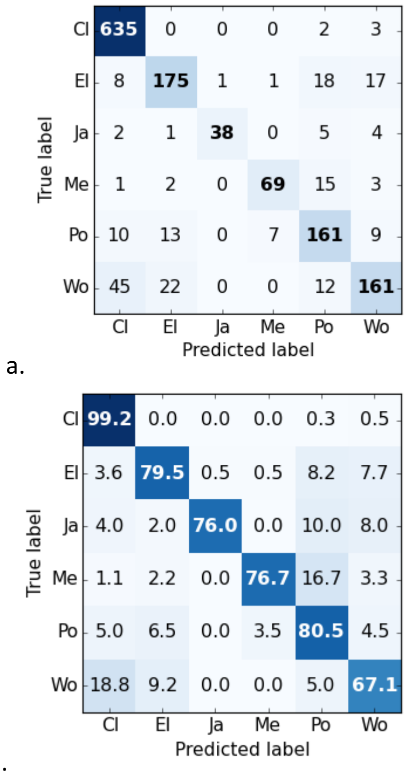
In an endeavour to prove that the effect of the data split is not specific to the GTZAN dataset only (due to its analytical issues as discussed earlier), but rather it is a general issue for datasets, we applied two sets of experiments on the ISMIR2004 dataset. In the first set, we used the ISMIR contest data split (729 files training / 729 files testing), where the 729 training data was divided into 90% training and 10% validation. To investigate the split influence, we repeated the experiment for 10 trials, where the files of each trial were randomly assigned between the 90% training set or the 10% validation set using the system clock as a seed for the random number generator. In the second set, we combined the contest splits resulting in a collection of 1458 files. The files for each category were randomly distributed across 10-folds and to validate the split influence; we repeated the 10-fold distribution for 10 trials using the system clock.

Table 7.16 lists the accuracies reported using the MCLNN on the ISMIR2004, where the it achieved a mean accuracy of 86% over a 10-fold cross validation (the confusion matrix is shown in Figure 7.5), and the highest accuracy for the contest split was 84.77%. The mean accuracy across the 10 trials of the 10-folds (100 runs) achieved an accuracy of 84.83% and the mean of the 10 trials of the contest split achieved 83.13%. Accuracies higher than the MCLNN exploit using hand-crafted features. For example, the highest accuracy of 94.38% exploited using psychophysiological properties of the human ear represented in the auditory cortical features [46] in combination with a multilinear dimensionality reduction. Despite achieving 94.4% using a Sparse Representation Classifier, the method did not exceed an accuracy of 75% on using a Linear-SVM for classification. Others attempts in [205] and [203] used rhythmic and hand-crafted features. On the other hand, the MCLNN achieved the listed accuracies without any hand-crafted features, which allows extending MCLNN to any multi-channel temporal signal, which we will explore in future work.

Table 7.16 ISMIR2004 Classification Accuracy

Classifier and Features	Acc. %
SRC + NTF + Auditory Cortical features [46][1]	94.38
KNN + Rhythmic descriptors and timbre [205][4]	90.04
SVM + Several Block-Level features [203][7]	88.27
MCLNN + Mel-Scaled Spectrogram (this work)[4]	**86.04 ± 1.44**
MCLNN + Mel-Scaled Spectrogram (this work)[5]	**84.83 ± 3.04**
MCLNN + Mel-Scaled Spectrogram (this work)[1]	**84.77**
GMM + NMF [43][3]	83.50
MCLNN + Mel-Scaled Spectrogram (this work)[2]	**83.13 ± 1.46**
SVM + Audio and Symbolic features [93][4]	81.40
Nearest Neighbour + Spec. Similarity and FP [91][6]	81.00
SVM + High-Order SVD [45][4]	80.95
SVM + Rhythmic Patterns and SSD [90][2]	79.70

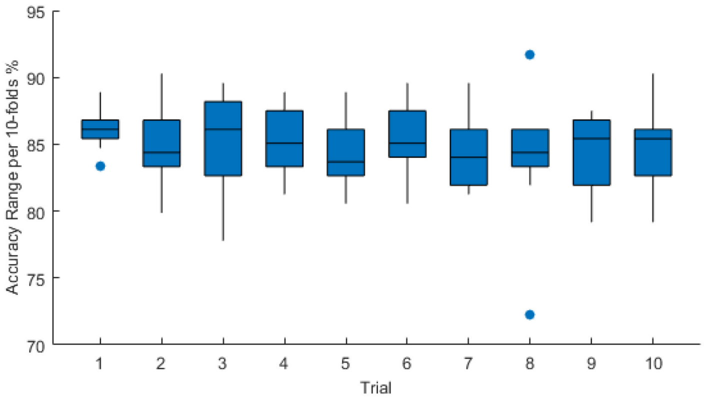
[1] Train 729 file / test 729 file
[2] 10× (Train 729 file / test 729 file)
[3] 5-fold cross-validation
[4] 10-fold cross-validation
[5] 10×10-fold cross-validation
[6] leave-one-out cross-validation
[7] Not referenced



Classes: Classical(Cl), Electronic(El), Jazz/Blues(Ja), Metal/Punk (Me), Pop/Rock(Po) and World(Wo). a) actual count and b) normalized count in percentage

Figure 7.5 Confusion matrix for the ISMIR2004

Figure 7.6 shows the variations in the accuracies across the 100 runs of cross-validation, ranging from 72.22% to 91.67%, which overlaps with similar variations in the GTZAN dataset.



A plot of 10 attempts of 10-folds cross-validation using the MCLNN

Figure 7.6 Boxplot for 10 trials on the ISMIR2004 dataset.

7.5 ESC-10

Table 7.17 summarizes the ESC-10 [23] dataset. The parameters used for the model, and the training complexity statistics are captured in Table 7.18, and in a different setting, we applied augmentation to the ESC-10 dataset, where the training complexity statistics are listed in Table 7.19.

The ESC-10 files are fixed length of 5 seconds with shorter events originally padded with silence. Accordingly, we are trimmed all the files to remove the zero paddings. All files were cloned several times and concatenated using FFmpeg [198]. Following the cloning process, 5 seconds were extracted from each file, before applying the processing discussed earlier in the common section.

Table 7.17 ESC-10 dataset

Release date	2015
Total #	400
Files format	.ogg
Classes #	10
Classes (instances)	Baby Cry (40) Chainsaw (40) Clock Tick (40) Dog Bark (40) Fire Cracking (40) Helicopter (40) Person Sneeze (40) Rain (40) Rooster (40) Sea Waves (40)
Files arrangement	5-folds
Sampling Rate	44100 Hz
File Duration	5 seconds

Table 7.18 Training complexity for the ESC-10 dataset

Model	Number of MCLNN layers (e = 300, e = 200)	2
	Number of Dense layers	2
	Window size (frames at n = 15)	31
	MCLNN trainable parameters (1st layer)	1,116,300
	MCLNN trainable parameters (2nd layer)	1,860,200
	Total trainable parameters	3,037,410
Dataset split	Frames count per clip	200
	Features count per frame	120
	Training count (clips)	240

	Validation count (clips)	80
	Testing count (clips)	80
Input	Segment size (frames at m = 2, k = 40)	101
	Training count (segments)	24000
	Validation count (segments)	8000
	Testing count (segments)	8000
	Batch size (segments)	600
Execution statistics	Average training time per fold (hours)	2
	Total training time (hours)	11
	Total memory (GB)	2.5

Table 7.19 Training complexity for the ESC-10 dataset with Augmentation

Model	Number of MCLNN layers (e = 300, e = 200)	2
	Number of Dense layers	2
	Window size (frames at n = 15)	31
	MCLNN trainable parameters (1st layer)	1,116,300
	MCLNN trainable parameters (2nd layer)	1,860,200
	Total trainable parameters	3,027,410
Dataset split	Frames count per clip	200
	Features count per frame	120
	Training count (clips)	3,120
	Validation count (clips)	80
	Testing count (clips)	80
Input	Segment size (frames at m = 2, k = 20)	81
	Training count (segments)	374,400
	Validation count (segments)	9,600
	Testing count (segments)	9,600
	Batch size (segments)	600
Time and Memory	Average training time per fold (hours)	13
	Total training time (hours)	67
	Total memory (GB)	15

We used two densely-connected layers of neurons of 100 nodes and a 50% dropout each following the common MCLNN architecture used for environmental sounds. The mini-batch size used 600 samples per batch. Table 7.20 lists accuracies achieved using the MCLNN in addition to other attempts in the literature. The MCLNN surpassed several CNN attempts, achieving an accuracy of 85.5% without augmentation compared for example to the Piczak-CNN that achieved 80% using 10 augmentation variants. The attempt of Piczak [87] used state-of-the-art CNN model formed of two convolutional, two pooling and two fully-connected layers of 5000 neurons each. The Piczak-CNN used two channels one for a 60 Mel-Scaled spectrogram and the other channel for delta (1st derivative across the spectrogram frames). Piczak extended the ESC-10 dataset with 10 augmentation variants for each file in the dataset, which involves applying a deformation to the signal such as time delay and pitch shifting. Augmentation is a method to increase the dataset and consequently the generalization of the model to more samples, which increases the accuracy. The influence of augmentation was also studied in [16]. The MCLNN achieved 85.25% using 12 augmentation variants (generated using Muda [200]), where 8 variants involved pitch shifting (ranging from -3.5 to 3.5 with a 0.5 semitone-shift), following a subset of the pitch-shift augmentations proposed in [16] and 4 variants of time delays (ranging from -0.3 to 0.3 logarithmic space-shifting). Despite the increase in the dataset size using augmentation, which consequently requires a larger model with more trainable parameters, we used the same MCLNN applied to the nonaugmented version of the dataset and the MCLNN was capable of achieving a comparable accuracy.

Table 7.20 MCLNN performance on the ESC-10 dataset compared with other attempts in the literature

Classifier and Features	Acc. % ± Std.
MCLNN (*k* = 40, 101 frames) + Mel-Scaled Spec. (this work)[2]	**85.50 ± 4.91**
MCLNN (*k* = 20, 81 frames) + Mel-Scaled Spec. (this work)[1]	**85.25 ± 4.70**
MCLNN (*k* = 1) + Mel-Scaled Spec. (this work)[2]	**83.00 ± 4.00**
SoundNet (layers = 5) + Raw Waveform [219][2]	82.30
MCLNN (*k* = 25) + Mel-Scaled Spec. (this work)[2]	**82.00 ± 5.04**
Piczak-CNN (Long Segment, 101 frames)+ Mel-Scaled Spec. [87][1]	80.00
Piczak-CNN (Short Segment, 41 frames) + Mel-Scaled Spec. [87][1]	78.50
CLNN (*k* = 25) + Mel-Scaled Spec. (this work)[2]	**77.50 ± 4.26**
CLNN (*k* = 40, 101 frames) + Mel-Scaled Spec. (this work)[2]	**75.75 ± 3.22**
SoundNet (layers = 8) + Raw Waveform [219][2]	75.50
CLNN (*k* = 1) + Mel-Scaled Spec. (this work)[2]	**73.25 ± 5.22**
Random Forest + MFCC [23][2]	72.70

[1] *Augmentation*
[2] *Without Augmentation*

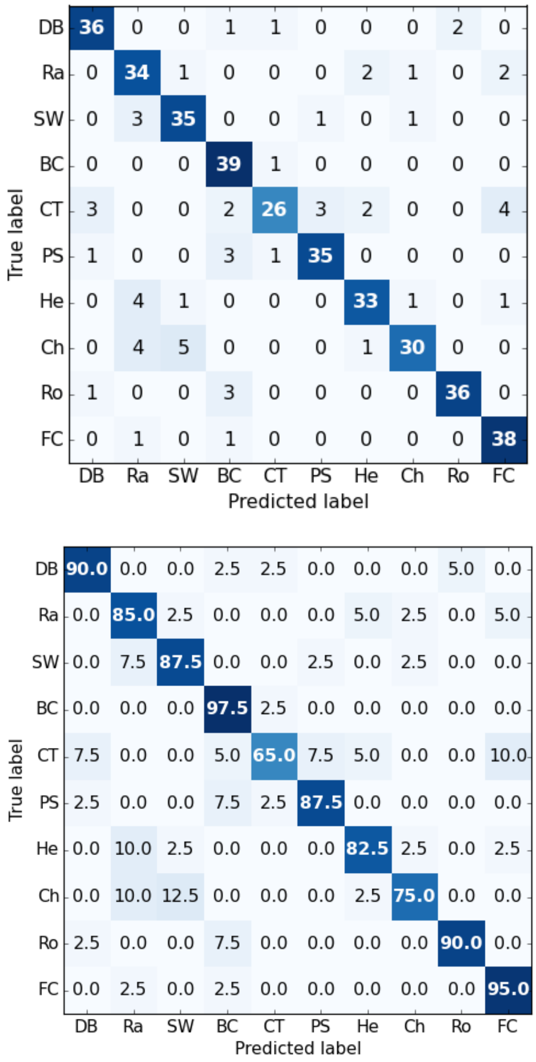
We adopted the spectrogram transformation used by Piczak to avoid the influence of the data transformation in evaluating the performance of the MCLNN. Piczak-CNN achieved 80% accuracy using 25 million parameters compared to the 3 million parameters utilized by the MCLNN. On the other hand, the MCLNN achieved 85.5% using 12% of the parameters used by Piczak-CNN with the same intermediate signal representation.

The work of Piczak studied the influence of the segment size introduced to the network, where he attempted using a *long segment* of 101 frames and a *short segment* of 41 frames. The MCLNN segment size for the model we used was 101 frames in length (n = 15, m = 2 and k = 40 + 1, 1 for the windows middle frame), which matches the same *long segment* of Piczak-CNN.

The work of Aytar et al. [219] proposed two different CNN models of depth 5 and 8 layers, respectively. They achieved an accuracy of 82.3% using the 5 layers model and 75.5% with the 8 layers architecture trained on the raw waveform. Their attempt did not only introduce a convolutional architecture for the sound classification problem, but they also proposed leveraging the natural synchronization between the vision and sound in videos to enhance the training of their proposed architectures. They experimented using deep architectures such as VGG [220] and AlexNet [13], each pretrained on two public datasets composed of millions of images (ImageNet is 1.2 million and Places is 10 million). The two deep CNN vision networks were used to extract features from 2 million videos forming a year of continuous sound and video, where the audio channel was used to train another deep CNN network on the raw signal. The accuracy of their pretrained sound network surpassed the performance in Table 7.20. Accordingly, we will consider using the MCLNN in combination with vision networks to enhance the performance. Arandjelovic et al. [221] proposed a similar paradigm to the SoundNet, which will be discussed in the next section.

To evaluate the performance of the CLNN, we used the same architectures utilized for the MCLNN, but without the masking. Table 7.20 lists the accuracies achieved using the CLNN. The influence of the mask is clear from the CLNN reported accuracies accounting for the functionalities introduced by the mask discussed earlier.

Figure 7.7 shows the actual confused file count and the normalized percentage of confusion. There is a high confusion rate between the Clock Tick sounds and the Fire Cracking, due to the overlapping short events. Confusion is also noticed among the Chainsaw sound and both the Rain and Sea Waves categories due to the common low-frequency components.



a.

	DB	Ra	SW	BC	CT	PS	He	Ch	Ro	FC
DB	36	0	0	1	1	0	0	0	2	0
Ra	0	34	1	0	0	0	2	1	0	2
SW	0	3	35	0	0	1	0	1	0	0
BC	0	0	0	39	1	0	0	0	0	0
CT	3	0	0	2	26	3	2	0	0	4
PS	1	0	0	3	1	35	0	0	0	0
He	0	4	1	0	0	0	33	1	0	1
Ch	0	4	5	0	0	0	1	30	0	0
Ro	1	0	0	3	0	0	0	0	36	0
FC	0	1	0	1	0	0	0	0	0	38

b.

	DB	Ra	SW	BC	CT	PS	He	Ch	Ro	FC
DB	90.0	0.0	0.0	2.5	2.5	0.0	0.0	0.0	5.0	0.0
Ra	0.0	85.0	2.5	0.0	0.0	0.0	5.0	2.5	0.0	5.0
SW	0.0	7.5	87.5	0.0	0.0	2.5	0.0	2.5	0.0	0.0
BC	0.0	0.0	0.0	97.5	2.5	0.0	0.0	0.0	0.0	0.0
CT	7.5	0.0	0.0	5.0	65.0	7.5	5.0	0.0	0.0	10.0
PS	2.5	0.0	0.0	7.5	2.5	87.5	0.0	0.0	0.0	0.0
He	0.0	10.0	2.5	0.0	0.0	0.0	82.5	2.5	0.0	2.5
Ch	0.0	10.0	12.5	0.0	0.0	0.0	2.5	75.0	0.0	0.0
Ro	2.5	0.0	0.0	7.5	0.0	0.0	0.0	0.0	90.0	0.0
FC	0.0	2.5	0.0	2.5	0.0	0.0	0.0	0.0	0.0	95.0

Classes: Dog Bark(DB), Rain(Ra), Sea Waves(SW), Baby Cry(BC), Clock Tick(CT),Person Sneeze(PS), Helicopter(He), Chainsaw(Ch), Rooster(Ro) and Fire Cracking(FC). a) actual count and b) normalized count in percentage

Figure 7.7 Confusion matrix for the ESC-10 dataset.

7.6 ESC-50

The ESC-50 [23] dataset has a collection of 50 classes of environmental sounds. The dataset is the parent collection of the ESC-10 dataset. We applied the same transformations applied on the ESC-10 dataset; we trimmed the files to remove silence, cloned the files to shorter than 5 seconds and extracted 5 seconds from each file in the dataset. Further pre-processing involves the one applied to the environmental sounds datasets as described in the common section earlier. Table 7.21 lists the ESC-50 classes. The parameters used for the model, and the training complexity statistics are captured in Table 7.22, and in a different setting, we applied augmentation to the ESC-50 dataset, where the training complexity statistics are listed in Table 7.23.

Table 7.21 ESC-50 dataset

Release date	2015		
Total #	2000		
Files format	.ogg		
Classes #	10		
Classes (instances)	Hand saw	Keyboard typing	Water drops
	Fireworks	Mouse click	Chirping birds
	Airplane	Knocking	Crickets
	Church bells	Drinking / sipping	Crackling fire
	Train	Snoring	Sea waves
	Engine	Tooth brushing	Rain
	Car horn	Laughing	Crow
	Siren	Footsteps	Sheep
	Chainsaw	Coughing	Insects
	Helicopter	Breathing	Hen
	Glass breaking	Clapping	Cat
	Ticking clock	Sneezing	Frog
	Alarm clock	Baby crying	Cow
	Vacuum cleaner	Thunderstorm	Pig
	Washing machine	Toilet flush	Rooster
	Can opening	Pouring water	Dog
	Creaks (door/wood)	Wind	
Files arrangement	5-folds		
Sampling Rate	44100 Hz		
File Duration	5 seconds		

Table 7.22 Training complexity for the ESC-50 dataset

Model	Number of MCLNN layers ($e = 300$)	1
	Number of Dense layers	2
	Window size (frames at $n = 14$)	29
	MCLNN trainable parameters	1,044,300
	Total trainable parameters	1,102,050
Dataset split	Frames count per clip	200
	Features count per frame	120
	Training count (clips)	1,200
	Validation count (clips)	400
	Testing count (clips)	400
Input	Segment size (frames at $m = 1$, $k = 40$)	69
	Training count (segments)	158,400
	Validation count (segments)	52,800
	Testing count (segments)	52,800
	Batch size (segments)	300

Execution statistics	Average training time per fold (hours)	5
	Total training time (hours)	25
	Total memory (GB)	9

Table 7.23 Training complexity for the ESC-50 dataset with Augmentation

Model	Number of MCLNN layers (e = 300)	1
	Number of Dense layers	2
	Window size (frames at n = 14)	29
	MCLNN trainable parameters	1,044,300
	Total trainable parameters	1,102,050
Dataset split	Frames count per clip	200
	Features count per frame	120
	Training count (clips)	6,000
	Validation count (clips)	400
	Testing count (clips)	400
Input	Segment size (frames at m = 1, k = 40)	69
	Training count (segments)	792,000
	Validation count (segments)	52,800
	Testing count (segments)	52,800
	Batch size (segments)	300
Execution statistics	Average training time per fold (hours)	22
	Total training time (hours)	111
	Total memory (GB)	29.5

We used two fully-connected layers of 100 nodes each of a dropout of 50% each. We used a shallow MCLNN model adopted from the common section but with an order n = 14 and extra frames k = 40. Table 7.24 lists the accuracies achieved using the MCLNN and other attempts. A shallow MCLNN achieved an accuracy of 62.85% without any augmentation. The Piczak-CNN referenced in the table is the same architecture used for the ESC-10 dataset, which is composed of 25 million parameters. It achieved an accuracy of 64.5% with 4 augmentations. The shallow MCLNN achieved a comparable accuracy, without augmentation, using approximately 1 million parameters. Accordingly, the accuracies reported with the MCLNN account for 4% of the parameters utilized by the Piczak-CNN. Moreover, the spectrogram transformation used for the MCLNN is the same transformation adopted by Piczak-CNN.

Augmentation increases the accuracy as studied by Salamon et al. in [16]. The Piczak-CNN [87] applied 4 augmentation variants using different time delays and pitch shifting.

To investigate the effect of the augmentation, we applied 4 augmentation variants (2 pitch-shifting and 2 time-delay). The MCLNN achieved 66.25%, which is higher than the augmented attempt using the Piczak-CNN.

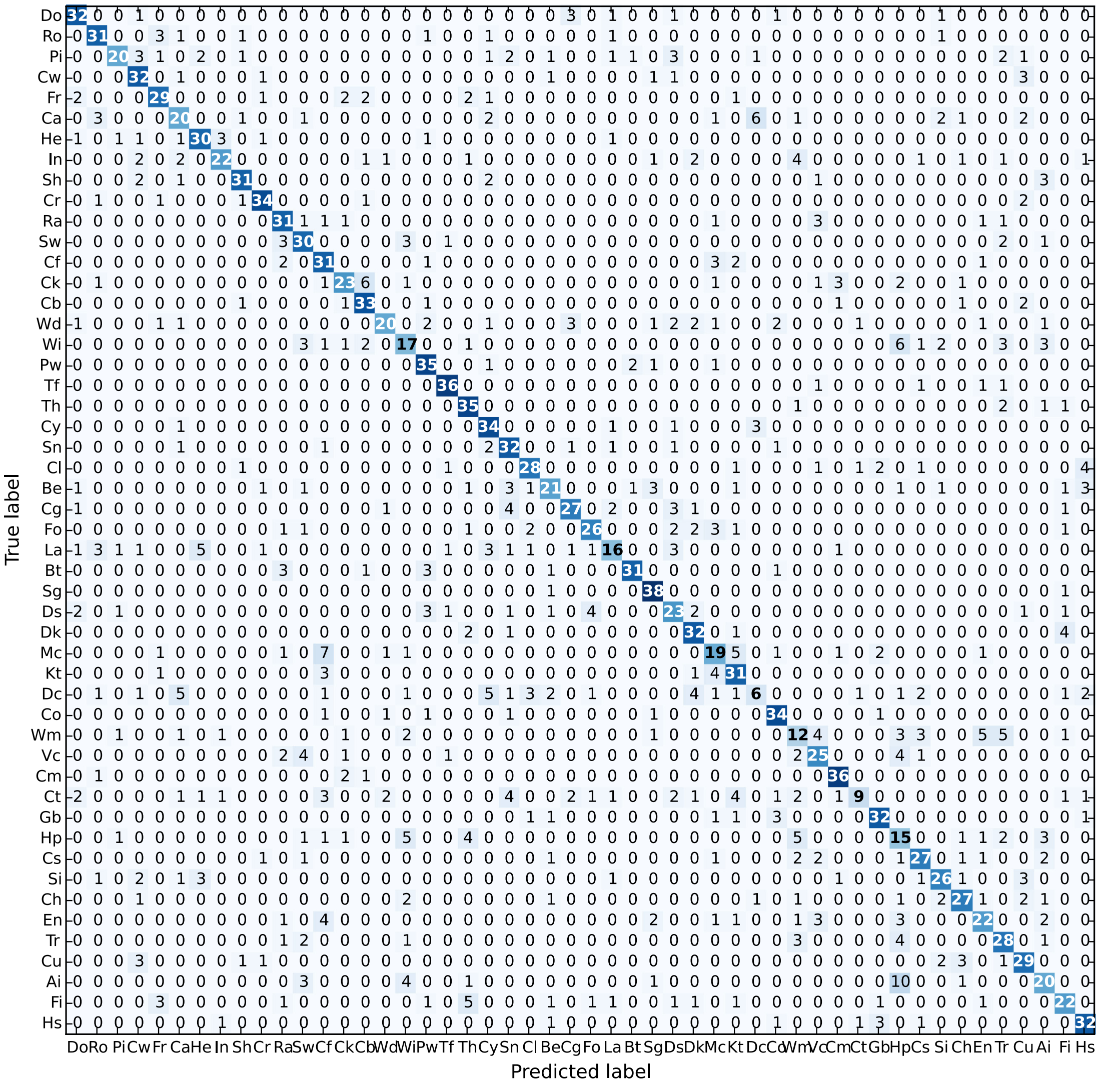
Table 7.24 MCLNN performance on the ESC-50 dataset compared with other attempts in the literature

Classifier and Features	Acc. % ± Std.
MCLNN (Shallow, k = 40, n = 14) + Mel-Spectrogram (This Work)[1]	**66.25 ± 1.47**
SoundNet (layers = 5) + Raw Waveform [219][2]	65.00
Piczak-CNN + Mel-Spectrogram [87][1]	64.50
MCLNN (Shallow, k = 40, n = 14) + Mel-Spectrogram (This Work)[2]	**62.85 ± 2.39**
L^3-Net (layers = 8) + Log-Spec. [221][2]	62.50
MCLNN (Deep, k = 6, n = 14) + Mel-Spectrogram (This Work)[2]	**61.75 ± 2.20**
SoundNet (layers = 8) + Raw Waveform [219][2]	51.10
Random Forest + MFCC [23][2]	44.30

[1] *Augmentation*
[2] *Without Augmentation*

The work of Aytar et al. [219] (SoundNet), as discussed in the previous section, proposed using two pretrained vision networks to instruct the training of a deep CNN network for the audio channel of a video. They proposed two deep architectures of the SoundNet of 5 layers and 8 layers, which achieved 65% and 51.1%, respectively. The work of Arandjelovic et al. [221] (L^3-Net) proposed a similar architecture, but they investigated the performance of training the vision network concurrently from scratch. They trained the vision and the audio network on a task they refer to as the Audio-Visual Correspondence (AVC). They trained their model using 0.5 million videos, where the input to the vision network was an image of 224 × 224 in size and the sound network was trained on a 257 bins logarithmically scaled spectrogram. The output of the two networks is further fused through dense layers. They achieved an accuracy of 62.5% with their architecture without pretraining, and with the help of the vision network, trained on the 0.5 million videos, they achieved higher accuracies than the ones in Table 7.24. The MCLNN surpassed both the L^3-Net and the Soundnet without the visual networks. However, both works achieved accuracies higher than the ones listed in the table by incorporating a visual network. Accordingly, induced by the L^3-Net and SoundNet, we will explore using the visual network to help in training an MCLNN deep network on the sound signal.

Figure 7.8 shows the confusion matrix for the 50 sound categories of the ESC-50 dataset using the MCLNN with augmentation.



Ai	Airplane	Cm	Clock Alarm	Do	Dog	In	Insects	Si	Siren
Be	Breathing	Co	Can opening	Ds	Drinking-sipping	Kt	Keyboard typing	Sn	Sneezing
Bt	Brushing teeth	Cr	Crow	En	Engine	La	Laughing	Sw	Sea Waves
Ca	Cat	Cs	Chainsaw	Fi	Fireworks	Mc	Mouse Click	Tf	Toilet flush
Cb	Chirping birds	Ct	Clock Tick	Fo	Footsteps	Pi	Pig	Th	Thunderstorm
Cf	Crackling fire	Cu	Church bells	Fr	Frog	Pw	Pouring water	Tr	Train
Cg	Coughing	Cw	Cow	Gb	Glass Breaking	Ra	Rain	Vc	Vacuum Cleaner
Ch	Car horn	Cy	Crying baby	He	Hen	Ro	Rooster	Wd	Water drops
Ck	Crickets	Dc	Door-wood	Hp	Helicopter	Sg	Snoring	Wi	Wind
Cl	Clapping	Dk	Door knock	Hs	Hand saw	Sh	Sheep	Wm	Washing Machine

Figure 7.8 Confusion matrix for the ESC-50 dataset.

7.7 Urbansound8k

The dataset is collected from the www.freesound.org. Accordingly, the original files have different sampling rate and sizes. The authors of the dataset, extracted clips of a maximum of 4 seconds from the files. As an initial pre-processing step, we cloned files having events of less than 4 seconds in duration, e.g. gunshots, several times and extracted 4 seconds from the generated files. Further spectrogram transformation followed the process described in the common section for environmental sounds. Table 7.25 lists a summary of the Urbansound8K [24] dataset. The parameters used for the model, and the training complexity statistics are captured in Table 7.26.

Table 7.25 Urbansound8K dataset

Release date	2014
Total #	8732
Files format	.wav
Classes #	10
Classes (instances)	Air Conditioner (1000) Car Horns (429) Children Playing (1000) Dog Bark (1000) Drilling (1000) Engine Idling (1000) Gun Shot (374) Jackhammers (1000) Siren (929) Street Music (1000)
Files arrangement	10-folds
Sampling Rate	Different sampling rates of original recordings from www.freesound.org
File Duration	≤ 4 seconds

Table 7.26 Training complexity for the Urbansound8K dataset

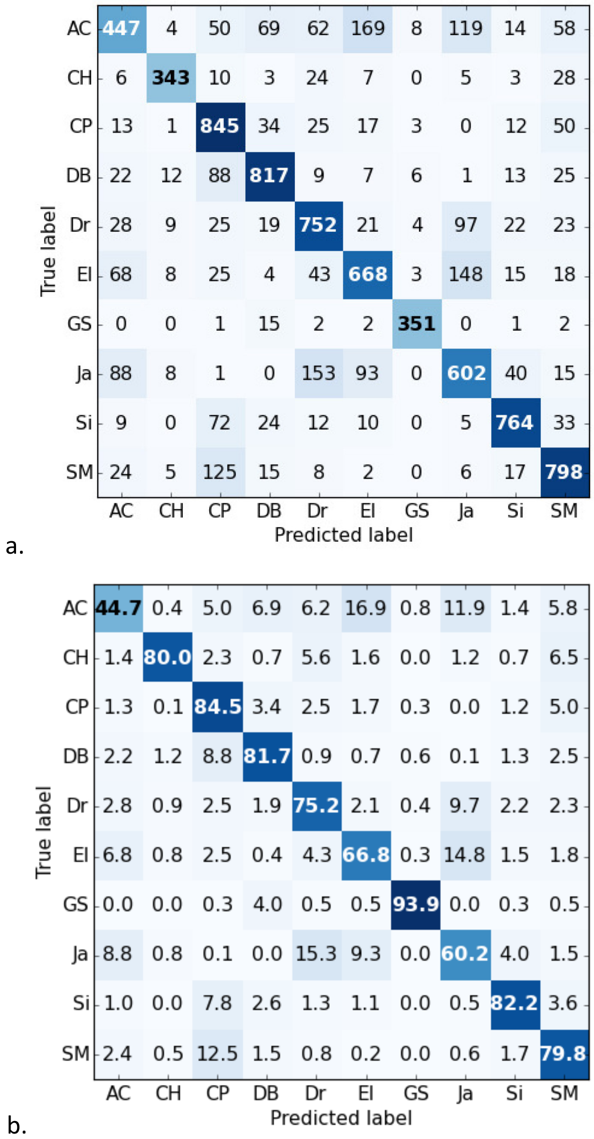
Model	Number of MCLNN layers (e = 300)	1
	Number of Dense layers	2
	Window size (frames at n = 15)	31
	MCLNN trainable parameters	1,116,300
	Total trainable parameters	1,173,010
Dataset split	Frames count per clip	170
	Features count per frame	120
	Training count (clips)	7,079
	Validation count (clips)	837
	Testing count (clips)	816

Input	Segment size (frames at $m = 1$, $k = 50$)	81
	Training count (segments)	318,555
	Validation count (segments)	36,720
	Testing count (segments)	37,665
	Batch size (segments)	500
Execution statistics	Average training time per fold (hours)	6.5
	Total training time (hours)	64.5
	Total memory (GB)	15

The MCLNN layers are followed by a global single dimensional mean pooling layer of extra frames $k = 5$ and two 100 neurons fully-connected layers with 50% dropout each. The segments extracted from the spectrogram used a step of 2, i.e. the overlapping frames between two successive segments = segment length – 2. The minibatch size was 500 samples per batch. The MCLNN achieved an accuracy of 74.37% using a shallow architecture and a long segment. Also, the MCLNN achieved 73.28% using a short segment in combination with a deep architecture as listed in Table 7.27. The highest non-neural based accuracy of 73.7% was in the work of Salamon et al. [78]. They applied an unsupervised learning attempt using Spherical k-means to establish a dictionary of PCA reduced features of a mel-scaled spectrogram with Random Forest as a classifier. Piczak-CNN [87] achieved an accuracy of 73.1%. Their model used 25 million parameters. Salamon et al. [16] used a deeper architecture than Piczak's with fewer parameters, and it achieved 73.0%. The MCLNN achieved accuracies that surpass the CNN architectures using approximately 1 million parameters for the shallow architecture and 3 million for the deep one. Additionally, the spectrogram transformation we applied is the same one used by Piczak-CNN.

Table 7.27 MCLNN performance on the Urbansound8K dataset compared with other attempts in the literature

Classifier and Features	Acc. %
MCLNN (Shallow, *k* = 51) + Mel-Spectrogram (This Work)	**74.37** ± **6.46**
Random Forest + Spherical K-Means + PCA + Mel-Spec.[78]	73.7
MCLNN (Deep, *k* = 5) + Mel-Spectrogram (This Work)	**73.28** ± **5.11**
Piczak-CNN + Mel-Spectrogram [87]	73.1
S&B-CNN + Mel-Spectrogram [16]	73.0
RBF-SVM + MFCC [24]	68.0



Classes: Air Conditioner(AC), Car Horns(CH), Children Playing(CP), Dog Bark(DB), Drilling(Dr), Engine Idling(EI), Gun Shot(GS), Jackhammers(Ja), Siren(Si) and Street Music(SM). a) actual count and b) normalized count in percentage.

Figure 7.9 Confusion matrix for the Urbansound8k dataset.

Figure 7.9 shows the confusion across the Urbansound8K categories. Low tonal components are effective in the confusion across Air Conditioner, Jackhammer, Drilling and Engine Idling categories. Similar findings were reported by Salamon et al. in [16].

7.8 YorNoise

YorNoise is a dataset focusing on vehicle sounds (road and rail traffic) that we collected in the city of York. The sound recordings were captured by a professional recorder (TASCAM DR-40) fitted on a tripod at the height of 1 m. The dataset was recorded at a sampling rate of 44 kHz with a mono channel, and a word depth of 16 bits. Recordings were taken in different locations near traffic movements, where vehicles were either speeding or shifting from stationary to the speed limit near bus stops. The traffic category contains all types of road vehicles irrespective of the size. The rail traffic was collected near the rail tracks outside the train station to avoid overlapping sound from passengers and the station loudspeaker announcements. The rail sounds were captured for trains of different sizes and types, e.g. cargo, passengers, etc., over tracks of varying distances from the microphone. The time taken by the longest train to pass across a microphone is approximately 40 seconds and the shortest 20 seconds. We listened to all the captured clips, which were 5 minutes each to validate the clips we were going to include in the dataset. Sounds of 4 seconds in duration were found to be sufficient for the human to distinguish environmental sounds as studied in [24]. Accordingly, all validated files were split into 4-second clips. We listened to all the 4 second files and dropped the files that were less than 4 seconds or silent ones. The filtered files were further redistributed into 10-folds. The distribution considered that the 4-second clips belonging to the same original 5 minutes file reside in the same fold to avoid contaminating the folds with sounds from the same origin and consequently biasing the testing accuracy. The dataset is publicly available[1]. The details of the dataset are listed in Table 7.28. The parameters used for the model, and the training complexity statistics are captured in Table 7.29.

Table 7.28 YorNoise dataset

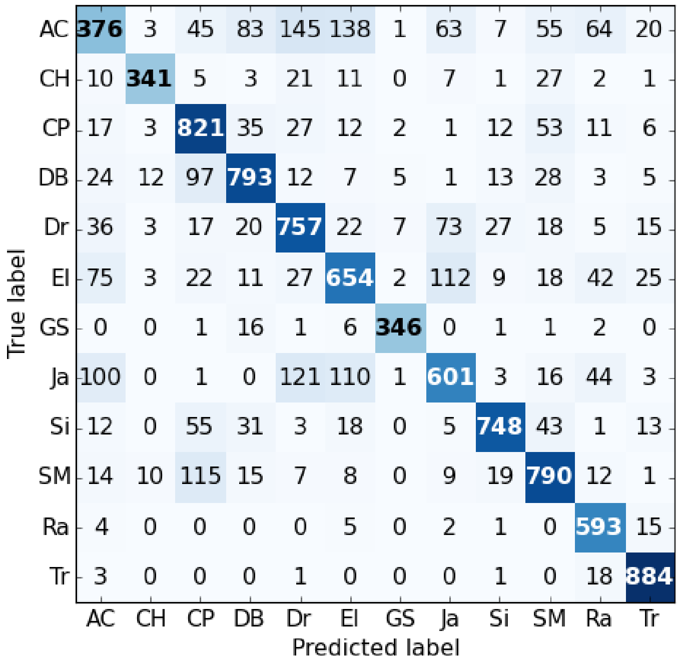
Release	Manually collected in the city of York
Total #	1527
Files format	.wav
Classes #	2
Classes (instances)	Rail (620) Traffic (907)
Files arrangement	10-folds
Sampling Rate	44100 Hz
File Duration	4 seconds

[1] https://github.com/fadymedhat/YorNoise

Table 7.29 Training complexity for the YorNoise dataset

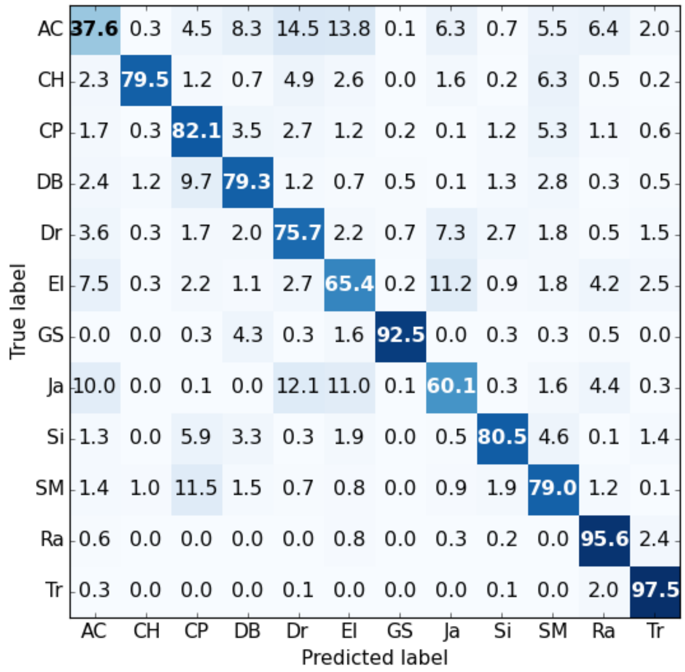
Model	Number of MCLNN layers (*e* = 300)	1
	Number of Dense layers	2
	Window size (frames at *n* = 15)	31
	MCLNN trainable parameters	1,116,300
	Total trainable parameters	1,173,212
Dataset split	Frames count per clip	170
	Features count per frame	120
	Training count (clips)	8,285
	Validation count (clips)	977
	Testing count (clips)	997
Input	Segment size (frames at *m* = 1, *k* = 50)	81
	Training count (segments)	372,825
	Validation count (segments)	43,965
	Testing count (segments)	44,865
	Batch size (segments)	500
Execution statistics	Average training time per fold (hours)	7
	Total training time (hours)	69.5
	Total memory (GB)	18

Since using the MCLNN on a two-class problem is not challenging, we appended the YorNoise dataset to the Urbansound8K, resulting in a total of 12 sound classes. We used the same model used for the Urbansound8k and spectrogram transformation described in the common section at the beginning of this chapter for the environmental sounds datasets. The MCLNN achieved a mean accuracy across 10-folds of 75.13% with a standard deviation of 5.02%. Figure 7.10 shows the confusion across the YorNoise and the Urbansound8k datasets. The YorNoise dataset was correctly categorized with 95.6% and 97.5% accuracy for the rail and traffic sound, respectively.


True label
Predicted label

	AC	CH	CP	DB	Dr	El	GS	Ja	Si	SM	Ra	Tr
AC	376	3	45	83	145	138	1	63	7	55	64	20
CH	10	341	5	3	21	11	0	7	1	27	2	1
CP	17	3	821	35	27	12	2	1	12	53	11	6
DB	24	12	97	793	12	7	5	1	13	28	3	5
Dr	36	3	17	20	757	22	7	73	27	18	5	15
El	75	3	22	11	27	654	2	112	9	18	42	25
GS	0	0	1	16	1	6	346	0	1	1	2	0
Ja	100	0	1	0	121	110	1	601	3	16	44	3
Si	12	0	55	31	3	18	0	5	748	43	1	13
SM	14	10	115	15	7	8	0	9	19	790	12	1
Ra	4	0	0	0	0	5	0	2	1	0	593	15
Tr	3	0	0	0	1	0	0	0	1	0	18	884

a.


True label
Predicted label

	AC	CH	CP	DB	Dr	El	GS	Ja	Si	SM	Ra	Tr
AC	37.6	0.3	4.5	8.3	14.5	13.8	0.1	6.3	0.7	5.5	6.4	2.0
CH	2.3	79.5	1.2	0.7	4.9	2.6	0.0	1.6	0.2	6.3	0.5	0.2
CP	1.7	0.3	82.1	3.5	2.7	1.2	0.2	0.1	1.2	5.3	1.1	0.6
DB	2.4	1.2	9.7	79.3	1.2	0.7	0.5	0.1	1.3	2.8	0.3	0.5
Dr	3.6	0.3	1.7	2.0	75.7	2.2	0.7	7.3	2.7	1.8	0.5	1.5
El	7.5	0.3	2.2	1.1	2.7	65.4	0.2	11.2	0.9	1.8	4.2	2.5
GS	0.0	0.0	0.3	4.3	0.3	1.6	92.5	0.0	0.3	0.3	0.5	0.0
Ja	10.0	0.0	0.1	0.0	12.1	11.0	0.1	60.1	0.3	1.6	4.4	0.3
Si	1.3	0.0	5.9	3.3	0.3	1.9	0.0	0.5	80.5	4.6	0.1	1.4
SM	1.4	1.0	11.5	1.5	0.7	0.8	0.0	0.9	1.9	79.0	1.2	0.1
Ra	0.6	0.0	0.0	0.0	0.0	0.8	0.0	0.3	0.2	0.0	95.6	2.4
Tr	0.3	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.0	2.0	97.5

b.

Classes: Air Conditioner(AC), Car Horns(CH), Children Playing(CP), Dog Bark(DB), Drilling(Dr), Engine Idling(EI), Gun Shot(GS), Jackhammers(Ja), Siren(Si), Street Music(SM), Rail (Ra) and Traffic (Tr). a) actual count and b) normalized count in percentage

Figure 7.10 Confusion matrix for the YorNoise and Urbansound8k

7.9 Summary

Through this chapter, we used several publicly available datasets used in benchmarking sound recognition models proposed in the literature. The models proposed in this thesis have shown an outperformance compared to other neural network based attempts including state-of-the-art Convolutional Neural Networks. The next chapter will provide further analysis with regard to the influence of different hyperparameters of the MCLNN and detailed comparison to Convolutional Neural Networks.

8
Analysis

The previously shown experiments have elaborated on the outperformance of the MCLNN compared to several neural network based architectures, in addition to the improved performance compared to several hand-crafted attempts and comparability to other methods applied to music genre classification and environmental sound recognition. The MCLNN has shown sustainability of the reported accuracies through extensive cross-validation experiments compared to the state-of-the-art methods reported in the literature as discussed throughout the GTZAN and the ISMIR2004 experiments. Through these datasets, we have shown the influence of the data split and the generalization of the MCLNN to datasets of different distributions. This is also evident in the environmental datasets using very similar MCLNN architectures with few differences of the tunable parameters among the environmental sound datasets (Urbansound8k, YorNoise, ESC-10 and ESC-50) and similarly for the music datasets (GTZAN, ISMIR2004, Ballroom and Homburg).

In this chapter, we investigate the impact of different hyperparameters used in the models proposed with the help of the ESC-10 dataset due to its moderate size, which allows the feasibility of this kind of evaluation. In the second section of the chapter, we present a set of unbiased performance comparison among a single layer of Masked Conditional, Conditional, Convolutional and Locally-Connected layers benchmarked by all the public datasets used in this work.

8.1 Hyperparameter Evaluation

In a different type of analysis, we used the ESC-10 dataset to investigate the effect of the mask and different hyperparameters introduced in the course of the CLNN and MCLNN architectures. We adapted the model referenced earlier in benchmarking the ESC-10 dataset as a baseline, and we gradually changed each individual parameter of the first layer in a 2-layered architecture, while fixing the other parameters to the baseline values without special fine-tuning of the step value. All experiments for the upcoming analysis are based on the mean value of 5-folds cross-validation totalling to 300 runs (60 × 5-folds).

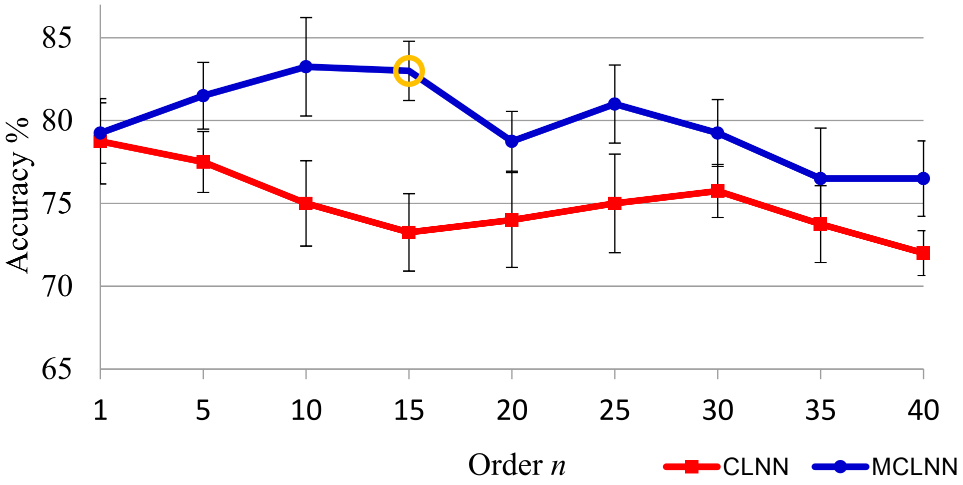


Figure 8.1 Accuracy and standard error on varying the Order n (baseline circled)

Figure 8.1 shows the effect of increasing the order n on the accuracy (in percentage terms) for a two-layered CLNN and MCLNN. The figure shows that on average the accuracy is directly proportional to the order n with regard to the MCLNN, where it decreases beyond $n = 15$. This is accounted for by the increase in the number of neurons together with the decrease in the number of training samples due to the increasing n since increasing n involves wider segments extracted from the spectrogram. The plot also shows the effect of the masking operation in the MCLNN compared to the accuracies achieved with a CLNN. The masking operation in the MCLNN boosted the accuracy at different values of n accounting for the properties, discussed previously, achieved by the mask.

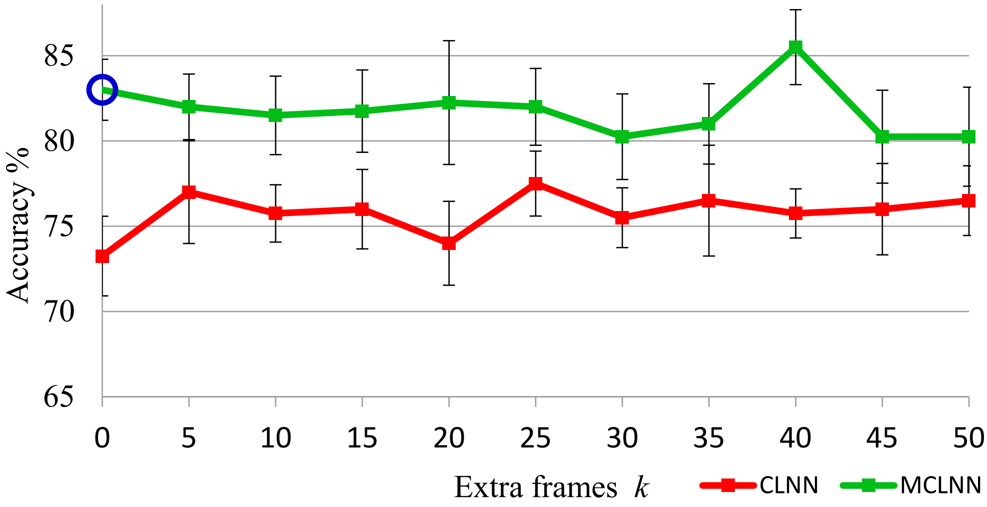


Figure 8.2 Accuracy and standard error on varying the Extra frames *k (baseline circled).*

Figure 8.2 shows the effect of the aggregation operation for both the CLNN and MCLNN. Still, the effect of the masking is clear in the accuracies of the MCLNN compared to the CLNN with a slight increase in the accuracy over the increasing *k*.

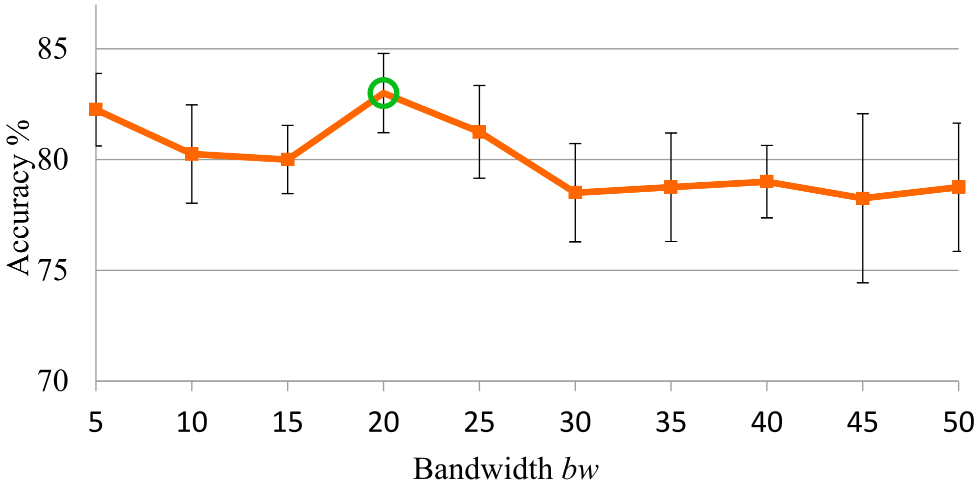


Figure 8.3 Accuracy and standard error on varying the Bandwidth *bw (baseline circled)*

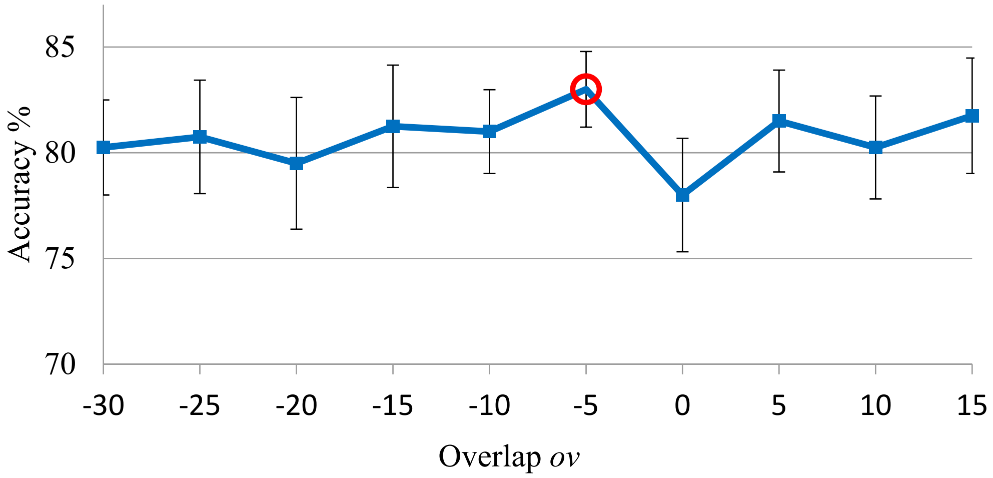


Figure 8.4 Accuracy and standard error on varying the Overlap *ov (baseline circled)*

Figure 8.3 and Figure 8.4 show the effect of varying the Bandwidth and the Overlap of the first layer of the MCLNN. In Figure 8.3, it is noticeable that increasing the bandwidth beyond *bw* = 20 causes a decrease in the accuracy. This is accounted to widening the scope of observation of a hidden node, which consequently prevents the node from learning about the distinctive features in a more focused region. On the contrary, decreasing the overlap, in Figure 8.4, using negative overlap values slightly increases the values with the increased sparseness, which suppresses the effect of the smearing of the energy across the frequency bins. This effect is depicted in the plot with the slight increase in the average accuracies of negative overlaps compared to positive ones.

8.2 Comparison to Convolutional Neural Networks

The intermediate signal representation and the general experimental settings have an influence, which could affect the overall performance of the reported accuracy. This presents a possibly biased result. In this section, we wanted to evaluate the performance of the MCLNN against Convolutional Neural Networks and its variant the Locally Connected Neural Networks (LCNN) [222], which are similar to CNN, but without weight sharing. We wanted to perform this evaluation in isolation from architectural, signal representation and hardware influences.

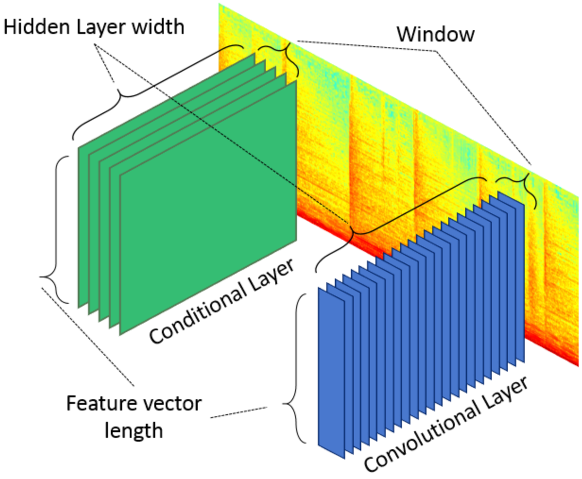


Figure 8.5 Conditional weight matrices scanning a spectrogram compared to the Convolutional filters

Figure 8.5 shows a single dimensional convolutional layer composed of a set of filters each having a length matching the length of the feature vector, mimicking the CLNN weight matrices in scanning the spectrogram. The figure shows the conditional layers of a CLNN and their corresponding convolutional layers. Despite both layers having the same number of weights, the conditional layer allows for independent training between the frames compared to the convolutional filters. The vector-matrix transformation, between an individual frame and the weight matrix, projects the frame in a different dimensional space matching the number of hidden nodes while preserving the projection spacing between successive projected vectors. On the contrary, in the CNN layer, the feature spacings across individual vectors is not preserved between successive frames projected into the feature-maps using the convolutional filters.

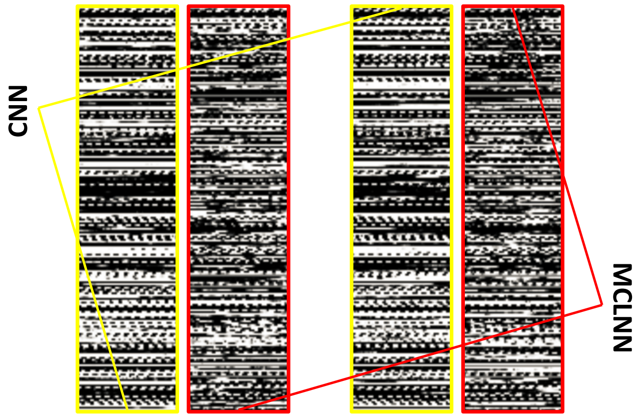


Figure 8.6 MCLNN generated segments compared to CNN

Figure 8.6 illustrates the output generated from an MCLNN and the corresponding output for the same input segments from a CNN. The pattern fluctuations shown in the MCLNN segments compared to the repetitive patterns appearing in the CNN pose that some features are left out by the CNN, which could be distinguishing properties to be used in classification.

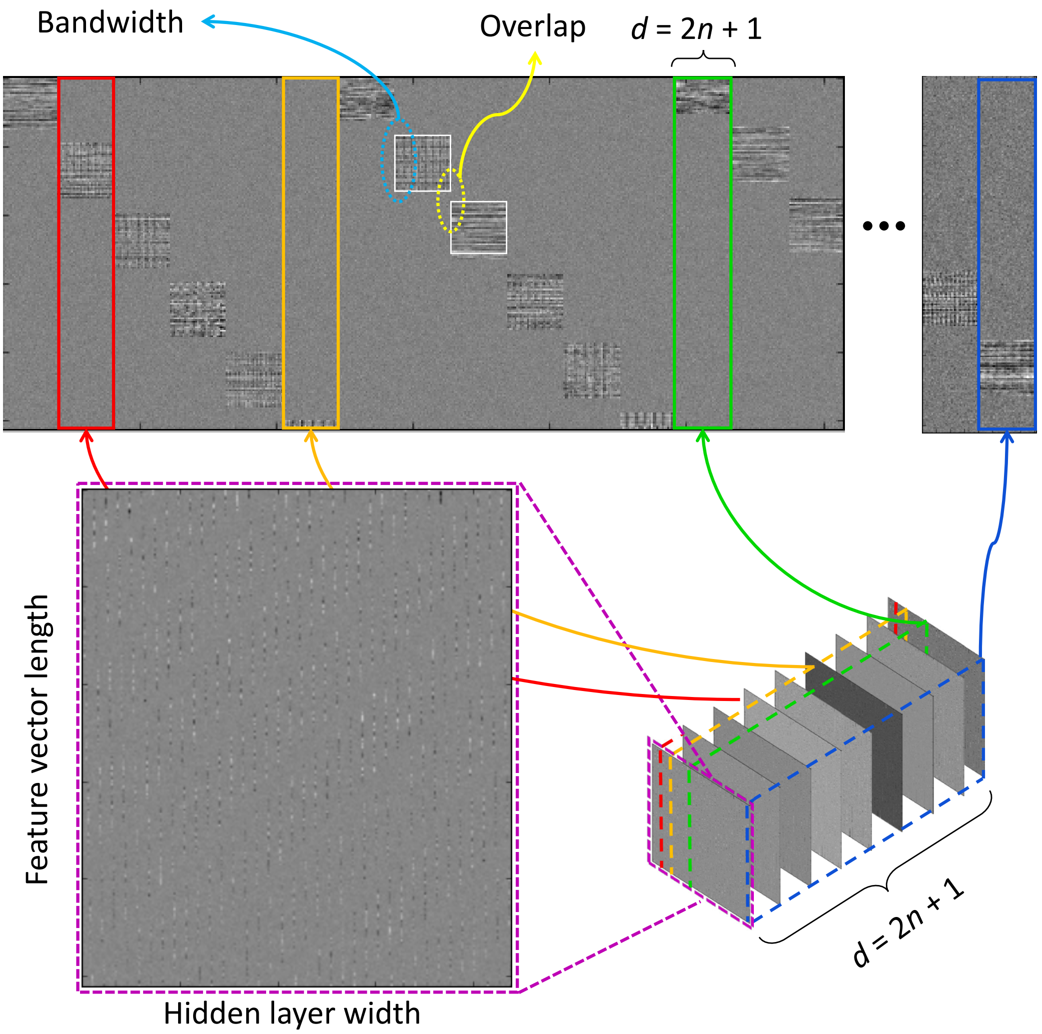


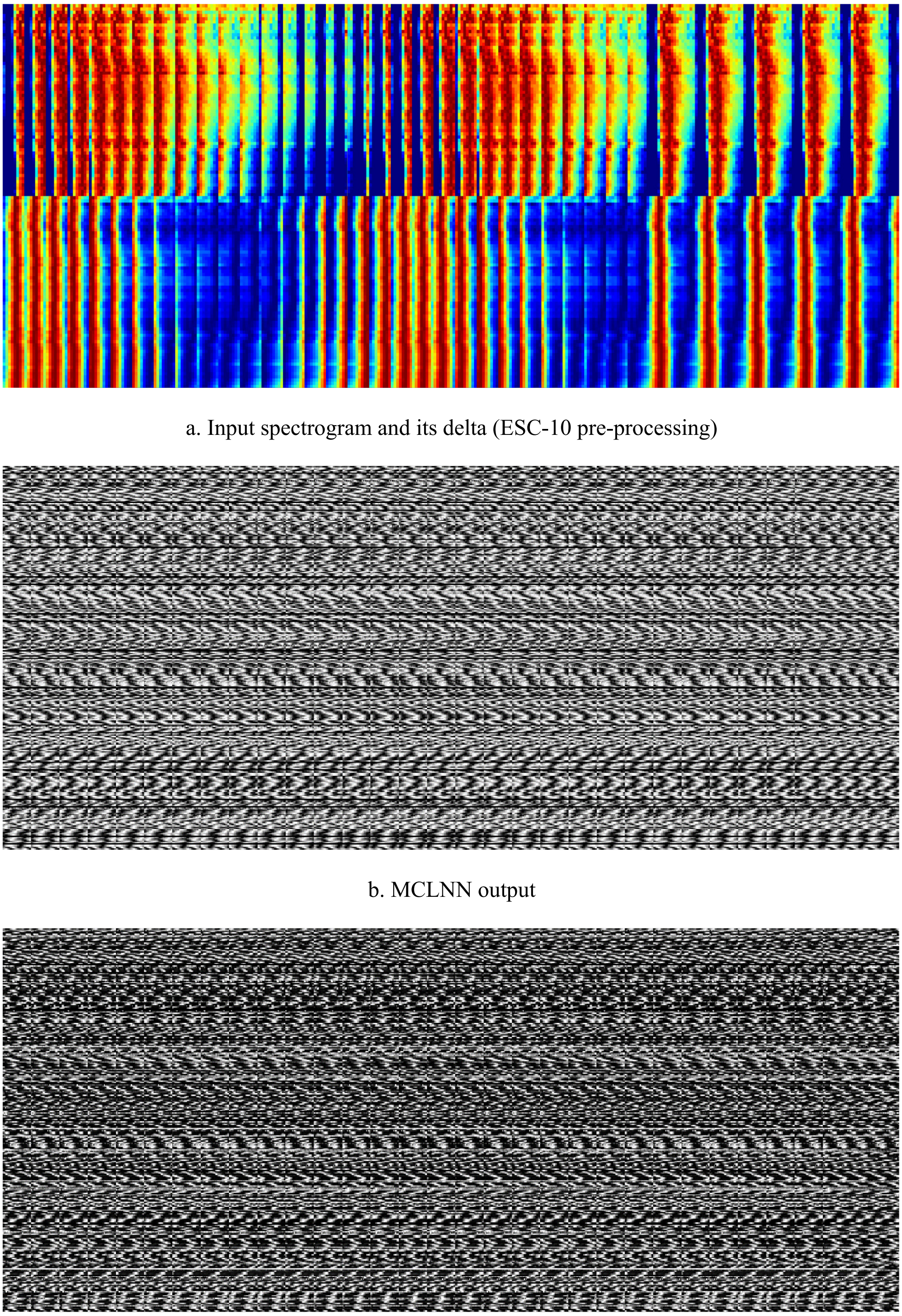
Figure 8.7 Learned conditional weight matrices

Figure 8.7 shows the learned weight matrices of an MCLNN of order n. There are d weight matrices following $d = 2n + 1$. Each of the d matrices is responsible for processing a single frame within the window of frames from a segment. The figure shows the learned pattern by a single weight matrix. The size of the matrix depends upon the feature vector length at the input and the number of neurons in the hidden layer. The pattern learned by each column represent the active connections to the corresponding hidden node from this specific matrix. The top section of the figure represents the concatenation of weight columns taken from different cross sections across the *d* matrices, where the figure depicts a complete view of the weights, represented by the highlighted rectangular slice, connected to each of the hidden nodes. The shift between the pattern learned in one slice and its neighbouring slices is due to the effect of the bandwidth and the overlap parameters of the mask. For each slice in the top section of the figure, the structured pattern appearing represents the trained weights and the random unstructured noise represents the disabled weights controlled by the mask.

The experiments in this section are applied using all the datasets used in the previous sections with their corresponding spectrogram representation used for either the music or the environmental sound classification tasks. The models involve a shallow architecture of each of the MCLNN, CLNN, CNN and LCNN followed by a pooling layer with the same *k* and *n* values used in each of the datasets relevant section discussed earlier with the absence of fully-connected layers, i.e. the output of the pooling layer is directly fed to a softmax layer. This section is not seeking to find the optimum architecture, but rather to provide an unbiased comparison of the accuracies between layers of different structure. The mean accuracy of 10/5-fold cross-validation for each dataset is reported in Table 8.1. The MCLNN achieved the highest accuracy across all the used datasets compared to the performance of the LCN and the CNN. Additionally, the CLNN achieved an accuracy that either surpasses or is comparable to a CNN and often surpasses an LCN, which shows to a certain extent an advantage in terms of performance in favour of the conditional structure of the CLNN and consequently the MCLNN. The table also lists the number of weights used by each layer for the different datasets in millions. The weights used by an MCLNN and a CLNN match exactly that of a CNN having filters of shapes proposed in Figure 8.5.

Table 8.1 Comparison of shallow architectures of MCLNN, CLNN, CNN, and LCN layers and the parameters used to the nearest million

	Urbansound8k	Ballroom	GTZAN	ISMIR	Homburg	ESC-10	ESC-50
MCLNN	**67.3**	**83.1**	**83.2**	**83.5**	**55.4**	**74.3**	**50.3**
CLNN	61.1	72.8	79.0	83.3	54.5	71.3	42.0
CNN	60.5	73.1	79.9	82.6	54.9	69.5	41.1
LCN	59.9	71.8	78.4	82.4	54.4	70.3	41.3
Parameter#							
LCN	57	129	5.6	3	1.2	57	42
MCLNN/CLNN/CNN	1	2	0.5	0.5	0.6	1	1



a. Input spectrogram and its delta (ESC-10 pre-processing)

b. MCLNN output

c. PRelu output

a) The input segments of the ESC-10 dataset, b) The output of the MCLNN in response to the input in (a), c) The output of the PRelu activations for the MCLNN output in (b).

Figure 8.8 Visualizations for 30 consecutive segments.

Figure 8.8 shows the MCLNN response for 30 consecutive input segments. The segments are overlapping for visualization purposes, but each segment has a width equal to $q - 2n$ and a height e, following the hidden layer node count.

8.3 Summary

This chapter presented an in-depth analysis of the effect of varying different hyperparameters of the models introduced in this work. Additionally, the chapter provided an evaluation of the proposed models in comparison to the Convolutional Neural Networks as a state-of-the-art model widely for image recognition and adapted to the sound recognition problem.

9

Conclusions and Future Work

SOUND recognition is still an open research problem either from the signal processing and intermediate representation or the point of view of pattern recognition and machine learning. The literature has considered the two folds of the problem in an endeavour to provide systems capable of distinguishing and classifying sound. Despite these attempts, we have not reached the stage of creating an artificial human-like hearing ability, and researchers approached the problem by tailoring the methods being introduced in relation to the type of sound being processed. Speech phonemes inspired most of the work in signal processing and pattern recognition models especially the use of Gaussian Mixture Model and Hidden Markov Model statistical combination. Similarly, in music genre recognition, cues special to music such as Timbre, Rhythm, and other musical perceptual properties were the driver in advancing methods targeted for this problem. Environmental sounds have been considered as well, exploiting methods devised for speech and music to fulfil application requirements such as surveillance, hearing impairments aids, and noise control to name a few.

The intermediate signal representation of a sound signal has been considered as a handcrafting problem, where a variety of techniques have been proposed to extract features in either the time-domain or the frequency-domain representation of a signal. The process of feature extraction is usually an exhaustive manual process in terms of engineering the feature combination that can enhance the performance of a pattern recognition model. Automating the feature extraction process will facilitate developing generic models that can operate on a signal whatever its type without the need to handcraft the features to fit a specific nature related to the signal under consideration.

Neural networks are pattern recognition models that have been around for years. They have primarily been used in classification with the help of handcrafted features introduced to them. Recent breakthrough attempts in neural networks have managed to use deep architectures of these models for hard problems in image recognition. The success of these deep architectures was not only in their use as classification models but in their ability to extract the features from raw images without using special handcrafted ones. Since the remarkable results they achieved in image recognition, they have been applied to a range of pattern recognition problems aiming to exploit their capabilities in automatically distinguishing relevant features. Despite the success of these models in image recognition, they have not achieved similar success for sound, especially when compared to handcrafted approaches.

The neural network attempts are usually adapted to the sound problem after they achieve success in other fields such as image recognition. Since they are not designed to exploit the sound nature, they may not optimally harness sound related properties. In this thesis, we have introduced a new neural network model for a multi-channel temporal signal recognition such as sound. The model we are proposing in this work is designed to consider both the temporal and the spectral aspect of the signal exploiting the spectrogram of a sound signal as an intermediate representation.

We have introduced the ConditionaL Neural Network (CLNN) and its extension the Masked ConditionaL Neural Network (MCLNN). The CLNN considers the temporal correlation across the frames of a temporal signal, which ensures that the influence of frames in proximity to each other is collectively taken into consideration rather than a bag-of-frame classification, where each frame is considered in isolation from its neighbours. The MCLNN extends the CLNN using a binary mask that embeds a systematic sparseness over the connections of the network. The enforced sparseness, extending from the filterbank design used in signal processing, allows embedding a filterbank-like behaviour inside the network. Filterbanks have been used widely in signal analysis to allow aggregating different ranges of frequency components of a signal. A filterbank is used to subdivide the frequency domain to provide another spectrogram representation that has a lower number of dimensions, matching the number of filters used, compared to a raw spectrogram. Additionally, the filterbank design depends upon the bandwidth of each filter and the overlapping bands between filters. Similar parameters are used in the design of the mask. In addition to the role of the mask in controlling the active connections, mimicking a filterbank, it automates the exploration of several feature combinations concurrently during training, using several shifted versions of the filterbank-like pattern.

Chapter 2 highlighted research efforts in sound recognition either in signal processing or machine learning disregarding the type of sound, i.e. speech, music and environmental sounds. The chapter attempted to refer to landmark attempts in both fields that started around seven decades ago or even more.

Chapter 3 explored Spectrograms and Scalegrams as examples of widely used two-dimensional representations of sound signals. The signal pre-processing is an important stage. The chapter listed some of the widely used methods, for example, dimensionality reduction and data standardization.

Chapter 4 presented some of the pattern recognition models used for temporal signals especially the combination of the Gaussian Mixture Model and the Hidden Markov Model used widely in speech recognition.

Chapter 5 demonstrated an in-depth analysis of neural networks, and it highlighted neural network based architectures relevant to this work. The chapter referred to advances in neural networks and their applications in automating the feature extraction stage, which aims to eliminate the need to handcraft the features required by a recognition model.

Chapter 6 explained the contribution of this work. The ConditionaL Neural Network (CLNN), a new model for sound recognition, that takes into consideration the temporal correlation across the frames of a sound signal. The CLNN is the platform upon which the Masked Conditional Neural Network (MCLNN) has been introduced. The MCLNN exploits the time-frequency representation of the signal by enforcing a systematic sparseness that follows a filterbank-like pattern.

Chapter 7 through an extensive set of experiments, has shown that the MCLNN without any special rhythmic or timbral analysis, especially for music datasets, sustains accuracies that surpass neural network based and hand-crafted feature extraction methods and comparable to others. Meanwhile, MCLNN still preserves the generalization that allows it to be adapted for any other multi-channel temporal representations. Through the datasets used in the experiments, we have shown the influence of the data split and the generalization of the MCLNN to datasets of different distributions. This is also evident in using very similar MCLNN architectures with few differences of the tuneable parameters across the datasets.

Chapter 8 provided an in-depth analysis to evaluate the effect of varying the MCLNN hyperparameters. Additionally, the chapter considered an unbiased comparison between the MCLNN and a Convolutional Neural Network (CNN) of a similar architecture without changing any of the hyperparameters of the model, e.g. learning rate, optimization function,…,etc. The comparison has been applied over all the datasets considered in this work, where the MCLNN surpassed the performance of an equivalent layer of a CNN over all the datasets.

Future work

The Masked ConditionaL Neural Network (MCLNN) achieves accuracies that surpass widely used models based on the state-of-the-art Convolutional Neural Networks and comparable to hand-crafted features using several publicly available datasets. The MCLNN has achieved these accuracies without any special handling related to the signal's nature, especially for musical signals, where several reported methods exploited musical perceptual properties to enhance the classification decision. The ability of the MCLNN to achieve this performance without any special pre-processing allows for the consideration of MCLNN for other multi-channel temporal signal representations, which we will consider for future work. Additionally, will include optimizing the mask patterns, considering different combinations of the order *n*, used to control the number of successive frames to be considered concurrently, across the layers and using MCLNN as a stand-alone feature extractor for other pattern analysis tasks. We will also consider applying MCLNN to other signals possessing a temporal nature. Similar to a Convolutional Recurrent Neural Network, merging the MCLNN to extract the local feature with the long-term dependencies captured by an LSTM will be explored. In addition to investigating the application of the masking behaviour in combination with other neural networks models such as the CNN.

Glossary

AdaBoost	Adaptive Boosting
ASR	Automatic Speech Recognition
BM	Boltzmann Machine
BoF	Bag of Frames
BPM	Beat per Minute
BPTT	Back Propagation Through Time
CASA	Computational Auditory Scene Analysis
CD	Contrastive Divergence
CE	Cross Entropy
CG	Conjugate Gradient
CLNN	ConditionaL Neural Network
CNN	Convolutional Neural Network
Conv-DBN	Convolutional Deep Belief Net
Conv-RBM	Convolutional Restricted Boltzmann Machine
CRBM	Conditional Restricted Boltzmann Machine
CRF	Conditional Random Field
CWT	Continuous Wavelet Transform
DBN	Deep Belief Net

DCT	Discrete Cosine Transform
DFT	Discrete Fourier transform
DWT	Discrete Wavelet Transform
ELU	Exponential Linear Unit
EM	Expectation Maximization
END	Environmental Noise Directive
ESR	Environmental Sound Recognition
FCRBM	Factored Conditional Restricted Boltzmann Machine
FFT	Fast Fourier Transform
FWT	Fast Wavelet Transform
GD	Gradient Descent
GMM	Gaussian Mixture Model
GPU	Graphical Processing Unit
HFCC	Human Factor Cepstral Coefficients
HMM	Hidden Markov Model
ICA	Independent Component Analysis
ICRBM	Interpolating Conditional Restricted Boltzmann Machine
KNN	K-Nearest Neighbours
LCNN	Locally Connected Neural Network
LDA	Linear Discriminant Analysis
LDB	Local Discriminant Bases
LPC	Linear Predictive Coding
LPCC	Linear Predictive Cepstral Coefficients
LSTM	Long Short-Term Memory

MCLNN	Masked ConditionaL Neural Network
mcRBM	mean-covariance Restricted Boltzmann Machine
MFCC	Mel-Frequency Cepstral Coefficients
MIR	Music Information Retrieval
MLE	Maximum Likelihood Estimation
MLP	Multi-layer perceptron
MP	Matching Pursuits
MRF	Markov Random Fields
MSE	Mean Square Error
NMF	Non-Negative Matrix Factorization
PCA	Principal Component Analysis
PLPCC	Perceptual Linear Predictive cepstral coefficients
Prelu	parametric rectified linear unit
PSD	Predictive Sparse Decomposition
RASTA-PLP	Relative Spectral Transform - Perceptual Linear Prediction
RBF-SVM	Radial Basis Function Support Vector Machine
RBM	Restricted Boltzmann Machine
Relu	rectified linear unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SNR	Signal to Noise Ratio
SOM	Self-organizing Map
STFT	Short Time Fourier Transform
SVM	Support Vector Machine

VQ	Vector Quantization
ZC	Zero Crossings

References

[1] J. R. Barber, K. A. Warner, D. M. Theobald, C. L. Burdett, C. Formichella, K. M. Fristrup*, et al.*, "Anthropogenic Noise Exposure In Protected Natural Areas: Estimating The Scale Of Ecological Consequences," *Landscape Ecology,* vol. 26, pp. 1281-1295, 2011.

[2] M. S. Hammer, T. K. Swinburn, and R. L. Neitzel, "Environmental Noise Pollution in the United States: Developing an Effective Public Health Response," *Environmental Health Perspectives,* vol. 122, pp. 115-9, Feb 2014.

[3] U. Kraus, A. Schneider, S. Breitner, R. Hampel, R. Ruckerl, M. Pitz*, et al.*, "Individual Daytime Noise Exposure During Routine Activities and Heart Rate Variability in Adults: A Repeated Measures Study," *Environmental Health Perspectives,* vol. 121, pp. 607-12, May 2013.

[4] E. E. M. M. v. Kempen, H. Kruize, H. C. Boshuizen, C. B. Ameling, B. A. M. Staatsen, and A. E. M. d. Hollander, "The Association between Noise Exposure and Blood Pressure and Ischemic Heart Disease: A Meta-analysis," *Environmental Health Perspectives,* vol. 110, pp. 307-317, 2002.

[5] U. Landstrom, E. Akerlund, A. Kjellberg, and M. Tesarz, "Exposure Levels, Tonal Components, and Noise Annoyance in Working Environments," *Environment International,* vol. 21, pp. 265-275, 1995.

[6] H. Leventhall, "Low frequency noise and annoyance," *Noise and Health,* vol. 6, pp. 59-72, 2004.

[7] S. Haykin, *Neural Networks: A Comprehensive Foundation*: Pearson, 1997.

[8] R. J. Williams and D. Zipser, "Gradient-based Learning Algorithms for Recurrent Networks and Their Computational Complexity," in *Backpropagation*, Y. Chauvin and D. E. Rumelhart, Eds., ed: L. Erlbaum Associates Inc., 1995, pp. 433-486.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput,* vol. 9, pp. 1735-80, Nov 15 1997.

[10] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," presented at the Advances in Neural Information Processing Systems (NIPS), 2009.

[11] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech Recognition With Deep Recurrent Neural Networks," in *International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2013.

[12] A. Graves, S. Fernandez, and J. Schmidhuber, "Multi-Dimensional Recurrent Neural Networks," in *International Conference on Artificial Neural Networks (ICANN)*, 2007.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Neural Information Processing Systems, NIPS*, 2012.

[14] C. Szegedy, L. Wei, J. Yangqing, P. Sermanet, S. Reed, D. Anguelov*, et al.*, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2015, pp. 1-9.

[15] C. Kereliuk, B. L. Sturm, and J. Larsen, "Deep Learning and Music Adversaries," *IEEE Transactions on Multimedia,* vol. 17, pp. 2059-2071, 2015.

[16] J. Salamon and J. P. Bello, "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," *IEEE Signal Processing Letters,* 2016.

[17] L. Hertel, H. Phan, and A. Mertins, "Comparing Time and Frequency Domain for Audio Event Recognition using Deep Learning," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2016.

[18] P. Barros, C. Weber, and S. Wermter, "Learning Auditory Neural Representations for Emotion Recognition," in *IEEE International Joint Conference on Neural Networks (IJCNN/WCCI)*, 2016.

[19] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Royal Statistical Society,* vol. 39, pp. 1–38, 1977.

[20] L. E. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains," *The Annals of Mathematical Statistics,* vol. 37, pp. 1554-1563, 1966.

[21] G. D. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE,* vol. 61, pp. 268-278, 1973.

[22] L. Rabiner and B. Juang, "An Introduction To Hidden Markov Models," *IEEE ASSP Magazine,* vol. 3, pp. 4-16, 1986.

[23] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *ACM International Conference on Multimedia* 2015, pp. 1015-1018.

[24] J. Salamon, C. Jacoby, and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research," in *Proceedings of the 22nd ACM International Conference on Multimedia*, Orlando, USA, 2014, pp. 1041-1044.

[25] K. H. Davis, R. Biddulph, and S. Balashek, "Automatic Recognition of Spoken Digits," *Journal of the Acoustical Society of America,* vol. 24, pp. 637-642, 1952.

[26] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation,* vol. 19, pp. 297-301, 1965.

[27] G. McLachlan and D. Peel, *Finite Mixture Models*: John Wiley & Sons, Inc. , 2000.

[28] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE,* vol. 77, pp. 257 - 286, 1989.

[29] X. He and L. Deng, "Discriminative Learning for Speech Recognition: Theory and Practice," *Synthesis Lectures on Speech and Audio Processing,* vol. 4, pp. 1-112, 2008.

[30] L. Deng and X. Li, "Machine Learning Paradigms for Speech Recognition: An Overview," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 21, pp. 1060-1089, 2013.

[31] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-Based Music Information Retrieval: Current Directions and Future Challenges," *Proceedings of the IEEE,* vol. 96, pp. 668-696, 2008.

[32] E. Wold, T. Blum, D. Keislar, and J. Wheaten, "Content-based classification, search, and retrieval of audio," *IEEE Multimedia,* vol. 3, pp. 27-36, 1996.

[33] H. Soltau, T. Schultz, MartinWestphal, and A. Waibel, "Recognition Of Music Types," in *International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 1998.

[34] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the fifth annual workshop on Computational Learning Theory, COLT*, 1992.

[35] M. I. Mandel and D. P. W. Ellis, "Song-Level Features And Support Vector Machines For Music Classification," in *International Conference on Music Information Retrieval, ISMIR*, 2005.

[36] G. Tzanetakis, G. Essl, and P. Cook, "Automatic Musical Genre Classification Of Audio Signals," in *International Symposium on Music Information Retrieval, ISMIR*, 2001.

[37] G. Tzanetakis and P. Cook, "Musical Genre Classification of Audio Signals," *IEEE Transactions On Speech And Audio Processing* vol. 10, 2002.

[38] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl, "Aggregate Features And AdaBoost For Music Classification," *Machine Learning,* vol. 65, pp. 473-484, 2006.

[39] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences,* vol. 55, pp. 119-139, 1997.

[40] R. Cruz, A. Ortiz, A. M. Barbancho, and I. Barbancho, "Unsupervised Classification of Audio Signals by Self-Organizing Maps and Bayesian Labeling," *Lecture Notes in Computer Science: Hybrid Artificial Intelligent Systems,* vol. 7208, pp. 61-70, 2012.

[41] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics,* vol. 43, pp. 59-69, 1982.

[42] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "A Survey of Audio-Based Music Classification and Annotation," *IEEE Transactions On Multimedia,* vol. 13, 2011.

[43] A. Holzapfel and Y. Stylianou, "Musical Genre Classification using Nonnegative Matrix Factorization-Based Features," *IEEE Transactions on Audio Speech and Language Processing,* vol. 16, pp. 424-434, Feb 2008.

[44] J. J. Aucouturier, B. Defreville, and F. Pachet, "The Bag-Of-Frames Approach to Audio Pattern Recognition: A Sufficient Model for Urban Soundscapes but not for Polyphonic Music," *Journal of the Acoustical Society of America,* vol. 122, pp. 881-91, Aug 2007.

[45] I. Panagakis, E. Benetos, and C. Kotropoulos, "Music Genre Classification: A Multilinear Approach," in *International Society for Music Information Retrieval, ISMIR*, 2008.

[46] Y. Panagakis, C. Kotropoulos, and G. R. Arce, "Music Genre Classification using Locality Preserving Non-Negative Tensor Factorization and Sparse Representations," in *International Society for Music Information Retrieval Conference, ISMIR*, 2009.

[47] J. Anden and S. Mallat, "Deep Scattering Spectrum," *IEEE Transactions on Signal Processing,* vol. 62, pp. 4114-4128, 2014.

[48] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun, "Unsupervised Learning Of Sparse Features For Scalable Audio Classification," in *International Society for Music Information Retrieval, ISMIR*, 2011.

[49] K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, "Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition," Cornell University Library, arxiv, 2008.

[50] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science,* vol. 313, pp. 504-7, Jul 28 2006.

[51] P. Hamel and D. Eck, "Learning Features From Music Audio With Deep Belief Networks," in *International Society for Music Information Retrieval Conference, ISMIR*, 2010.

[52] H. Lee, Y. Largman, P. Pham, and A. Y. Ng, "Unsupervised Feature Learning for Audio Classification using Convolutional Deep Belief Networks," in *Neural Information Processing Systems (NIPS)*, 2009.

[53] S. Dieleman and B. Schrauwen, "End-To-End Learning For Music Audio," in *International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2014.

[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, pp. 2278-2324, 1998.

[55] A. v. d. Oord, S. Dieleman, and B. Schrauwen, "Deep Content-Based Music Recommendation," in *Neural Information Processing Systems, NIPS*, 2013.

[56] J. Pons, T. Lidy, and X. Serra, "Experimenting with Musically Motivated Convolutional Neural Networks," in *International Workshop on Content-based Multimedia Indexing, CBMI*, 2016.

[57] The European Parliament and of the Council, "Directive 2002/49/EC of the European Parliament and of the Council relating to the Assessment and Management of Environmental Noise," 2002.

[58] A. Dufaux, L. Besacier, M. Ansorge, and F. Pellandini, "Automatic Sound Detection and Recognition for Noisy Environment," in *European Signal Processing Conference (EUSIPCO)*, 2000.

[59] L. Couvreura and M. Laniray, "Automatic Noise Recognition in Urban Environments Based on Artificial Neural Networks and Hidden Markov Models," in *International Congress and Exposition on Noise Control Engineering (INTER-NOISE)*, 2004.

[60] M. Cristani, M. Bicego, and V. Murino, "Audio-Visual Event Recognition in Surveillance Video Sequences," *IEEE Transactions on Multimedia,* vol. 9, pp. 257-267, 2007.

[61] D. Chesmore and J. Schofield, "Acoustic Detection of Regulated Pests in Hardwood Material," *European and Mediterranean Plant Protection Organization Bulletin (EPPO),* vol. 40, pp. 46-51, 2010.

[62] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards Robotic Assistants in Nursing homes: Challenges and Results," *Robotics and Autonomous Systems,* vol. 42, pp. 271-281, 2003.

[63] M. Popescu and A. Mahnot, "Acoustic Fall Detection Using One-Class Classifiers," in *Annual International Conference of the Engineering in Medicine and Biology Society, EMBC*, 2009, pp. 2-6.

[64] R. S. Goldhor, "Recognition of Environmental Sounds," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1993, pp. 149-152 vol.1.

[65] P. Gaunard, C. G. Mubikangiey, C. Couvreur, and V. Fontaine, "Automatic Classification Of Environmental Noise Events By Hidden Markov Models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998, pp. 3609-3612.

[66] D. O'Shaughnessy, "Linear predictive coding," *IEEE Potentials,* vol. 7, pp. 29-32, 1988.

[67] T. Zhang and C.-C. J. Kuo, "Hierarchical Classification Of Audio Data For Archiving And Retrieving," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1999, pp. 3001-3004.

[68] S. Akhtar, M. Elshafei-Ahmed, and M. S. Ahmed, "Detection of Helicopters Using Neural Nets," *IEEE Transactions on Instrumentation and Measurement,* vol. 50, pp. 749 - 756, 2001.

[69] E. D. Chesmore, "Application of Time Domain Signal Coding and Artificial Neural Networks to Passive Acoustical Identification of Animals," *Applied Acoustics,* vol. 62, p. 16, 2001.

[70] M. Cowling and R. Sitte, "Comparison of Techniques for Environmental Sound Recognition," *Pattern Recognition Letters,* vol. 24, pp. 2895-2907, 2003.

[71] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fagerlund, T. Sorsa*, et al.*, "Audio-Based Context Recognition," *IEEE Transactions on Audio, Speech and Language Processing,* vol. 14, pp. 321-329, 2006.

[72] F. Su, L. Yang, T. Lu, and G. Wang, "Environmental Sound Classification for Scene Recognition using Local Discriminant Bases and HMM," in *International Conference on Multimedia (MM)*, 2011, p. 1389.

[73] N. Saito and R. R. Coifman, "Local Discriminant Bases and their Applications," *Journal of Mathematical Imaging and Vision,* vol. 5, pp. 337-358, 1995.

[74] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-Dependent Sound Event Detection," *Journal on Audio Speech and Music Processing (Eurasip),* Jan 9 2013.

[75] S. Chu, S. Narayanan, and C.-C. J. Kuo, "Environmental Sound Recognition with Time-Frequency Audio Features," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 17, pp. 1142-1158, 2009.

[76] S. G. Mallat and Z. Zhang, "Matching Pursuits with Time-Frequency Dictionaries," *IEEE Transactions on Signal Processing,* vol. 41, pp. 3397-3415, 1993.

[77] O. Dikmen and A. Mesaros, "Sound Event Detection using Non-Negative Dictionaries Learned from Annotated Overlapping Events," in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013.

[78] J. Salamon and J. P. Bello, "Unsupervised Feature Learning for Urban Sound Classification," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015.

[79] I. S. Dhillon and D. S. Modha, "Concept Decompositions for Large Sparse Text Data Using Clustering," in *Machine Learning*. vol. 42, ed, 2001, pp. 143-175.

[80] J. Salamon and J. P. Bello, "Feature Learning With Deep Scattering for Urban Sound Analysis," in *European Signal Processing Conference (EUSIPCO)*, 2015.

[81] G. Wichern, J. Xue, H. Thornburg, B. Mechtley, and A. Spanias, "Segmentation, Indexing, and Retrieval for Environmental and Natural Sounds," *IEEE*

Transactions on Audio, Speech, and Language Processing, vol. 18, pp. 688-707, 2010.

[82] S. Chachada and C.-C. J. Kuo, "Environmental Sound Recognition: A Survey," in *Asia-Pacific Signal and Information Processing Association (APSIPA)*, 2013.

[83] S. P. Mohanapriya, E. P. Sumesh, and R. Karthika, "Environmental Sound Recognition Using Gaussian Mixture Model And Neural Network Classifier," in *International Conference on Green Computing Communication and Electrical Engineering, ICGCCEE*, 2014, pp. 1-5.

[84] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and Classification of Acoustic Scenes and Events," *IEEE Transactions on Multimedia,* vol. 17, pp. 1733-1746, 2015.

[85] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic Sound Event Detection using Multi Label Deep Neural Networks," in *International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1-7.

[86] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional Neural Networks for Speech Recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing,* vol. 22, pp. 1533-1545, Oct 2014.

[87] K. J. Piczak, "Environmental Sound Classification with Convolutional Neural Networks," in *IEEE international workshop on Machine Learning for Signal Processing (MLSP)*, 2015.

[88] T. Li, M. Ogihara, and Q. Li, "A Comparative Study on Content-Based Music Genre Classification," in *ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*, 2003.

[89] T. Li and G. Tzanetakis, "Factors in Automatic Musical Genre Classification of Audio Signals," in *IEEE workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[90] T. Lidy and A. Rauber, "Evaluation Of Feature Extractors And Psycho-Acoustic Transformations For Music Genre Classification," in *International Conference on Music Information Retrieval, ISMIR*, 2005.

[91] E. Pampalk, A. Flexer, and G. Widmer, "Improvements Of Audio-Based Music Similarity And Genre Classification," in *International Conference on Music Information Retrieval, ISMIR*, 2005.

[92] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic genre classification of music content: a survey," *IEEE Signal Processing Magazine,* vol. 23, pp. 133-141, 2005.

[93] T. Lidy, A. Rauber, A. Pertusa, and J. M. Inesta, "Improving Genre Classification By Combination Of Audio And Symbolic Descriptors Using A Transcription System," in *International Conference on Music Information Retrieval*, 2007.

[94] G. Peeters, "Spectral and Temporal Periodicity Representations of Rhythm for the Automatic Classification of Music Audio Signal," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 19, pp. 1242-1252, 2011.

[95] M. R. Ito and R. W. Donaldson, "Zero-Crossing Measurements for Analysis and Recognition of Speech Sounds," *IEEE Transactions On Audio And Electroacoustics,* vol. 19, pp. 235 - 242, 1971.

[96] S. K. Park, R. M. Kil, Y.-G. Jung, and M.-S. Han, "Zero-Crossing-Based Feature Extraction for Voice Command Systems Using Neck-Microphones," in *Lecture Notes in Computer Science*. vol. 4491, ed: Springer Berlin Heidelberg, 2007, pp. 1318-1326.

[97] M. Ghulam, J. Horikawa, and T. Nitta, "A Pitch-Synchronous Peak-Amplitude Based Feature Extraction Method For Noise Robust ASR," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, Toulouse, 2006.

[98] R. Gubka and M. Kuba, "A Comparison Of Audio Features For Elementary Sound Based Audio Classification," in *International Conference on Digital Technologies (DT)*, 2013.

[99] W. Zhu, Y. Wang, and Q.-F. Zhu, "Second-Order Derivative-Based Smoothness Measure for Error Concealment in DCT-Based Codecs," *IEEE Transactions On Circuits And Systems For Video Technology,* vol. 8, pp. 713 - 718, 1998.

[100] K. P. Balanda and H. L. MacGillivray, "Kurtosis: A Critical Review," *The American Statistician,* vol. 42, pp. 111-119, 1988.

[101] S. Busson, C. Gervaise, A. Barazzutti, B. Kinda, V. Jaud, L. Chauvaud*, et al.*, "Higher-order statistics for bioacoustic click detection," in *10`eme Congres Francais d'Acoustique*, Lyon, 2010.

[102] D. P. W. Ellis, X. Zeng, and J. H. McDermott, "Classifying Soundtracks With Audio Texture Features," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, Prague, 2011, pp. 5880–5883.

[103] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE,* vol. 66, pp. 51-83, 1978.

[104] W. Chu and B. Champagne, "A Noise-Robust FFT-Based Auditory Spectrum With Application in Audio Classification," *IEEE Transactions On Audio, Speech, And Language Processing,* vol. 16, pp. 137-150, 2008.

[105] B. Boashash, *Time-Frequency Signal Analysis and Processing – A Comprehensive Reference*. Oxford: Elsevier, 2003.

[106] S. S. Stevens, J. Volkmann, and E. B. Newman, "A Scale for the Measurement of the Psychological Magnitude Pitch," *Acoustical Society of America,* vol. 8, p. 185, 1937.

[107] P. M. P1, S. B. Yaacob, A. Nazri, and S. Kumar, "Classification of Vowel Sounds Using MFCC and Feed Forward Neural Network," in *Signal Processing & Its Applications, CSPA*, Kuala Lumpur, 2009, pp. 59 - 62.

[108] T. Kinnunen, R. Saeidi, Filip Sedlák, Kong Aik Lee, J. Sandberg, M. Hansson-Sandsten*, et al.*, "Low-Variance Multitaper MFCC Features: A Case Study in Robust Speaker Verificatio," *IEEE Transactions On Audio, Speech, And Language Processing,* vol. 20, pp. 1990 - 2001, 2012.

[109] G. I. Sapijaszko and W. B. Mikhael, "An Overview Of Recent Window Based Feature Extraction Algorithms For Speaker Recognition," in *IEEE International Midwest Symposium on Circuits and Systems, MWSCAS*, 2012, pp. 880 - 883.

[110] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *The Journal of the Acoustical Society of America,* vol. 87, p. 1738, 1990.

[111] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "RASTA-PLP speech analysis technique," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 1992, pp. 121-124.

[112] M. D. Skowronski and J. G. Harris, "Human factor cepstral coefficients," *The Journal of the Acoustical Society of America,* vol. 112, p. 2279, 2002.

[113] F. Hönig, G. Stemmer, C. Hacker, and F. Brugnara, "Revising Perceptual Linear Prediction (PLP)," in *INTERSPEECH, ISCA*, 2005, pp. 2997-3000.

[114] B. Ricaud and B. Torrésani, "A survey of uncertainty principles and some signal processing applications," *Advances in Computational Mathematics,* vol. 40, pp. 629-650, 2013.

[115] P. D. Shukla, "Complex Wavelet Transforms And Their Applications," M.Phil., Signal Processing Division, University of Strathclyde in Glasgow, Scotland, United Kingdom, 2003.

[116] L. Cohen, "The Wavelet Transform and Time-Frequency Analysis," in *Wavelets And Signal Processing*, L. Debnath, Ed., ed: Birkhauser Boston, 2003.

[117] J. C. Pedraza-Ortega, E. Gorrostieta-Hurtado, M. Delgado-Rosas, S. L. Canchola-Magdaleno, J. M. Ramos-Arreguin, M. A. Aceves Fernandez*, et al.*, "A 3D Sensor

Based on a Profilometrical Approach," *Sensors (Basel),* vol. 9, pp. 10326-40, 2009.

[118] S. Mallat, *A wavelet tour of signal processing: the sparse way*: Academic Press, 2009.

[119] G. Li and A. A. Khokhar, "Content-based Indexing and Retrieval of Audio Data using Wavelets," *IEEE International Conference on Multimedia and Expo (ICME),* vol. 2, pp. 885 - 888, 2000.

[120] C.-H. Chuan, S. Vasana, and A. Asaithambi, "Using Wavelets and Gaussian Mixture Models for Audio Classification," pp. 421-426, 2012.

[121] I. T. Jolliffe, "Principal Component Analysis and Factor Analysis," in *Principal component analysis*, ed: Springer, 1986, pp. 115-128.

[122] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science,* vol. 2, pp. 559-572, 1901.

[123] A. S. W. Wong and S. K. Chalup, "Towards Visualisation Of Sound-Scapes Through Dimensionality Reduction," in *IJCNN*, Hong Kong, 2008, pp. 2833 - 2840.

[124] M. Scholz and R. Vigario, "Nonlinear PCA: a new hierarchical approach," in *European Symposium on Artificial Neural Networks, ESANN*, 2002.

[125] K. A. Sheela and K. S. Prasad, "Linear Discriminant Analysis F-Ratio for Optimization of TESPAR & MFCC Features for Speaker Recognition," *Journal Of Multimedia,* vol. 2, p. 6, 2007.

[126] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*: John Wiley & Sons, Inc., 2001.

[127] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks,* vol. 13, pp. 411-430, 2000.

[128] R. Mogi and H. Kasai, "Noise-Robust Environmental Sound Classification Method Based On Combination Of ICA And MP Features," *Artificial Intelligence Research,* vol. 2, 2012.

[129] P. Paatero and U. Tapper, "Positive Matrix Factorization: A Non-Negative Factor Model with Optimal Utilization of Error Estimates of Data Values," *Environmetrics,* vol. 5, pp. 111-126, 1994.

[130] D. D. Lee and H. S. Seung. (1999) Learning The Parts Of Objects By Non-Negative Matrix Factorization. *letters to nature*.

[131] B. Wang and M. D. Plumbley, "Musical Audio Stream Separation By Non-Negative Matrix Factorization," in *UK Digital Music Research Network, DMRN*, 2005.

[132] T. O. Virtanen, "Monaural Sound Source Separation by Perceptually Weighted Non-Negative Matrix Factorization," Tampere University of Technology, Institute of Signal Processing, 2007.

[133] R. Schalkoff, *Pattern Recognition Statistical, Structural and Neural Approaches*: John Wiley & Sons, Inc., 1992.

[134] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*: John Wiley & Sons, Inc., 2001.

[135] K. Itoyama, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Simultaneous processing of sound source separation and musical instrument identification using Bayesian spectral modeling," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2011, pp. 3816-3819.

[136] C. Sutton, "An Introduction to Conditional Random Fields," *Foundations and Trends® in Machine Learning,* vol. 4, pp. 267-373, 2012.

[137] Y. Wang and D. Wang, "Cocktail Party Processing via Structured Prediction," in *Neural Information Processing Systems Conference, NIPS*, 2012.

[138] A. S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound* MIT, 1990.

[139] Y. Hifny and S. Renals, "Speech Recognition Using Augmented Conditional Random Fields," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 17, pp. 354-365, 2009.

[140] V. Vapnik and A. Lerner, "Pattern recognition using generalized portrait method," *Automation and Remote Control,* vol. 24, pp. 774–780, 1963.

[141] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Fifth Annual Workshop on Computational learning theory , COLT*, 1992, pp. 144-152.

[142] D. Luenberger, *Linear and Nonlinear Programming*: Addison-Wesley, 1984.

[143] M.-W. Mak and S.-Y. Kung, "Low-Power SVM Classifiers for Sound Event Classification on Mobile Devices," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 1985 - 1988.

[144] R. Saeidi, H. R. Sadegh Mohammadi, and M. K. Amirhosseini, "An Efficient GMM Classification Post-Processing Method for Structural Gaussian Mixture Model Based Speaker Verification," vol. 1, pp. I-909-I-912, 2006.

[145] H. C. Bao and Z. C. Juan, "The research of speaker recognition based on GMM and SVM," 2012, pp. 373-375.

[146] S. P. Mohanapriya, E. P. Sumesh, and R. Karthika, "Environmental sound recognition using Gaussian mixture model and neural network classifier," 2014, pp. 1-5.

[147] N. L. Biggs, E. K. Lloyd, and R. J. Wilson, *Graph Theory 1736-1936*: Clarendon Press, 1986.

[148] Z. Ghahramani, "An Introduction to Hidden Markov Model and Bayesian Networks," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 15, pp. 9-42, 2001.

[149] M. G. a. S. Young, "Architecture of an HMM-Based Recogniser," *Foundations and Trends in Signal Processing,* vol. 1, pp. 195–304, 2007.

[150] A. C. Damianou and N. D. Lawrence, "Deep Gaussian Processes," in *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2013.

[151] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature,* vol. 323, pp. 533-536, 1986.

[152] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Computing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds., ed: MIT Press, Cambridge, Massachusetts, 1986.

[153] A. Cauchy, "Méthode générale pour la résolution des systems d'équations.," *Compte Rendu à l'Académie des Sciences,* vol. 25, pp. 536-538, 1847.

[154] J. Barzilai and J. Borwein, "Two-Point Step Size Gradient Methods," *IMA Journal of Numerical Analysis,* vol. 8, pp. 141-148, 1988.

[155] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards,* vol. 49, p. 409, 1952.

[156] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," pp. 177-186, 2010.

[157] J. Duchi, E. Hazan, and Y. Singer, "Adaptive SubgradientMethods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research* vol. 12, pp. 2121-2159, 2011.

[158] M. D. Zeiler. (2012, ADADELTA: An Adaptive Learning Rate Method.

[159] D. Kingma and J. Ba, "ADAM: A Method For Stochastic Optimization," in *International Conference for Learning Representations, ICLR*, 2015.

[160] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," *Journal of Machine Learning Research, JMLR,* 2011.

[161] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *International conference on machine learning (ICML)*, 2013.

[162] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *IEEE International Conference on Computer Vision, ICCV*, 2015.

[163] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast And Accurate Deep Network Learning By Exponential Linear Units (ELUs)," in *International Conference on Learning Representations* 2016.

[164] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research, JMLR,* vol. 15, pp. 1929-1958, 2014.

[165] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning (ICML)*, 2015.

[166] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations By Back-Propagating Errors," *Nature,* vol. 323, pp. 533-536, 1986.

[167] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting And Composing Robust Features With Denoising Autoencoders," in *Proceedings of the 25th international conference on Machine learning, ICML*, 2008, pp. 1096-1103.

[168] K. Zen, M. Suzuki, H. Sato, S. Oyama, and M. Kurihara, "Monophonic sound source separation by non-negative sparse autoencoders," in *IEEE International Conference on Systems, Man, and Cybernetics*, USA, 2014, pp. 3623-3626.

[169] L. Boltzmann, *Lectures on Gas Theory*: Courier Corporation, 1995.

[170] S. E. Fahlman, G. E. Hinton, and T. J. Sejnowski, "Massively Parallel Architectures for Al: NETL, Thistle, and Boltzmann Machines," in *National Conference on Artificial Intelligence, AAAI*, 1983.

[171] P. Smolensky, "Information Processing in Dynamical Systems: Foundations of Harmony Theory," in *Parallel distributed processing: explorations in the microstructure of cognition*, D. E. Rumelhart and J. L. McClelland, Eds., ed, 1986, pp. 194-281.

[172] G. Hinton. (2010, A Practical Guide to Training Restricted Boltzmann Machines.

[173] A. Fischer and C. Igel, "An Introduction to Restricted Boltzmann Machines," *Lecture Notes in Computer Science,* vol. 7441, pp. 14-36, 2012.

[174] G. E. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation,* vol. 14, pp. 1771-800, Aug 2002.

[175] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," *Journal of Machine Learning Research,* vol. 9, 2010.

[176] G. W. Taylor, G. E. Hinton, and S. Roweis, "Modeling Human Motion Using Binary Latent Variables," in *Advances in Neural Information Processing Systems, NIPS*, 2006, pp. 1345-1352.

[177] G. W. Taylor and G. E. Hinton, "Factored conditional restricted Boltzmann Machines for modeling motion style," in *Proceedings of the 26th Annual International Conference on Machine Learning, ICML*, 2009, pp. 1025-1032.

[178] A.-R. Mohamed and G. Hinton, "Phone Recognition Using Restricted Boltzmann Machines " in *IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP*, 2010.

[179] E. Battenberg and D. Wessel, "Analyzing Drum Patterns Using Conditional Deep Belief Networks," in *International Society for Music Information Retrieval, ISMIR*, 2012.

[180] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," vol. 6791, pp. 52-59, 2011.

[181] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning, ICML*, 2009, pp. 1-8.

[182] P. J. Werbos, "Backpropagation Through Time: What It Does and How to Do It," *Proceedings of the IEEE,* vol. 78, p. 11, 1990.

[183] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 06, pp. 107-116, 1998.

[184] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional Recurrent Neural Networks for Music Classification," in *arXiv preprint arXiv:1609.04243*, 2016.

[185] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," in *NIPS workshop on Deep Learning and Representation Learning*, 2014.

[186] A. Graves and J. Schmidhuber, "Framewise Phoneme Classification With Bidirectional Lstm and Other Neural Network Architectures," *Neural Networks,* vol. 18, pp. 602-10, Jun-Jul 2005.

[187] K. Lee, Z. Hyung, and J. Nam, "Acoustic Scene Classification Using Sparse Feature Learning And Event-Based Pooling," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, 2013, pp. 1 - 4.

[188] L. Wyse, "Audio Spectrogram Representations for Processing with Convolutional Neural Networks," in *International workshop on Deep Learning and Music*, 2017.

[189] J. Pons and X. Serra, "Designing Efficient Architectures for Modeling Temporal Features with Convolutional Neural Networks," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017.

[190] K. K. Palawal, "Use of Temporal Correlation between Successive Frames in a Hidden Markov Model Based Speech Recognizer," in *International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 1993.

[191] M. Lin, Q. Chen, and S. Yan, "Network In Network," in *International Conference on Learning Representations, ICLR*, 2014.

[192] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals & systems*. USA: Prentice-Hall, 1996.

[193] B. L. Sturm, "The State of the Art Ten Years After a State of the Art: Future Research in Music Information Retrieval," *Journal of New Music Research,* vol. 43, pp. 147-172, 2014.

[194] F. Font, G. Roma, and X. Serra, "Freesound technical demo," pp. 411-412, 2013.

[195] M.-A. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon, "Multiple instance learning: A survey of problem characteristics and applications," *Pattern Recognition,* vol. 77, pp. 329-353, 2018.

[196] R. Al-Rfou, G. Alain, A. Almahairi, and e. al., "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints,* vol. abs/1605.02688, May 2016.

[197] F. Chollet. (2015). *Keras*. Available: https://github.com/fchollet/keras

[198] FFmpeg Developers. (2016). *FFmpeg*. Available: http://ffmpeg.org/

[199] M. McVicar, C. Raffel, D. Liang, O. Nieto, E. Battenberg, J. Moore*, et al.* (2016). *LibROSA*. Available: https://github.com/librosa/librosa

[200] B. McFee, E. J. Humphrey, and J. P. Bello, "A Software Framework for Musical Data Augmentation," in *International Society for Music Information Retrieval (ISMIR)*, 2015, pp. 248-254.

[201] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle*, et al.*, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech and Language Processing,* vol. 14, pp. 1832-1844, 2006.

[202] U. Marchand and G. Peeters, "The Modulation Scale Spectrum and its Application to Rhythm-Content Description," in *International Conference on Digital Audio Effects (DAFx)*, 2014.

[203] K. Seyerlehner, M. Schedl, T. Pohle, and P. Knees, "Using Block-Level Features for Genre Classification, Tag Classification and Music Similarity Estimation," in *Music Information Retrieval eXchange, MIREX*, 2010.

[204] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer, "Evaluating Rhythmic Descriptors for Musical Genre Classification," in *International AES conference*, 2004.

[205] T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer, "On Rhythm And General Music Similarity," in *International Society for Music Information Retrieval, ISMIR*, 2009.

[206] A. Lykartsis and A. Lerch, "Beat Histogram Features For Rhythm-Based Musical Genre Classification Using Multiple Novelty Functions," in *Conference on Digital Audio Effects (DAFx-15)*, 2015.

[207] H. Homburg, I. Mierswa, B. Moller, K. Morik, and M. Wurst, "A Benchmark Dataset for Audio Classification and Clustering," in *International Symposium on Music Information Retrieval*, 2005.

[208] Y. Panagakis, C. L. Kotropoulos, and G. R. Arce, "Music Genre Classification via Joint Sparse Low-Rank Representation of Audio Features," *IEEE/ACM*

Transactions on Audio, Speech, and Language Processing, vol. 22, pp. 1905-1917, 2014.

[209] C. Osendorfer, J. Schluter, J. Schmidhuber, and P. v. d. Smagt, "Unsupervised Learning of Low-Level Audio Features for Music Similarity Estimation," in *Workshop on Speech and Visual Information Processing in conjunction with the International Conference on Machine Learning (ICML)*, 2011.

[210] Y. Panagakis and C. Kotropoulos, "Music classification by low-rank semantic mappings," *EURASIP Journal on Audio Speech and Music Processing,* 2013.

[211] K. Seyerlehner and G. Widmer, "Fusing Block-Level Features for Music Similarity Estimation," in *International Conference on Digital Audio Effects (DAFx-10)*, 2010.

[212] K. Aryafar and A. Shokoufandeh, "Music Genre Classification Using Explicit Semantic Analysis," in *International ACM workshop on Music Information Retrieval With User-Centered and Multimodal Strategies (MIRUM)*, 2011.

[213] F. Moerchen, I. Mierswa, and A. Ultsch, "Understandable Models of Music Collections Based on Exhaustive Feature Generation with Temporal Statistics," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

[214] J. Schluter and C. Osendorfer, "Music Similarity Estimation with the Mean-Covariance Restricted Boltzmann Machine," in *International Conference on Machine Learning and Applications, ICMLA*, 2011, pp. 118-123.

[215] K. K. Chang, J.-S. R. Jang, and C. S. Iliopoulos, "Music Genre Classification via Compressive Sampling," in *International Society for Music Information Retrieval, ISMIR*, 2010.

[216] S. Sigtia and S. Dixon, "Improved Music Feature Learning With Deep Neural Networks," in *International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 2014.

[217] J. Bergstra, M. Mandel, and D. Eck, "Scalable Genre and Tag Prediction with Spectral Covariance," in *International Society for Music Information Retrieval, ISMIR*, 2010.

[218] Y. Panagakis, C. Kotropoulos, and G. R. Arce, "Non-Negative Multilinear Principal Component Analysis of Auditory Temporal Modulations for Music Genre Classification," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 18, pp. 576-588, 2010.

[219] Y. Aytar, C. Vondrick, and A. Torralba, "SoundNet: Learning Sound Representations from Unlabeled Video," in *Neural Information Processing Systems (NIPS)*, 2016.

[220] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations, ICLR*, 2015.

[221] R. Arandjelovic and A. Zisserman, "Look, Listen and Learn," in *IEEE International Conference on Computer Vision, ICCV*, 2017.

[222] Y.-h. Chen, I. Lopez-Moreno, T. N. Sainath, M. Visontai, R. Alvarez, and C. Parada, "Locally-Connected and Convolutional Neural Networks for Small Footprint Speaker Recognition," in *INTERSPEECH* 2015.