UNIVERSIDAD AUTÓNOMA DE YUCATÁN

FACULTAD DE MATEMÁTICAS
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

.

TESIS PARA OPTAR EL TÍTULO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

Autor: Juan José Alvarado Leaños

Asesores: Víctor Uc Cetina y Anabel Martín González

Mérida, Yucatán, Febrero, 2020
MÉXICO

CONTENTS

Chapter I

Introduction and Motivation

Environmental sounds carry a good deal of information about the surrounding environment, from individual events, like a window breaking or a door knob, to the sound scene of a particular setting, park or restaurant. Section 1.1 introduces the basic concepts and problems associated with the computational analysis of sound scenes and events. We motivate the computational problem by presenting a brief review of various applications where the methods can be used, methods that are data-driven and analyze environmental sound information automatically. The main differences between environmental and other major audio understanding such as automatic speech recognition and music information retrieval is presented.

In order to motivate the field of sound recognition, let us consider the following example. You are in your car and you say open the door you might want your car to understand your voice command and open the car's door. In that case we need a automated system that is able to recognize spoken words. This problem is called in the academic world as Automatic Speech Recognition (ASR). But we would like also to our smart car system to detect when a crystal is broken a window is broken and to differentiate it from a bump in the chassis. We now need machine understanding of another kind of sound. We can also go about it using a mechanical sensor. But thanks to the ability of sound to propagate through sound and convey meaningful information we can also detect a window breaking in a house even if we don't have a mechanical sensor as is the case in the car.

This way it is easy to see that sound provides clear advantages, but it also poses serious challenges. As a starting point, sound is a complex temporal signal which is very difficult to summarize in a simple class identificator, an integer from 1 to n with n the number of classes we're interested in recognizing, i.e., a gunshot or a car crash. In order to have this human-like understanding of sound, researchers in the community have adapted machine learning approaches by using datasets and training learning models of said datasets.

The availability of sound recording devices allows us deploy in the future smart audio sensors in not just cars but homes, and parks in cities. Effectively achieving a smart city or a smart

home, different smart spaces. Also it has been proposed smart sound libraries where one can use high-level search queries to explore and browse a given repository of sound and video. Film and music creation would be greatly facilitated by automatic systems that are able to understand sounds, music, and speech as humans do.

More technically, the purpose of this work is to present the design and development of AclNet, an end-to-end neural network designed for Acoustic Event Detection (AED) of environmental sounds. Environmental sounds are those common, everyday sounds that cannot be simply categorized as speech or music. AED or environmental sound classification (ESC), as was once called, is a relatively understudied research problem. However, it is vital in various applications ranging from smart hearing aids, smart cities, smart home assistants and for the Internet of Things (IoT) devices. Sound-intelligent devices may be able to aid in the surveillance of protected natural parks, cities, and homes by allowing a rapid response to emergencies. In the next section, we delve more deeply into some application domains of AED.

1.1 Basic Concepts and Terms

As mentioned previously, an environmental sound is defined as an everyday non-music, non-speech sound [54]. However, this definition is not without ambiguity. There are multiple cases of considerable overlap between the categories of speech, music and environment sounds. Nowadays the emphasis is not so much in separating sounds in environmental versus non-environmental but rather in the fundamental concept of an acoustic event and the secondary derived concept of an acoustic scene. An acoustic event is a sound that is perceived as individual, in contrast a sound scene is a collection of sound events that tend to co-occur [6]. Moreover, to add another layer of complexity to the conceptual framework used. Also, a sound event can be polyphonic.

By its very nature a sound event has a beginning and an end. Hence, detection can be done at multiple levels. The full temporal resolution detection outputs not just the class or type of sound event but also the timestamps of beginning and end. Other types of detection can be done at the whole sample level or at individual chunks of the input. Finally, tagging occurs when we do multi-class classification at the level of a recording.

Usually paired with ESC are two other tasks: acoustic scene classification (ASC) and sound event localization (SEL). In sound localization, the task is to output the spherical coordinates of a multi-channel recording of a sound or multiple sound sources. Acoustic event detection (AED), acoustic scene classification (ASC) and sound event localization (SEL) are three interrelated tasks that allow a computer to understand sound similar to how animals perceive sound. Most challenges have any variation of the three fundamental tasks.

1.2 Applications

Because of the ubiquity of sound signals in almost any situation and environment, numerous applications are possible including audio surveillance systems that detects dangerous sound events [47], hearing aids [1], smart home monitoring [77] and video content highlight generation [5]. Let us move into some detail in each of these applications.

In particular, for automatic surveillance systems, sound analysis is used as a supporting secondary signal of video-based event detection. In principle, it is possible to recognize dangerous events by the analysis of their acoustic properties. To achieve that goal, a feature extraction and a classification technique must be specified. The system, then, gives one of two output messages, one representing a no-danger situation and the other alarming the operator about that a dangerous event has been detected. These systems are less complex because their output is binary and thus higher accuracies, above 90%, can be achieved in conditions of low noise [47].

As another example of an application, let us consider the fact that as much as 13% population in developed countries suffer from debilitating hearing loss [10]. However most of these people don't own a hearing aid, but even those who do don't wear them because of irritating and unpleasant amplified sounds or whistles produced as artefacts in these devices. This is because most hearing aids cannot automatically adapt to the changing acoustical environment the user finds herself in. In order to enhance listening comprehension when the user goes from one sound environment to another there have been developments of automatic sound classifiers that provide the sound aid the acoustic context in which it is immersed [1]. Thus the objective of researcher in this field is to perform classification followed by adaptation to increase intelligibilty and comfort for users of hearing aids and preserve or enhance quality-of-life.

Now, in the context of a medical monitoring system, sound classification can provide vital information by taking into account calls of distress from the resident or patient and from sounds from falls, for example. Progress in aids and assistive technologies may represent a cost-efficient way to support the supply of informal care and official care provisions. Specially, given the fact increased lifespans have caused the aging of the general population all the industrialized world. In this context the central challenge is to provide full access to high-quality services for the elderly or a patient that may require care at their home. As a concrete example, [77] take into account information from medical sensors is combined with sound detection to identify bevhavior changes and ascertain the state of the patient in order to detect critical or a distress situation.

Finally, consider the detection of certain events in video data, in what is called video highlighting. This allows an increase the ease of access, the ease of browsing and summarization in general of audiovisual material. Similarly to our first example, audio is used as a support signal that aids visual analysis. In other words, classification of audio events may create a more thorough description of the video's content or it may help the refinement of the detection of highlights. Highlighting allows to select a portion of a long video and summarize it in the way that it contains the most important developments in the match, when we are talking about sport

videos, for example. As a concrete case, consider the study by Ballan et al. [5] where generic audio concepts like excited speech, whistling, or domain specific are combined to produce different audio descriptors for later proccessing by a classificator such as SVM or deep belief networks in order to highlight certain segments in sports videos.

Sound classification systems can be deployed as a sensor network in cities or natural parks. And the density depends on the specific application. For example, more resolution in events requires a denser network. Dense type networks maybe needed for smart city applications where it is important to know precisely or have a precise map of the different sources of alarm sound or noises as in [7]. Sound recognition systems that are part of a dense network sensor node, and can send messages through the network to alert of intrusions to houses, car crashes at intersections, or the sound of a chainsaw in a protected forest, for example.

As part of stand-alone home devices or IoT devices, domotics (give assistance to people with disabilities), noise monitoring in cities, sound databases for database queries and search by example, multimodal applications (assisting in the detection of actions in videos). The most developed applications areas are four in number: sound retrieval in databases, bioacoustic applications, smart cities and smart homes. Following we will describe briefly each of these application domains or areas.

In sound retrieval systems, it is importat to annotate sounds to order and have a sense of order in the database, also there is the problem of searching with a sample query instead of using the property of using the same technique. Also for video databases we can use sound event detection to classify videos in the database. We have then sound categorization important to annotate video and sounds and also to search by example (QBE) [9].

It is important to talk about query by example, audio fingerprinting in general audio as opposed to music and so on. Audio-based content filtering or retrieval that is the case, right now MFCC descriptors are widely used but we can use thumbnails based with detectors so that users can remember and recognize more easily a given recording in the database. So we have three forms of audio parameters that can be useful to understand an audio collection: fingerprinting, search by content and thumbnailing.

Let us end this section with a general comment about the challenges that we see is present in more or less all the applications mentioned above. One of the hindrances to a more active research in this field is a widespread fragmentation and sparsity in reproducibility and comparability. Most studies so far, have been studies on datasets that are either very specific, small or proprietary. This sparcity of publicly standardized datasets and publicly available and difficulty in accessing the original code for study replication, coupled with the lack of access of the original code for study replication, make research reproducibility of research a great challenge. That is in stark contrast to handwritten recognition or large-scale image classification which have been prominently used for baseline comparisons. Only recently with ESC-50 [54] andinitiatives such as the Urban Sound project [60] bring some needed change for the community.

1.3 Problem Overview

To help the reader better understand the core of the task of acoustic event recognition, we now provide a high-level description of the problem and the motivation for our proposed approach. Along the way we can gain a better understanding of some of the design choices and overall approach path taken.

Imagine that we want to build a detector of dog barks for a disabled person, a deaf inhabitant of a home, i.e, a smart detector of dog barks. If we would like it to be automatic, how can I go about programming such a detector? Computers don't have ears like us, all they "see" is a waveform.

One way to do it, would be to look at the waveform of a dog bark and measure the zero crossings for example, which give you a rough sense of the periodicity of the sound. Another would be to look at the envelope of the waveform and the time where its peak occurs (the attack time) of the sound. Also, if you know some things about Fourier analysis you could also look at spectrograms, which are time-frequency representations of sound. Similarly to waveforms, sound experts measure envelopes and things like the lowest frequency component known as the fundamental frequency (F_0).

A third class of features could be the ones that take inspiration from our own perceptual experiences with sound, for example, loudness or timbre, or brightness of the sound. We could ask people about what kind of characteristics they "feel" when they hear examples of dog barks. These class of features is called psychoacoustic features.

After analyzing many waveforms we have come to establish some patterns of association in the acoustic properties of dog barks, its features. For example, for signals that have this amount of zero crossings and attack time, if the fundamental frequency is greater than 10 Hz it is very likely that the signal represents a dog bark. Or, if the amplitude is more than 50 dB and the rate of the decay of the temporal envelope is this high, you can look at the fundamental frequency to decide into bark and non-bark sound. That kind of knowledge can be succintly represented/put in a decision tree. In general, once you have a set of features each with a threshold value or values one can come up with a decision tree of that form. A decision tree can be easily converted to pseudocode so that a computer programmer could translate that to a suitable programming language and the pattern dog bark detector can be run in a computer system, be it a smartphone, laptop, embedded computer like the one there are in microwaves and cars.

That would be what could be called the analytic-deductic method of solving the problem of detecting dog barks from noise automatically via a computer. And it requires a large amount of time and effort to discover this patterns in the features of dog barks. However, there is a completely different approach that is also possible. Imagine that after many hours of being exposed to waveforms of barks I have become very good at identifying barks of dogs just by looking at the waveforms, I've become an expert recognizer of barks. It has become a gut-feeling that with good accuracy can give very good results. I have acquired a implicit knowledge

that is very hard or impossible to describe how I do it. This mode of solving the problem of detection could be labeled as empirical-inductive because it arises from me being exposed to a large number of examples of waveforms. The same thing can be said to any other representation of sounds like spectrograms. Machine learning systems like the one presented in this work are a form of empirical-inductive systems, that automize the kind of gut-feeling decision making that humans do. They are equivalent to a form of perception together with decision making. And that is why they are said to have intelligence, or be a form of cognitive technologies.

There are many machine learning models available, but neural networks are the ones that have the capacity to perform well given enough data and have the property similar to our one sense of hearing, taking the raw audio and making a classification decision. But how do they do it? How do neural networks work? A rough explanation would be the following. In artifical neural networks, neurons are represented by a single real number, also called an activation. This neuronal unit is connected to others neurons through a set of weights (more real numbers) that are also called parameters, And this parameters are updated through a training procedure, similar to the process of going through a lot of pictures of sounds and getting used to recognize them by intuition.

This way the key components of the system can be introduced as follows:

1. A dataset with supervision, that is, we know in advance what class any given sound of the dataset belongs to.

2. A neural network, which maps input sounds to a list of class probabilities. The length of this list is equal to the number of classes that we wish to classify.

1.4 Problem Statement

The goal of this work is to describe the design and implementation of a neuronal classifier (a deep convolutional neural network) of acoustic events taken from the ESC-50 environmental sound dataset with good level of accuracy using few parameters by applying an end-to-end training methodology and comparing it with transference and the state-of-the-art.

Chapter II

Prior Work

In this chapter we present the two diverging schools of approaching the problem of sound recognition: feature engineering and end-to-end learning. Special emphasis is put on the main area of concern of this work: automated recognition of environmental sounds. Environmental sounds have been traditionally been classified with the use handcrafted features, see section 2.1. However, nowadays artificial neural network approaches and big data methods are starting to dominate the field. We explain this new paradigm in section 2.2. Finally, in the last and third section, we explain an intermediary approach which seeks to learn features separatedly from learning a discriminator model, section 2.3.

2.1 Feature Engineering

Handcrafted features are those that are obtained through the use of pre-defined operations that extract a set of real numbers (known as features) from the input signal so that the new set of features is a better representation of the input, where "better" is defined based on the kind of downstream tasks. So, for example, a given set of handcrafted features might be good for certain downstream task like speech recognition but not for music genre classification; or, likewise, the representation is fit for speech recognition but not for speech enhancement, and so on.

Therefore, hand-engineered features and the procedure to obtain them had to be modified or tailored according to the problem at hand by the system engineer. This is in contrast to automatically obtaining said features, see next section for that approach. Hand-engineered features can be categorized in frequency-based (or spectral features), time-based or perceptually-motivated.

Let us discuss some of the most salient properties of the single most used spectral feature used in machine listening: mel-filterbanks. The set of operations to extract them are the following: first, a short-time Fourier transform (STFT) is computed; then a triangular bandpass filter

is applied. Those filters in the filterbank are spaced on a logarithmic scale (the mel-scale) that is designed to reproduce the non-linear relation between frequency and human perception of pitch [68]. Finally, the resulting features are compressed logarithmically; and similarly to the steps before, this one is designed to reproduce the psychophysical finding that humans have a non-linear sensitivity to loudness [22], see the following equation:

$$p = k \ln \frac{S}{S_0} \tag{2.1}$$

Equation 2.1 is called Fechner's law for its proponent and here p denotes the perceived level, k a constant, and S the volume of a sound or intensity level of a stimulus, more generally. It is important to note that this approach of replicating human sound perception has been historically successful with the well-known work of [19] where they compared different handcrafted features and concluded that mel-filterbanks were the best for word recognition. That result and many others has also been reviewed in the work [49]. A landmark theoretical result was obtained decades after the initial proposal of mel-filterbanks by [2], who showed that this type human-inspired filters exhibit invariance to temporal shifts and small deformations. All very beneficial properties for the purposes of machine learning. Despite of all these benefits and evidence of performant capabilities, there are some problems associated with them. First, the original experiments that led to their design could not be replicated according to [27], prompting the authors of said studies to proclaim that the original studies aren't valid because of their disqualifying bias.

Other authors have revised the original scale. O'Shaughnessy in his textbook [50] and later in the study by Umesh et al. [76]. Yet, other investigators have inquired in alternatives to log-compression like obtaining the tenth root of the signal, [61], or cubic root, [48]. Despite all these considerations, some of the aforementioned biases may be beneficial for some application domains but not for others as we explained before. That is when the power of automatically-generated features as opposed to manually-created can be gleamed. Inspired in this new way of thinking about feature extraction we have two different approaches, learning the feature extractor jointly with the feature discriminator as in modern deep learning techniques, or learning one first and then the other, as in the feature learning community approach, Sect 2.3.

Finally, we can combine the mel scale and the cepstral coefficient to obtain MFCCs a well-known and popular feature of speech that allow the decomposition of gearbox and oral cavity [63]. Amazingly, MFCCs were the state-of-the-art before deep learning and feature learning took off. The main drawback of these other handcrafted features, other than mel-filterbanks, is that, the performance was usually marginal or patchy, that is, applied to some sounds but not all.

The introduction of deep learning techniques in this context has slowly begun in the last few years. However, these efforts are still mostly limited to analyzing highly pre-processed acoustic features. [3], [39], [55]. At the same time, classification of environmental sounds

is still predominantly based on applying general classifiers, such as support vector machines, hidden markov models, and gaussian mixture models, to manually extract features. For more information on said feature engineering methods and the general classifiers thereby used see the following reviews for a detailed analysis [12], and [6].

2.2 Deep Structured Learning

Deep Structured Learning, or just Deep Learning (DL) [43], is a relatively recent methodology that relies on a hierarchy of feature extractors. Besides hierarchical representations and high-layer count another features of this subfield is the very large model size, in terms of parameter count, and the very large datasets which these high-capacity models are able to fit. It began to be a popular methodology since applied to image classification and winning the ImageNet [57] contest by a wide margin. Now the techniques have been transferred to other machine learning tasks and data domains, such as speech and text. In particular, recent works have addresssed the problem of acoustic event detection with deep neural networks. Following is mainly a historical account of the development of deep models for sound classification.

2.2.1 Beginnings: ESC-50 and PiczakNet

In 2014, sound researcher Karol Piczak at Warsaw University introduced the ESC-50 dataset [54] pooled from the Freesound sound-sharing site so that a total of 2,000 sound examples of 5 seconds could be further enhance improve the research on AED. In the same year, Piczak published the first architecture [53] to solve ESC-50 classification. The input is logmel spectrograms or MFCCs plus delta-MFCCs. The network proposed by Piczak are still relevant despite their being forerunners specially when combined with powerful feature learning techniques that augment the filterbank beyond simple logmel and logmel deltas. See our discussion of feature learning below.

2.2.2 Important Developments: Audio Set and Transference Methodologies

During the following years many innovative developments appeared in the literature. Prominent among these novel approaches is the concept topic of transference. Knowledge transfer [42], as is sometimes called, can be done across modalities, like from video to sound as in SoundNet. Aytar's SoundNet [4] which uses a teacher-student framework to transfer embeddings from one modality (visual) to another (sound). Using videos from the video-sharing site Youtube the researchers trained a sound detecting network without the need for labels or supervision. Kumar et al. [42] transferred from applied a more conventional form of transference, sound to

sound in a way called transductive because the it was across different sound tasks. The novelty resided in the thoroughness and complexity of the transferred modes approaches.

Following closely the big data trends from other fields of machine learning, AudioSet [24] was introduced as a very large-scale natural sound dataset with its own ontology of sounds. This allowed Kumar et al. [42] to try transfer learning as a pivot, pioneering the application in applying transference from one audio dataset to another of the same type. And this work serves as an inspiration and model for our own transference efforts. With a large-scale dataset in hand now bigger more complex models were used to solve audio classification problem and the general problem of general sound recognition. In [30], the researchers tried out well-known famous standard well-trodden models as ResNets and VGG-like [66] networks.

Finally, another very innovative approach to sound recognition was taken by EnvNets [73]. The researchers proposed the vital, at least for this work, method of end-to-end learning for acoustic detection. EnvNets use a transpose operation and treat the feature maps obtained in the first stage of feature extraction as spectrograms. This strategy was inspired by a speech recognition work, namely [59]. While the accuracy was not perfect further developments such as audio-specific augmentation (see Section 4.2.1) and between class learning [74] increased the performance of EnvNets to a close second (84% five-fold accuracy with 48kHz sampling rate).

For a more thorough coverage of current deep learning practices and methods, as well as their application to different domains, see recent published textbooks on the topic of deep learning in general: [26], speech recognition [80], and general deep learning from the authors of the previous work [20].

2.2.3 Contemporary Practices and Challenges

ConvNets since their inception have been modifying and improving mainly through the development, among these developments are fully convolutional topologies, global average pooling, strided convolutions instead of pooling layers, et. Dai et al. [18] applied these modern improvements to train very deep one-dimensional convnets for raw audio. Thus, just like EnvNets, DaiNets are also another end-to-end networks. Besides the developments there are also long-standing problems and questions that are very important today. Those questions include: How to introduce mixing into the training procedure? Tokozume et al. [74] tried to predict the mixing ratio. How should transference be done? Can it be done without supervision, that is unsupervised training or self-supervised approaches.

Deep learning approaches still pose the challenge of high memory and computing needs, so nowadays alternatives such as non-negative matrix factorization (NMF) [8] are very much in the research schedule of various groups. NMF, in particular, is a form of feature learning that promises good accuracy with lower resource requirements. Another form of performant feature learning is ConvRBM for filterbank learning as described in Sect. 2.3. Like non-negative

factorization, it promises economical means of achieving good performance. However, the high-end of performance is nowadays captured wholly by the heavy and deep models of deep learning and big data.

We can envision that in the future, a move to self-attention is possible. Self-attention [15] is another tool in the toolbox of deep learning that allows to be in the know. RNNs and DNNs are not very useful because of the high dimensionality of sound, the important aspect of sound is that there is a lot of redundancy in the time domain signals. For processing sound, we need an effective way to cope with the high sampling rate. Convolution with pooling is the standard approach, although as we have seen two-dimensional convolutions might not be appropriate. However one-dimensional convolutions are advisable, and so are two-dimensional convnets with self-attention.

2.3 Feature learning

An approach that lies between end-to-end learning approaches and feature engineering is feature learning. The relation between the three approaches to supervised classification is discussed in Section 3.2.2, here we are going to focus on previous works that apply feature learning to the sound collection, ESC-50. The most performant of these previous efforts is Sailor convolutional boltzmann machine to create a new set of filter bank to do filter bank learning by doing reconstruction the result was a set of filters that resembled the gammatone filters for some frequencies and the fourier filters for other, these filters when combined with the scores of a different PiczakNet would give much better results, an improve with respect to PiczakNet. When Sailor et al.'s ConvRBM [58] is fused with PiczakNet at the level of the output scores (class logits) gives 86% versus the best result for end-to-end learning 85% (cite our aclnet paper) with 48 kHz sampling rate.

The importance of this prior work is that ConvRBM combined with two Piczak's networks [53] give an outstanding good performance with extraordinarily low parameter requirements. This economical performance system can be leverage for constrained applications because of the peanuts size of the Piczak Network and the low cost of learned filterbank. In chapter 5, where we discuss the results and compare with other models there is a greater discussion about the tradeoffs between models. For now, we can state that it is a hard baseline to beat but that transfer learning did the job for deeper more complex of higher capacity models.

Chapter III

Theoretical Framework

The main goal of this chapter is to provide the theoretical foundations in order to understand the mathematics and the main conceptual tools to engage in an informed discussion and analysis and critique that will be covered in the following chapters. Hence, this chapter starts with an introduction of the important concept of spectrograms, how are they are calculated from a time-varying sound signal. WHy are they used their use and function and advatnages oft his use. Later we introduce the main methodology of this work: end-to-end training of deep learning models. Later, in the third and last section the architectural model chosen of our detector is described to some depth: convolutional neural networks.

3.1 Spectral Analysis

Sound signals, to be able to be processed by computers, need to be discrete in time and amplitude. Sound digitization is the process of mapping a continous-time signal to discrete time (sampling) and from continuous amplitude to discrete amplitude values (quantization). Thus to do spectral decomposition to digitized sound the correct transformation is a discrete time fourier transform (DTFT) that is discrete itself (discrete fourier transform).

Sampling has an interesting relationship with spectral analysis. This relationship is encapsulated by the Nyquist Theorem, which relates in which sampling (f_s) is done and the frequency range we discussed in the description of the short-time FT.

Quantization is done with a piece of hardware called analog-to-digital converter and usual output values are 16-bit or 48-bit integers that are able to describe 32,768 and 140,737e9 signal values, respectively. The number possible signal amplitudes is the bit depth of a digital sound file format is the number of levels used in the quantization and corresponds to the resolution of the signal or the image in pixels.

Spectrograms, also called sonograms when extracted from sound, are a compact time-frequency representation of signals. Spectrograms are computed using the short-time Fourier transform, a special form of a Fourier Transform (SFTF). There are, in fact, two types of spectrograms: amplitude and phase spectrograms. However, phase spectrograms are usually not taken into consideration, because of the low importance and relevance for signal processing further down. Therefore, when speaking of amplitude spectrograms it is commonly abreviated as spectrograms.

The Short-time Fourier Transform, or STFT, is the application of the Fourier Transform (FT) at each step or frame of the signal. The signal is supposed to be stationary, or periodic within the time frame. Whenever the STFT is used three parameters are needed to specify it completely, the hop size, the window or frame length and the windowing function used. The usual windowing functions are Hamming and Hanning, and their purpose is the smooth out the signal at the boundaries of the frame.

Summing up, the parameters to fully define a STFT are: 1) hop length, 2) frame length (or size), 3) windowing function, 4) frequency range (the filterbank). The hop length is chosen so that there is some overlap between one frame and the next. A common overlap value is 50% and makes sure that there is some correlation between one FT and the next. There is some complete notion of stationarity. This notion of stationarity is very important principle that underlies the reason the motivation or the explanation of why we can use STFT, is admissible.

Common windowing functions are Hamming and Hanning [45]. Both functions have in common that they taper off (i.e. approach zero smoothly) at the at boundaries. This type of Windowing allows the windowed function to satisfy the requirements of the Fourier Transform, that is the periodicity constraint, when a signal is windowed both the beginning and end is zero and thus free of jumps or discontinuities when the windowed signal is extended by repeating infinitely before in time and in the future.

The frequency range is important, specially the nonlinear frequency ranges that are based on simple psychoacoustics. Logmel is the prime example. Piczak [53] uses also the logmal center frequencies and the logmel deltas. As a preliminary step before obtaining a given spectral feature set, the waveform is converted to a frequency domain representation. The frequency-domain representation on a linear scale may be obtained with a discrete-time Fourier transform (DFT):

$$X(f) = \sum_{n=-\inf}^{\inf} x(n) \exp(-i2\pi f n) \tag{3.1}$$

The resulting representation $X(f)$ is periodic with periodicity equal to the sampling frequency and the frequency $f = f_s/2$ is called Nyquist frequency. The $x(n)$ spectral representation can be recovered from the spectral representation by way of the inverse Fourier transform in discrete time (IDFT). In practice a windowed frame of length N of the signal $x(n)$ is used

to apply the DFT. This is referred in the literature as short-time Fourier Transform (STFT). The equation for STFT is:

$$X(j,f) = \sum_{k=0}^{N-1} w(k)x(jN+k)\exp(\frac{-i2\pi jf}{N}) \tag{3.2}$$

where $w(k)$ is the windowing function (rectangular, Hanning, Hamming, etc) used to smooth out some of the effects of the discrete Fourier transform approximation, and to make the signal periodic and continuous at the edges of the frames. When there is no overlap as in Eq. 3.2 the hop between frames is equal to the length of the frames (N). However it is commonplace to choose a hop that is smaller than the frame length N to allow for some overlap.

Oftentimes specific frequency bands should be enhanced while others are to be attenuated in order to facilitate the work of a discriminator that takes the frequency bands as input. We now present the most common filterbanks, set of filters that enhance or attenuate certain bands according to a mathematical law.

Equivalent rectangular bandwidth (ERB) scale: This representation changes the center frequency and bandwidth to approximate something similar to auditory filters in the cochlea [25]:

$$\text{ERB}(f) = 24.7 \times \left(4.37\frac{f}{1000} + 1\right) \tag{3.3}$$

Gammatone filters: They are filters whose impulse response is a sinusoidal modulating wave by an envelope that has the form of a scaled gamma distribution function. This impulse function gamma(n) is given by:

$$\text{gamma}(n) = an^{\gamma-1}\exp(-2\pi bn)cos(2\pi f_c n + \Phi) \tag{3.4}$$

where a is the amplitude, γ is the filter order and b is the temporal decay constant, f_c and Φ the frequency and phase of the carrier, respectively

Mel scale: This scale approximates the psychological human perception of pitches of pure sounds. The analytic expression that relates the linear frequency scale (in Hz) and the mel scale is:

$$\text{mel}(f) = \frac{1000}{\log 2}\log\left(1 + \frac{f}{1000}\right) \tag{3.5}$$

3.2 Supervised End-to-End Learning

In this section we are going to explain the differences and compare the different approaches to feature extraction, those are: feature engineering, feature learning and end-to-end training. While the first two were already explained in Section 2.1 and Section 2.3, respectively. Here we are going to give them a fresh look through the comparison that we take when we compare them as we explain more fully end-to-end systems.

Feature engineering methods utilize feature extractors that have no learnable parameters. So, if we denote the feature extraction process as f, v is a feature vector and x is an input signal, we have: $v = f(x)$. This f can be a Fourier Transform or a zero-crossing rate calculator etcetera. The important point is that there are no free parameters that can be modified by a learning algorithm. All parameters, such as hop and frame length in STFT are constant.

In contrast, feature learning procedures allow f to be dependent on learnable parameters, that we shall denote as θ_{FE}. This way, we obtain: $v = f(x; \theta_{FE})$. This v can 1-dimensional or 2-dimensional as in the feature engineering method. However, here the objective is to reconstruct the input or to map the input to another space that has a different property. This v then is known as a latent representation or latent space. And vectors in latent space may be better suited for classification than the original input space. This mapping between visible and latent space is learned through the set of parameters grouped in θ_{FE}. The main point of feature learning methods then is that feature extraction and feature classification is done separately, there are two training processes that are carried out individually.

In contrast, end-to-end systems combine feature extraction and feature classification into a single training procedure. Thus, we have: $\sigma = g(x; \theta)$. σ being a vector with the class predictions for the input x where θ_C is the parameter set used to map the input from feature space to a particular class. The size of the vector/tensor σ is the same as the number of classes that we wish to classify. This way we can see that end-to-ende systems require a higher number of parameters (the same as feature learning) but they possibly have higher performance.

3.3 Convolutional Neural Networks

Convolutional neural networks date back to the 1980s as in Fukushima et al.'s Neocognitron [23] and LeCun et al.'s backpropagated handwriting recognizer [44], yet only now have they been adopted in the multimedia space as the method of choice for processing high-dimensional raw data: speech, images, music, and video, when big data is available. Without a doubt the work of Krizhevsky et al. [41] represented the breakout of this technology into the spotlight in the academy and the public in general. This work achieved first place in the 2012 ILSVRC [57] competition by a wide margin. Initially these models were used for visual recognition of house numbers [64], handwritten digits [16], and traffic signs [17]. However they quickly took

over other more challenging tasks and diversified into other data modalities like we have said. Convnets take advantage of the property of locality in data and that's the primary reason why they have proven to be viable solutions to problems in domains of very different kind.

However part of this successes is the computing environment surrounding convnets, the large datasets, the training techniques, the hardware accerelation and other concomitant facts that accompany other popular deep learning models like LSTM and Transformer. However, convnets have a very special niche of applications due to the special properties of the convolution operation, the core of convents. In the following we will discuss the primary reasons: a) the reduced parameter count required for convolving inputs as opposed to the ones required for doing simple matrix multiplication; and b) a special and very useful form of inductive bias.

The reason of this incredible success that the convolution, the underlying operation of CNNs, provide convolutional models with a) reduced parameter count and b) a special and very useful form of inductive bias. We will explain both of this goodness properties of the underlying operation of ConvNets the core, the heart of the CNNs in the following subsection. Reduced parameter count is of prime importance because it allows the CNNs to fit an arbitrary function with less parameters thus it is statisticallly efficient. It requires consequently less data.

3.3.1 Motivation for Using CNNs

Convolution are a weighted sum of neighboring signal values. Thus, but so is the fourier transform, if we take into account (as a kind of analogue of the STFT) if the analogy is drawn between the frame as the neighborhood in the convolution and then the convolution STFT is also a convolution but with real coefficients. See Equation.

Convolutions are also a form of template matchers. For example, when we want to detect edges or to detect corners we use a template of a corner and convolve this template with the image, the resulting two-dimensional output is a real number whose magnitude tells us the degree of correspodnence of the image patch and the template. It is the likelihood of the patch being a corner.

The output of a convolutional layer is called a feature map. It is called such because, if we consider the two-dimensional case, every point in the feature map represents the likelihood of matching the feature in the input image. The impressive fact is that we can use convolutions to process not only images but features map themselves!! These latter being image-like.

3.3.2 The ConvBNReLU Block

The essential component of a convnet is a convolution layer, however, these networks are not made only of convolutional layers. Rather, convolutional layers are part of a block of operations

that include:

1. A convolution layer
2. Batch Normalization
3. Rectification

The bundling of operations allow the neural network to be trained to large depths without hurting the performance of the gradient descent because of exploding or vanishing gradients problem.

In particular, rectification through ReLU is the simplest and cheapest in terms of computational cost. Also there is a biological plausibility argument here. Since it is known that most neurons have sparse activations (reponses) to a wide range of stimuli, like the negative input region of the ReLU. Also,

Batch normalization is an operation that involves substracting the batch mean and dividing by the batch deviation so that the new mean is zero and the new deviation is one. Batch normalization allows the training of deeper models by shifting the covariate shift of the activations.

It is important not to confuse input normalization, which we describe in Sect. 4.4, and batch normalization, which we describe in the following discussion. In contrast to input normalization, which is a normalization done only to the input and is elementwise, batch normalization is done between any pair of hidden layers in the network.

3.3.3 Pooling

After every convbnrelu block there is usually a pooling layer. This layer is always paired with convolutions because they work together very well in the task of inducing a translation-invariant representation in the feature map representation. As an added bonus, pooling allows the possibility of eliminating redundant information thereby allowing higher layers to represent information at a higher resolution level or scale. Computer vision have, historically, used Laplace pyramids so that features at different scales could be recovered. The combination of convolution + pooling is the equivalent of Laplace pyramids in the field of artificial neural networks. This way we can see that convolution followed by pooling allow deeper layers to process features at a gross level of detail; less detail and less redundant information. There are multiple ways of reducing the window size of pooling, the most common is by taking the maximum nubmer (maxpooling) but there is also mean pooling or average pooling.

The main function of the pooling operator, eliminating redundant information, is taken to an extreme of reducing information, with global average pooling (GAP) [18]. GAP is an incredibly

useful operation that sits at the very end of the convnet and summarizes a whole feature map to a single real number a 1-d tensor of one entry. That can be used to represent a single class log-probability. For example, if we have are doing a 50-class classification task we could obtain the log probabilities of a given input by processing the input with the convnet and in the last layer using a convnet with 50 output channels. The resulting 50 feature maps may be converted to a 1-d tensor of log-probabilities with the use of a single operation of global average pooling, which average all elements of each of the 50 feature maps. This operation is effectively equivalent similar to using average pooling with a window size equal to the size of the input feature map.

In general, pooling is a layer that has no learnable parameters. In order to specify completely the opration one only needs the window size and the operation (max vs. mean, e.g.). This lack of parameters is in part the reason why GAP [18] is used with very good results for classification, where one avoids the overparametrization brought about by the use of densely connected layers (also known as linear layers). Popular networks such as VGG [66], or EnvNets [73] in case of acoustic event detection, make use of a last stage of dense connections for classification. However, it has been known in the community that such topologies offer little advantage and complicate training because of the much greater parameter count, thus, damaging statistical and computational efficiency.

Chapter IV

Methodology

In this chapter, we describe the main methods used for training our environmental sound classifier. We start by describing the two publicly available sound collections, small-scale specialized ESC-50 and the much larger audio-tagging dataset, Audioset. Then, in the next section, we describe and explain the different data augmentation techniques: audio-specific ones and data-agnostic techniques. No matter the type of dataset augmentation its purpose is to bolster generalization and robustness to the classification problem.

4.1 Sound Datasets

In this section we describe the two sound datasets used for this work. The bulk of the experiments were carried out in the environmental sound collection ESC-50. In the latter stage of research, however, a much larger sound dataset was used as a platform from where to pretrain a sound recognition model to solve the ESC problem in ESC-50. This approach of using two datasets is known as trasductive learning and allowed us to achieve state-of-the-art performance on the single-task ESC-50 corpus via supervised pretraining on Audioset.

For this work, two publicly available datasets of natural sounds were used: ESC-50 and Audioset. ESC-50 is actually part of a set of closely-related datasets: ESC-10, ESC-50, and ESC-US. ESC-10 is a proof-of-concept small-scale dataset made from a subset of 10 classes takes from the fifty classes in ESC-50. ESC-50 contains 40 examples per 50 categories amounting to 2,000 audio recordigs. Meanwhile, ESC-US is a much bigger, 250,000 audio recordings, compilation of unlabeled environmental sound. It is a dataset specially designed for the development of unsupervised learning approaches for ESR. The common feature of the ESC family of datasets is that all of them are environmental sounds in the sense defined in section 1.1, thus, there is no music and no speech. All audio samples were taken from the sound sharing project Freesound. On the other hand, Audioset is a large-scale dataset of natural sounds taken from

	ESC-50	**Audio Set**
Number of examples	2,000	1,183,235
Duration of sample	5s	10s
Annotation	A single label	Tags (Multiple labels with no temporal resolution)
Acoustic Contents	Environmental only	No restrictions
Number of classes	50	527
Performance metric	5-fold accuracy	mAP/AUC/d-prime

Table 4.1: Comparison between the sound datasets ESC-50 and Audio Set.

YouTube videos, that is the samples contain music and speech. Just like ESC-50, Audioset is labeled manually, however, besides this point, the aforementioned datasets share very few commonalities. The following table compares the two of them.

4.1.1 ESC-50

ESC-50 is a compilation of 2,000 recordings taken from Freesound project grouped in 50 classes of 5 different categories:

Animal Sounds	Human Sounds	Natural Landscapes	Indoor Sounds	Outdoor Sounds
Dog, rooster, pig, cow, frog, cat, hen, insects, sheep, crow	Baby crying, sneezing, clapping, breathing, coughing, footsteps, laughing, tooth brushing, snoring, drinking/sipping	Rain, sea waves, crackling fire, crickets, chirping birds, water drops, wind, pouring water, toilet flush, thunderstorm	Knocking, mouse click, keyboard typing, creaks (wood/door), can opening, washing machine, vacuum cleaner, alarm clock, ticking clock, glass breaking	Helicopter, chainsaw, siren, car horn, engine, train, church bells, airplane, fireworks, hand saw

Figure 4.1: ESC-50 sound examples are grouped in four distinct semantic categories: animal sounds, human sounds, natural landscapes, indoor sounds and outdoor sounds.

The selection of classes of this dataset is meant to be comprehensive while not too complex while, at the same time, mantaining ease of use. The downside of this approach to design is that it somewhat small; it contains 40 examples per class. If there is substantial intraclass variability then there will be problems with generalization. However, even taking into account all these cons, ESC-50 is widely used precisely because of its simplicity and good coverage of the wide spectrum of what an environemtnal sound is. For practitioners, ESC-50 is quite useful as in

a first stage of the technological development stage and for prototyping general acoustic event recognizers.

Owing also to its adoption in the community is the fact that humans have been tested using in ESC-50 giving as a result 81.3% accuracy. Human listeners apparently have difficulty recalling more some sounds that other, 34.1% and 51% for example for washing machine for crickets while 100% for baby crying or dog and rooster.

All recordings are single channel, 44.1 kHz, in Ogg Vorbis file format, organized in 5 groups called folds so that a fold-cross-validation technique can be used readily. The fold number of a file is part of its name and is used in machine learning workflows to evaluate using 5-fold cross vailation accuracy which is the standard way of reporting results of works that use ESC-50.

As a summary ESC-50 is a widely used dataset with a useful/good set of environmental sounds that have been adopted as the de facto go-to dataset for environemntal sound recognition. The only problem it its small number of examples which could impair generalization. However, researchers with a specific application in mind resort to acquired capturing their own data, see section 3.2. Finally, another solution to this problem that has not been thoroughly studied is to use the unlabeled ESC-US to train an encoder of environmental sounds as a previous stage to apply finally the learned representations to help solve the supervised problem of ESC-50 classification. This has not yet been explored fully, as far as we know and it is sometimes called unsupervised pretraining or unsupervised representation learning, see below Sect. 6.1 for more information.

4.1.2 Audioset

Audio Set [24] is both a dataset and an ontology of audio events. As an sound dataset Audio Set seeks to provide a comprehensive coverage of real-world sounds at the scale and magnitude of what ImageNet provides for images. As an specification of an ontology, the goal is to provide a well-structured hierarchy that can aid human labeling by allowing the human to quickly and directly find the set of terms that best describe a given sound. Given this goal of ease of use it was important during the development of the set of events to add categories without any overlap or duplication. The structured set of audio event categories provided is called the Audio Set Ontology.

It is important to notice that, in constrast to ESC-50 above, Audio Set considers all sound events (environmental, speech and music) rather than a limited domain, e.g., environmental only. Given the importance of creating a general-purpose audio event recognizer, it was necessary to define the set of events the system should recognize (the aforementioned ontology).

Audio Set excerpts were taken from the YouTube Corpus, a set of labeled YouTube segments comprising an identifier, start time, end time and one or more labels. The segments in this dataset are all 10 seconds long except when the underlying video has a shorter duration. It is

important to note that each segment carries one or more ontology class labels.

The overall count of labels is not uniform. "Music" is particularly common, present in 56% of the segments. Single segments can have multiple labels (on average 2.7 labels per segment). For this reason, class imbalance could be an issue when training using Audio Set. Nevertheless, the unbalanced training set is the one more commonly used because of its large scale. The unbalanced train set contains 1,771,873 segments and the evaluation set contains 17,748.

In summary, the Audio Set dataset is a collection of general audio events, comprising an ontology of 632 audio event categories and a collection of 1,789,621 labeled 10 sec excerpts from YouTube videos. The associated ontology is hierarchically structured with the goal of covering all acoustic distinctions made by a typical user.

4.2 Data Augmentation

4.2.1 Sound Data Augmentation

It is costumary to extend sound dataset, be it speech [59], animal [69], urban [60] or environmental [73], by transforming datapoints into others that are semantically and perceptually equivalent. For example, human ears perceive as the same a sample of speech that has been shifted in time. Other times, the transformation retains the same label eventhough the sound may be perceived different or have a different signification. As an example, we can give a common data extension operation: spectral masking. In which, only a part of the whole frequency spectrum of a sound datapoint is preserved and the rest is discarded. Provided that the retained part is informative enough the label of the new datapoint should be the same as the original.

This way we can divide sound augmentation operations in those that give no change in signification and perception and those that do. Adding noise is another example of an operation that introduces or distorts the original signal. In the machine learning community, we have examples of phase shifting [60], spectral masking with temporal masking [52].

Two data augmentation schemes were used in this work. But before any of these where applied a simple preprocessing was introduced. This preprocessing consisted on normalization followed by padding-cropping as we will detail now. Thus, the first step of this preprocessing in our pipeline is normalization. Normalization ensures that the input signal is within the range $[-1, +1]$. $x \rightarrow x/2^{N-1}$ where N is the resolution of the sound waveform in bits, and the exponent is one less from zero because the whole range is used to represent negative and positive numbers, it goes from $[-32,767, +32,768]$. Amounting to a total of $2^N = 2^16 = 65,536$. Usually $N = 8$ for most sound file formats. Thus, in our experiments the first preprocessing step is to divide the signal by $2^7 = 32,768$ to make it more easier for neuronal units initialized with weights around zero and variance of one to process them more easily.

The other step in the preprocessing state is one that turned out to be crucial for attaing good performance, see Results chapter. This preprocessing amounts to the following: let T be the input length of a network seconds. During the training phase, we padded $T/2$s of zeros on each side of a training sound and randomly cropped (selected a segment) a T-s section from the padded sound. In contrast, during the testing phase, we also padded $T/2$s of zeros on each side of a test sound and cropped 10 $T - s$ sections form the padded sound at regular intervals. We then input these 10 crops to the network and averaged all softmax outputs.

Following [74] we also experimented with a regime of strong sound augmentation. Whereas, in addition to zero padding and random cropping, scaling was performed by multiplying with a factor that is randomly selected between $[0.8, 1.25]$ and scaling by increasing the gain with a factor randomly selected from -6 dB to $+6$ dB. Crucially, scaling was performed before zero padding and gain augmentation was performed just before feeding the sound to our model (after padding).

Previous works have achieved time-invariance through the use of convolutional boltzmann machine [36], convNets [51], [31]. At other times, convolution can be shuned and a multilayer perceptron can model phase-invariant filters when it's overcomplete or overparametrized [75].

It is important to note that this "costume" has a purpose and its function is to increase percentage points in accuracy the performance of sound recognition because it creates new data, and so it is especially useful when working with datasets of limited size, which is often the case given that high-quality sound recordings are with high-quality labels are very expensive to get.

No noise addition was used since that is more of the speech recognition community and some environmental sounds are very similar to noise, in its spectral qualities and perceptual qualities. So to avoid cross-recogntion among classes and so on noise is costumarily not added to environmental data.

4.2.2 Modality-independent Augmentation

Modality-independent augmentation, or mixup, is a technique to regularize model training by augmenting the dataset with virtual input-target pairs $(\bar{x}, \bar{y})$. It was first proposed at ICLR 2018 by [83]. Since then, this technique has proven useful in the training of large models on large datasets, see for example [38]. Also, as a way to increase robustness to noise and a form of regularization [83].

Mixup is based on the principle of Vicinal Risk Minimization (VRM) [13] where a heuristic of linearity is employed to create a neighborhood around each datapoint. Such linear constraint smooths out the classification boundary between examples of different classes. In contrast to other data augmentation methods, mixup is independent of the data modality, it can be applied equally well to images, sounds, or tabular data because of its inherent generality.

To introduce the principle of VRM, consider the standard formulation of neural network training, Empirical Risk Minimization (ERM), [79]. In the context of ERM for supervised learning, there is a data-generating distribution P, that ponders or weights the loss function associated with mistaken outputs, those outputs or predictions that do not match with the true label, resulting in the following integral:

$$R_\delta(f) = \int \ell(f(x), y) \mathrm{d}P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(f\left(x_i\right), y_i\right) \tag{4.1}$$

where R is the expected risk or expected error, and $\ell(\cdot, \cdot)$ is the loss function that increases when the prediction of x, $f(x)$ diverges from its true value y. However, in this case R is replaced by R_δ because now the empirical distribution is used rather than the true data-generating distribution $P(x, y)$. Moreover, the integral is replaced by a sum because the continuous random variables are sampled into discrete values.

The problem of ERM is that, although there exist a well laid out theory supporting it, [78], most modern neural network models do not satisfy the conditions required for the theory to be applied. Specifically, training by minimizing 4.1 is guaranteed to converge if the model size stays the same, that is, if the model capacity is not increased as the dataset increases. Modern works, however, see do not set model capacity to a fixed value, and rather increase model size as dataset size increases. Nevertheless, even in those conditions model performances are seen to improve. What is needed is a form of regularization of the model capacity so that the supporting classical theory can be applied.

In order to solve that issue, VRM [13] replaces the naive distribution estimate P_δ with vicinal distribution $v(\cdot, \cdot|x, y)$:

$$R_v = \frac{1}{n} \sum_{n=1}^{n} v(\bar{x}, \bar{y}|x, y) \tag{4.2}$$

where

$$\bar{x} = (1 - \lambda)x + \lambda x' \tag{4.3}$$

$$\bar{y} = (1 - \lambda)y + \lambda y' \tag{4.4}$$

The mixup vicinal distsribution can be understood as a form of data extension that encourages the model o behave linearly the regions between two different training examples. In [83] it is argued that this linear behavior reduces the amount of detrimental oscillations of the loss function that ultimately leads to uncertain class predictions, especially when classifying outside the traning distribution. Linearity is also a good intuitive bias from the perspective of Occam's razor since it is one of the simplest possible relationships. Because these reasons we apply mixup to our training especially when dealing with large sampling frequencies (high resolution data) and large models. In these cases, models trained with mixup are more stable in terms of

model predictions and gradient norms for in-between training examples.

4.3 Architecture and Model Scaling

4.3.1 Architecture

The model architecture proposed is called AclNet, taken from Acoustic CLasification Network, and it is an end-to-end acoustic pattern classificator. It is fully convolutional and so it accepts input sounds of all sizes. It process the variable-sized inputs by way of two sequential stages. The first of these stages consists of one-dimensional convolutions designed to extract a spectrogram-like representation of the input. The second stage, then, process this 2-D representation in a manner similar to other image-processing convnets and outputs a vector of class probabilities.

One of the defining characteristics of AclNets is that both stages are trained simultenously. Thus, the feature extracted by the first module is optimized for the task at hand and not based on heuristics or rules of thumb as is in feature engineering or feature learning approaches. Higher levels of performance are hence possible because the feature extraction is fine-tuned to the problem of acoustic detection just as we described in Section 3.3. However, in order to fully utilize the fine tuning of features it is important to keep generalization error low by avoid overfitting, that is memorizing the dataset examples. This is why end-to-end learning systems, and their increased number of parameters require more regularization. This kind of regularization is provided in our training procedure by the following strategies:

1. data augmentation for sound
2. vicinal risk minimization (mixup)
3. weight decay, or L_2 penalization
4. dropout operations between layers

These are the main techniques to avoid overfitting in our training procedure and models. These regularization measures improve performance since we don't see a degradation of the test accuracy during training, we only found overfitting when the model capacity is greatly increased through the application of a 1.5 width multiplier. Please see chapter 5 for further details about training curves (Sect. 5.1) and scaling outcomes (Sect. 5.2).

In between the feature extraction and feature classification stages we have a transposition operation. That allows the neural network to work with the intermediary tensor output tensor

of the first stage to be processed as an image, or grid-like tensor. This transposition changes the order of the dimensions of the tensor, interchanging the output channel dimension with the

In terms of the properties of the architecture we can say in a high-level summary succintly that it takes inspiration from two other networks: DaiNet [18] and EnvNet [73]. EnvNets introduced the use of a the filters of 1-d convolutions as a learnable filterbank, transposition and a Vgg-like image-processing back-end for the last classification stage. From DaiNet we took inspiration of a fully convolutional network (basically fully connected layers are replaced by global average pooling) and aggresive pooling in the 1-d stages and a small filter size, to lower parameter requirementes and enhance statistical and computational efficiency. The result is a scalable, end-to-end, modern network that can be used for constrained applications or for transfer learning in case of big problems to solve.

A particularly intriguing way to put the overall design of AclNet is by the formula: EnvNet+ DaiNet^2 = AclNet. Note that DaiNet [18] is squared because, this network is a purely one-dimensional convnet, thus we translated the convolutional hyperparameters to its two-dimensional equivalents. For example, instead of using (3) kernels we used 3x3 kernels and so on. Or the same thing with pooling layers window sizes. Also note that this is an informal way to put the relationship between the network and the plus sign is not really an sum it is simply the combination of two very different design paradigms.

Traditionally, within the field of deep learning, acoustic modeling is divided into two distinct parts: (1) the design of a feature representation of the audio data, and (2) building a suitable discriminative model based on that representation. Nevertheless, it is often quite challenging and labor-intensive to find an adequate representation during the first phase above, the so-called "feature-engineering" process. Additionally, the heuristically designed features might not be optimal for the discriminative task. By using simpler features, deep neural networks can be viewed as extracting feature representation jointly with classification, rather than separately [59].

In order to maximize the representation learning in the convolutional layers, the proposed network architectures (AclNets) are fully convolutional, that is, without fully connected layers, only convolutional blocks consisting of batch normalization [35] and ReLU activation functions interspersed with some pooling layers. Also, fully convolutional topologies can be applied to input audio of varying lengths. Through the application of batch normalization and a careful design of down-sampling layers, the difficulties in training very deep models are overcomed while keeping the overall computation cost at an adequate low.

The AclNet architectures take as input raw sound waveforms, represented as a long one-dimensional vector, instead of the traditional hand-tuned features or specially-designed audiograms. While designing AclNet the key design elements taken into account were:

Fully convolutional topology: High-dimensional fully connected (FC) layers are present in most deep convolutional networks for classification. Typically 2 or more fully connected (FC) layers of dimensions 4096 as in [66], [41] for discriminative modeling, leading to a very high

number of parameters. Here it is presupposed that most of the learning occurs in the convolutional layers, and with a sufficiently expressive representation from convolutional layers, no FC layers are necessary. Therefore, a fully convolutional design is proposed [**?**], [46]. Instead of FC layers, a single global average pooling layer is used which reduces each feature map into a single floating point number through an averaging of all activations along the temporal dimension. This way the network is forced to learn good representation in the convolutional layers, by removing the fully connected, potentially leading to better generalization.

A large first layer-receptive field: The first receptive field or kernel size in the first convolutional layer is very important since a very large or a very small receptive field won't be able to capture enough information for subsequent layers. Thus, similar to the window size for many MFCC computation, the first layer receptive field is designed to cover a 10-millisecond duration of the input. A much smaller or larger receptive field was found to give poor performance. On the other hand, if a small receptive field is used for all convolutional layers such as in [66], which uses 3x3 in pixel for all layers, a lot more layers would be needed in order to extract high level features. That substantial increase in depth could be computationally very expensive. Finally, the sampling rate of the audio affects the receptive field size in the first layer, since a field size of 80 at 8kHz sampling rate is at a different length scale than at 16kHz sampling rate.

Very deep networks: In order to build very deep networks, a very small receptive field (3x3) is used for all layers but the first 1D convolutional layers. Historically, 3x3 receptive fields were first popularized by [66] for natural images. This design reduces the number of parameters in each layer and controls the model sizes and computation cost as depth is increased. Furthermore, we aggressively reduce the temporal resolution in the first two layers by 16x with large convolutional and max pooling strides to limit the computation cost in the rest of the network [70]. After the first two layers, the reduction of resolution is complemented by a doubling in the number of feature maps. In the visual domain this change in resolution and the number of features maps leads to more specialized filters at the higher layers (e.g., feature maps responding to faces) and more basic filters at the bottom (e.g., feature maps responding diagonal lines). We use rectified linear units (ReLU) for lower computation cost, following [82], [66].

Batch Normalization: Some auxiliary layers called batch normalization (BN) [35] that alleviates the problem of exploding and vanishing gradients, a common problem in optimizing deep architectures, are adopted here. BN normalizes the output of the previous layer so the gradients are well-behaved. This makes possible training very deep networks (M18, M34-res) that were not studied previously [62]. Following the example in [35], we apply batch normalization on the output of each convolutional layer before applying the ReLU non-linearity.

4.3.2 Model Scaling

The principal goal of all scaling methodologies is to find an optimal point in the trade-off between accuracy (or, more generally, performance) and model cost (in size and latency). In this

setting, optimal is taken to be a function of the desired application and deployment platform. So there is no way a researcher can say in advance what the best operating condition will be for a given application. Only we can hope for in to have the best performance with the least computational cost. And if we think in terms of the graphs shown in Section 5.2 the most efficient models are those in the upper left corner of a plot of parameter count vs. performance, or, alternatively, operation count vs. performance.

Performance metrics are also application-dependent. In certain cases, latency is a better measure than simply operation count. In others, the accuracy is not the best way to measure performance and because a low false-positive rate is critical or very important.

Through the use of certain **scaling hyperparameters** models can be increased in size (scaled up) or decreased in size (scaled down). Prominent among these hyperparameters is the **width-multiplier** which is very simple and convenient.

Another simple scaling procedure is to increase the **resolution** of the input data. In the case of images, the resolution is measured in the number of pixels. In the case of sound, the "resolution" corresponds to the sampling rate. A high sampling rate equals a better quality of the sound for a hearer. Thus, the sampling frequency is, in fact, a scaling hyperparameter. And similarly, we can increase it or decrease depending on whether what we want to optimize: performance or model size.

There are other less straightforward scaling procedures. One of this is the use of compact low-resource cheap versions of convolutions. An standard convolution applies a single filter to all input channels and then sums up the resulting output of each convolution. However there is a lot of wasted computation if we can do the same operations by factorizing the application to all filters and summation operation that we just described.

4.4 Training and Evaluation

Datasets for machine learning, being usually too large to fit unchanged in memory, are usually partitioned in equal-sized portions named mini-batches, or just batches. The partioning of the datasets, training and evaluation datasets, is very important in order to understand the training procedure of machine learning models that process large amounts of data. Specifically, the training process is a doubly iterated procedure, a loop within a loop. One outer loop iteration is called an epoch and an epoch represents a single pass through the whole training dataset. Thus, to run through a single epoch it is necessary to run through number of batches = dataset size / batch size times. This way inner loop is indexed by the batch number. The batch size is inversely related with the dataset size.

The training of AclNet is done with the cross-entropy criterion using momentum stochastic gradient descent. Weights of EnvNets were initiliazed randomly. This is partially because it is

reported that handcrafted weight initialization such as gammatone initialization [61] does not notably improve the classification performance [75], [31]. However, the main purpose is to learn another feature representation that complements handcrafted features such as mel filterbanks.

In all of our experiments, we used stochastic gradient descent with momentum of 0.9, weigth decay of $2 \times 10^{-}4$, and a batch size of 64. We trained the model using the following learning rate schedule with 3 different phases: 0.2 for the first 500, 0.04 for the next 1000, and 0.016 for the last 500, for a total of 2,000 epochs for each of the five folds. Also, for the first 100 epochs we disabled mixup as a form of warming up weight actualization and improve initial convergence. To improve convergence, we used a $0.1\times$ smaller learning rate for the first warmup epochs (10). We then terminated training after 1,000 epochs. We doubled the epochs and the learning rate schedule when using mixup, as we mentioned in the Results section.

We apply 50% of dropout [67] to the convolutional layers to prevent overfitting with a low value of 10%. Compare this value to the one commonly used for fully connected layers, 50%. This is in part because of the weight sharing feature of convolutions as we discussed earlier. In addition, we apply batch normalization [35] to all the convolutional layers to accelerate learning.

In ESC-50, there is no test set, rather only a validation set that iterates across different folds. That is, validation is done via the procedure known as fold cross-validation. The dataset itself is already preconfigured split in five folds so that this validation procedure is standardized. Each sound file has in its name a number between zero and four corresponding to the subset or fold in which it belongs. A sound file can only belong to a single fold making these five folds a partition of the whole set. This pre-established fold splitting of the dataset is very convenient for comparison across different models and training procedures.

4.5 Transfer Learning

A recurrent working hypothesis in the deep learning community and, especially, in computer vision is that network architectures that perform better on a large dataset like ImageNet necessarily perform better on similar datasets and in similar tasks. Examples of similar vision tasks are: classification on different datasets [65], [21], image segmentation [28], [14] or object detection [33]. A related assumption or hypothesis is that better performing network architectures can lead to learning better features that can be transferred across semantically similar tasks. In this work, we test the aforementioned hypotheses in the context of environmental sound recognition and to test the transferability of a large-scale model (our AclNet) trained to extract relevant features from the large-scale dataset to the field to ESC-50. The result of this investigations will allow us to ascertain the transferability of both AudioSet features and our AudioSet classification architecture (the AclNet architecture).

We demonstrate the training of AclNet in Fig 4.3. x_0 is training input, an element of the dataset AudioSet, and y_0 is the target of the associated element of AudioSet. $FC_{AudioSet}$ is the

head, a fully connected layer for AudioSet tagging. There are several options available to us when we one says that one is going to transfer from one dataset to another. To be more precise, we present to different strategies that were implemented in this work in the following list:

1. Training an audio-tagging model from scratch. All parameters are randomly initialized. The model is an instance of AclNet, except for the final fully-connected layer which depends on the task dependent number of outputs. It is important to note that this particular configuration is used as a baseline to be compared with other transfer learning strategies, since it is no real transference of knowledge, because of the random initialization and because of training from scratch.

2. AclNet is used as a feature extractor. For the new ESC-50 task, the embedding features of audio waveforms are calculated by using AclNet. Then, the embedding features are used as input to a classifier, such as a fully-connected neural network. When training on the new task, the parameters of the AclNet are frozen, held constant and not trained. Only the parameters of the classifier built on the embedding features are trained. Fig. 4.3b) shows this strategy, where ESC-50 is the new task, and FC_{ESC-50} is the fully connected layer for the new task. The AclNet is used as a feature extractor. A classifier is built on top of the extracted embedding features.

3. Fine-tune AclNet. AclNet is used for the new task, except the final fully-connected layer which is randomly initialized. All parameters are fine-tuned on the new task. Fig 4.3 c) demonstrate fine-tuning in our AclNet transfer framework.

Much of the analysis in the corresponding section in the Results chapter (Sect. 5.4 requires comparing accuracies across the two datasets of different levels of granularity. When fitting a neural network model to accuracy performance across multiple datasets, it is costumary to consider the effects of model and dataset to be add up, as if their effect were independent of each other but interact through a simple sum. Therefore, using the raw five-fold accuracy of ESC-50 as a variable used for comparison can be problematic. In particular, an increase of one percent in accuracy is different if it is relative to a baseline accuracy. In cases like this, the logit transformation is commonly employed [40] for the analysis of proportions, like accuracies, and the linear additive change δ in logit-transformed accuracy has a simple interpretation as a multiplicative change as $\exp \delta$ when a correct prediction is made. We now give the logit transfomation as follows:

$$\text{logit}\left(\frac{n_{\text{correct}}}{n_{\text{correct}}+n_{\text{incorrect}}}\right)+\delta=\log\left(\frac{n_{\text{correct}}}{n_{\text{incorrect}}}\right)+\delta=\log\left(\frac{n_{\text{correct}}}{n_{\text{incorrect}}}\exp(\delta)\right) \tag{4.5}$$

As an added bonus, results after the logit transformation do not depend on whether performance is measured in terms of accuracy or error rate, because $\text{logit}(p) = -\text{logit}(1-p)$. With this in mind, we end this section by mentioning that, as will be shown in the next chapter, our

results reveal clear advantages of transferring features and architectures in the context of sound recognition. AudioSet weights provide a starting point for features on a the new classification task. However, perhaps what is needed is a way to learn how to adapt features, similarly to the problem of zero-shot or few-shot learning.

Layer	Stride	Out dim	Out Chans	Kernel size
Conv1	S1	$C1, 1, 20480/S1$	C1	9
Conv2	S2	$64, 1, 20480/(S1S2)$	64	5
Maxpool1	1	$64, 1, 128$	64	$160/(S1S2)$

Layer	Stride	Out dim	Out Chans	Kernel Size
Conv3	1	$32, 64, 128$	32	3×3
Maxpool2	1	$32, 32, 64$	32	2×2
Conv4	1	$64, 32, 64$	64	3×3
Conv5	1	$64, 32, 64$	64	3×3
Maxpool3	1	$64, 16, 32$	64	2×2
Conv6	1	$128, 16, 32$	128	3×3
Conv7	1	$128, 16, 32$	128	3×3
Maxpool4	1	$128, 8, 16$	128	2×2
Conv8	1	$256, 8, 16$	256	3×3
Conv9	1	$256, 8, 16$	256	3×3
Maxpool5	1	$256, 4, 8$	256	2×2
Conv10	1	$512, 4, 8$	512	3×3
Conv11	1	$512, 4, 8$	512	3×3
Maxpool6	1	$512, 2, 4$	512	2×2
Conv12	1	$50, 2, 4$	50	1×1
Avgpool1	1	50	50	2×4

Figure 4.2: The upper table includes AclNet low-leve features, with 1.28s 16kHz samples as input. An input dimension of 20,480 samples. (S1,S2,C1) are the stride of the first and second layer, and the number of channels of the first layer, respectively. We tried different values to ascertain the optimal values of these hyperparameters. The last kernel size is adjusted based on the stride in order to output a fixed dimension for the next section.

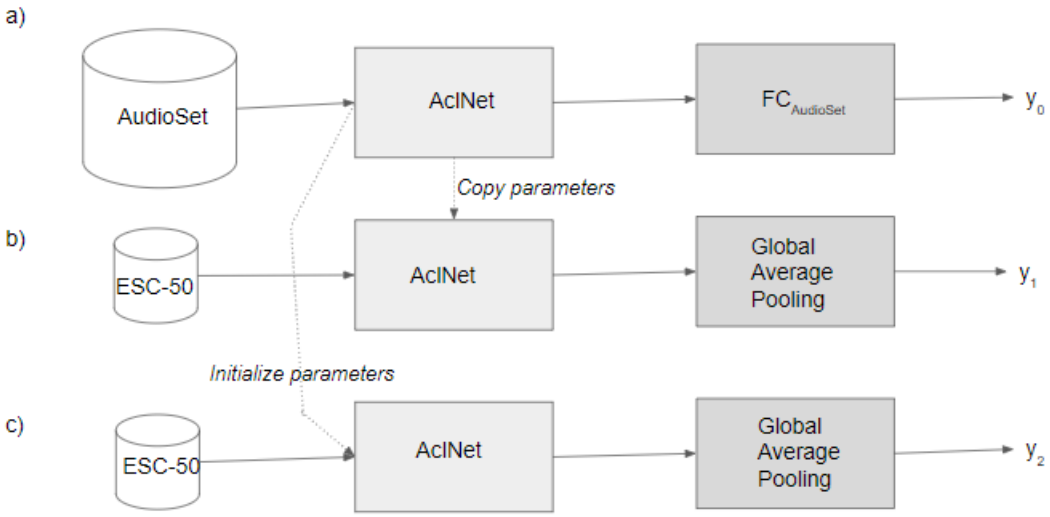

Figure 4.3: a) AclNet is pretrained with the AudioSet dataset. b) For the new task, AclNet is used as a feature extractor. A classifier is built on the extracted embedding features. The shaded rectangle indicates the parameters are frozen and not trained. c) For the new task, the parameters of a AclNet are initialized with a AclNet pretrained on AudioSet. Then all parameters are fine-tuned to the new classification task.

Chapter V

Results and Discussion

In this chapter we discuss and present the most important results of our work. First, the learning curves which are plots of training accuracy versus training cycles are presented in order to visualize the improvement obtained by our training methods and architectural choices. Second, the results on scaling up and down the AclNet architecture is done to obtain a family of models that may have lower accuracy but be more efficient with parameters and operations. Finally, in the last two sections we compare the state of the art and present the results of transferring AclNet features from AudioSet to ESC-50.

5.1 Initial Experiments and Learning curves

First, we conducted experiments with the aim of finding the best hyperparameters for our low-level feature extraction system in order to demonstrate the effectiveness of raw signal training. These initial experiments were performed on the validation set.

Particularly, we compared the accuracy for different values of the input length of the sound files. Candidates for this length were chosen between $0.5, 1.0, 1.5, 2.0, 2.5, 3.0$ seconds, following [73]. In the latter work, the authors first simplified their network for raw feature extraction so that the low level features were obtained with a single convolutional using a filter size of sixty-four, the same filter size used for logmel architectures [53]. In contrast, we do not simplify the network and use the full low-level feature extraction showed in Fig. **??**.

We were able to reproduce the result in Tokozume et al. [73], by finding that accuracy is at is highest when the input length is between 1s and 1.5s, with a very small difference of performance within that range. Very large (5s) or very small (0.5s) input lengths degrade performance quite considerably, suggesting that the random cropping procedure we did later (see Sect. 4.2.1) is essential for the model to receive enough information without overfitting. Too few informa-

tion (as in 0.5s of input length) and learning cannot proceed with enough accuracy and speed. Conversely, too much signal length (5s, e.g.) and overfitting ensues. Following these results, we set the input length to be 1.5 seconds or equivalently 72,000 samples when using a 48kHz sampling frequency.

In Tokozume et al. [73], analysis showed that two convolutions of kernel size 8 worked best for this dataset. So it is an indication or corroboration of current results. our own results. Our experiments confirmed that two convolutions being optimal, but we also found that slightly reducing the kernel size of second convolution had no impact on accuracy. Our best setting is with kernel sizes of nine and five for the first two convolutions.

In order to determine the choice of other low-level feature parameters we did a grid search of the parameter space over the ranges: number of channels $8, 16, 32$, stride in the first layer $2, 4, 8$, and stride in the second layer $2, 4$.

We trained AclNet using both standard convolution (SC) and depth-wise separable convolutions (DWSC) settings with width multiplier of 1.0, and found the values of (C1, S1, S2) = (8,2,2) for SC and (16,2,4) for DWSC gave the best accuracy. For the remainder of experiments, we will default to using these best settings for SC and DWSC. The experiments showed that there was about a 3% difference between the best and worst parameters for each of the settings. The best result in both cases was not the highest complexity, which is (32,2,2). We suspect the heavier low-level feature network settings might be overfitting, and that with more training data we could reach a different conclusion.

In theory, data augmentation is a useful way to prevent a system from overfitting and help regularize (and improve accuracy) especially when training large models. We apply mixup, see Sect. 4.2.2, and a set of sound-specific augmentation techniques, Sect. 4.2.1. In figure 5.1 we provide experimental evidence that it is indeed the case that sound data augmentation and mixup are effective in improving performance both, separately (curves green and red) and combined (purple curve). This allows us to conclude that, used in combination, both forms of augmentation the difference in training they make is beneficial and greater than the one obtained when using any one of them separately.

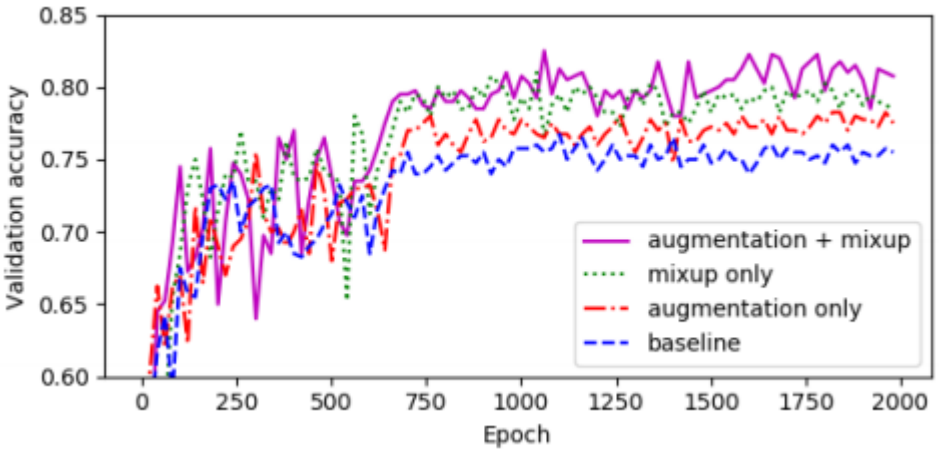

Figure 5.1: Comparison of four learning curves. Note the beneficial effect of sound augmentation (broken red curve) , and mixup (broken green curve) and both combined (purple). Baseline here means the same architecture without mixup and without sound data augmentation. Test accuracy saturates roughly at the same time (750 epochs) in all cases.

All experiments shown in Fig. 5.1 were done using a width multiplier of 1.0 and standard convolutions. As mentioned previously, we see an obvious improvement with each individual augmentation, and that mixup by itself is more effective than the other form of augmentation. The best result was achieved when augmentation was combined with mixup, which had an absolute improvement of more than 5% above the baseline without any augmentation. We note mixup is conceptually similar to between class learning [74], which was also shown to work well for ESC-50, but not in our experiments.

Finally, we experimented with the choice of the hyperparameter α in mixup, and found that values between 0.1 to 0.2 worked well for the larger size architectures, thus for the remainder of experiments, we default to using this combined augmentation with mixup $\alpha = 0.1$.

The learning curves in Fig. 5.1 show the relationship between the performance and the number of training epochs. Previous works [59], [42], [18] described experiments were conducted using different numbers of training epochs. The figure shows that validation accuracy of EnvNet with various number of training epochs. This figure shows that standard learning approximately 750 training epochs are sufficient for ESC-50; and this number is sufficient for other training configurations. This is in contrast to what [74] found for their mixing method, between-class learning: the performance of it was lower than that of standard learning when using less than 600 training epochs. Between-class learning improve accuracy only when a sufficiently large number of training epochs is used. The number of training epochs also increase when there are many classes. In contrast, our methods do not require larger training times, but the convergence time do increase with the number of classes, as we saw when training AclNet with AudioSet, see Sect 5.4.

Sampling rate	Conv type	LLF params (k)	LLF MMACS	HLF params (k)	HLF MMACS	Total params (k)	Total MMACS	Width multiplier	Accuracy (%)
16k	DWSC	1.44	4.35	13.91	2.93	15.35	7.28	0.125	75.38
16k	DWSC	1.44	4.35	153.43	31.07	154.87	35.42	0.5	80.40
16k	DWSC	1.44	4.35	567.92	113.7	569.4	118.1	1.0	80.90
44.1k	DWSC	1.81	17.98	13.91	2.96	15.72	20.94	0.125	75.50
44.1k	DWSC	1.81	17.98	153.43	31.33	155.23	49.31	0.5	81.75
44.1k	DWSC	1.81	17.98	567.92	114.6	569.73	132.59	1.0	83.10
44.1k	SC	6.99	80.9	77.21	8.88	84.21	131.17	0.125	82.30
44.1k	SC	6.99	80.9	1190.0	132.72	1197.0	255.01	0.5	83.95
44.1k	SC	6.99	80.9	4730.0	524.67	4737.0	646.97	1.0	85.0
44.1k	SC	6.99	80.9	10620	786.56	10627	867.45	1.5	**85.65**

Figure 5.2: ESC-50 five fold accuracies with AclNet at select configuration of parameters and mult-adds. Note the degradation of performance with decreasing width and resolution. Beneficial tradeoffs can be made in certain circumstances, e.g. 81.75% accuracy at almost 10× less parameters (sixth row).

5.2 Model Scaling

In this section we first investigate the effects of depth-wise convolutions as well as the choice of shrinking by reducing the width of the network and the resolution of the input sound (sampling frequency) based on the two hyper-parameters introduced in Sect. 4.3.2: width multiplier and resolution multiplier. We then investigate AclNets capacity to scale down and still classify accurately within certain budget constraints.

First, we show results for AclNet with depthwise separable convolutions compared to a model built with full convolutions. In Table 5.2 we see that using depthwise separable convolutions compared to full convolutions only reduces accuracy by about 1% on ESC-50 was saving tremendously on multiply-adds and parameters.

Table 5.2 shows the accuracy, computation and size trade offs of shrinking the AclNet architecture with the width multiplier. Also, table 5.2 shows the accuracy, computation and size tradeoffs for different resolution multipliers by training members of the AclNet family with reduced input resolutions. On the other hand, based on the Fig. 5.3 it can be said that accuracy drops off smoothly across resolution. Also, as can be seen in the Fig. 5.3 accuracy drops off smoothly until the architecture is made too small.

To understand the tradeoff between complexity and accuracy we ran three sets of experiments using 1) 16kHz input with depth wise separable convolutions, 2) 44.1kHz input with DWSC, and 3) 44.1 kHz input with standard convolution (SC). For each set, we did the 5-fold validation with width multiplier configured at 1/32, 1/16, 0.125, 0.25, 0.5, 0.75, 1.0 and even greater than one, 1.5 and 2.0. Figure shows the accuracy versus the mult-adds for each of the settings, color-coded by sets. For each of these settings, increasing complexity generally led to better accuracy. The exception is at the highest width multiplier, where it is possible that

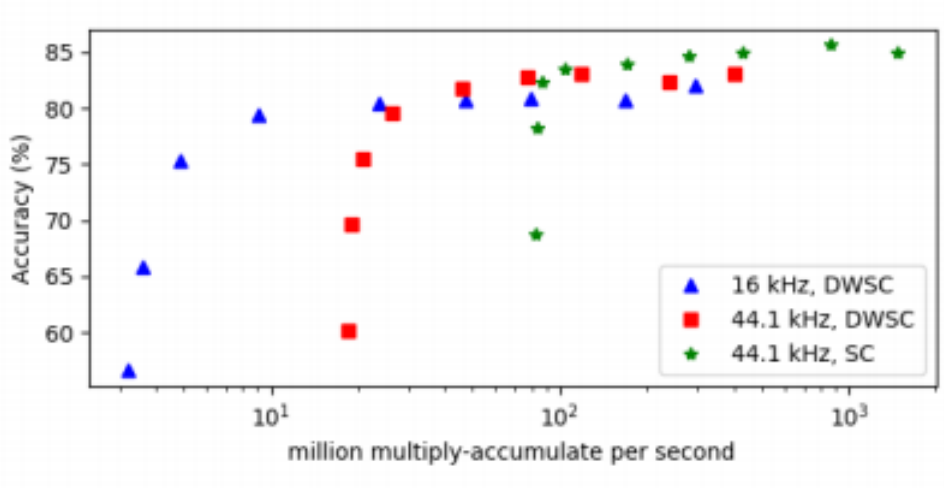

Figure 5.3: Graceful degradation of 5-fold accuracy for different resolutions and convolutional operations. The greater the curve abuts to the upper left corner the better the tradeoff one can obtain by scaling down a given model. In particular, the blue curve provides the best tradeoffs in general (16 kHz and depth-wise convolution.

we encounter diminishing returns. In all cases, the width multiplier below 0.25 accentuates the drop in accuracy. Another observation is that for the same high-level features settings, 44.1 kHz sampling rate improves accuracy by around 2%.

Table 5.2 shows a subset of these experiments, with details of low-level features and high-level features overall complexity and accuracy. Our best accuracy of 85.65% was acheived with 44.1 kHz sampling rate, standard convolution and 1.5 width multiplier. At the time of this writing, this is the best single system accuracy reported for ESC-50 (second overall behind an ensemble system [58]). With depth wise separable convolution models, we can see that the total parameter and mult-adds are significantly lower than standard covnoltuion for the same width-multiplier. The result on 44.1 kHz, depth-wise and 0.5 width multiplier has 81.75% which exceeds human accuracy of 81.3% [54], was achieved with only 155k parameters, and 49.31 mult-adds. We note that human accuracy is also exceeded with standard convolution, width multiplier of 0.125, a model that has a modest 84k parameters and 131.17 mult-adds. As a comparison of complexity, EnvNet-V2 [74], which at the time has the best single model accuracy of 84.9% uses 101 million parameters and 1033 mult-adds. Our best model with accuracy of 85.65% has about a tenth of the parameters and sixteen percent less operations.

5.3 Comparison with the State-of-the-art

With the previous initial experiments we determined the detailed hyperparameters of our front-end system. Now, we compare the performance of our completed design to logmel-CNN [53],

Model	Accuracy
AclNet (42 kHz, SC)	**85.65**%
EnvNet-v2 (best) [74]	84.90%
CNN pretrained on AudioSet [42]	83.50%
Human accuracy [54]	81.30%
logmel-CNN (baseline) [53]	76.90%
Soundnet [4]	74.20%

Table 5.1: Leaderboard of the best models for ESC-50 classification. The accuracies are standard five-fold accuracies. Source: `https://github.com/karolpiczak/ESC-50`. Retrieved Febraury 2020.

human accuracy, EnvNet-v2 [74], the transferred network of Kumar et al. [42], SoundNet [4] on the validation set, which at the time of writing were the best performing neural network architectures. As we can see in the table, our model beats the current state-of-the-art by 0.75% accuracy percentual points. This result is noteworthy because it adds weight to the hypothesis that end-to-end networks are beneficial in fact, superior to hand-crafted feature extractors and discriminators. Moreover, this result indicates that our network model learns a set of features capable of surpassing the classic mel-filterbank.

In [73] the authors proposed extending their CNN that takes raw waveform with a logmel-CNN that takes logmel features. Here we do not combine networks because ours is too big by itself and that would stray us off from one of our main goals which is use as few parameters as possible. For this reason the use of ensembles was not tried in this experimental work, but, we hypothesize that it will be of little value given the considerable gap between the logmel-CNN and our results (approximately 9%). However it would be interesting to test this hypothesis in the near future.

As is customarily done, our model was evaluated with a 5-fold cross-validation scheme with a single training fold used as the validation set; thus, each model trained with 1,200 samples, 40 per class divided by 5, one for each fold. We used the fold decided by the designer of the set, Piczak [54]. Finally, we will see in the next section that architecture beats the non-neural network SOTA of 86.5% and for quite a wide margin with supervised pretraining meaning that the features that it extracts are good fit for a transfer. Or equivalently that the tasks are very similar semantically.

5.4 Transfer learning

To investigate the generalization ability of AclNets, we transfer this family of neural networks to ESC-50 classification task. Table 5.4 shows the 5-fold cross-validation accuracy of AclNet system. Sailor et al. [58] proposed a state-of-the-art system for ESC-50, achieved an accuracy of 0.865 using unsupervised filterbank learning with a convolutional restricted Boltzmann

	STOA [58]	Scratch	Fine-tune	Freeze
Acc.	0.865	0.843	**0.955**	0.922

Table 5.2: Comparison between the work of Sailor et al. [58] with the best results so far, and our different transfer methodologies, as explained in Sect. 4.5

machine. Our fine-tuned system achieves an accuracy of 0.955 outperforming previous state-of-the-art systems by a wide margin. The feature extraction systems that freeze the earlier stages of AclNet achieve accuracies of 0.922. On the other hand, training our model from scratch achieves an accuracy of 0.843. Using an AclNet for the initialization of the feature extractor achieves the best performance. Finally, both the fine-tuned model and the model using the AclNet as a feature extractor outperform the models trained from scratch.

Numerous previous works have demonstrated the superiority of features learned from data over generic, hand-engineered features. Later it was found, that given the availability of large quantities of training data provided by large-scale datasets like ImageNet, features learned by convolutional neural networks could substantially outperform previous feature-extraction methodologies. It became evident that representations also provided accuracy gains over hand-engineered features when transferred to other tasks.

Our results reveal a clear advantage of transferring features among different natural sound datasets. However not all features are transferable as we shown when the weights are held constant. The best fixed acoustic features do not come from the best AclNet trained. Fine-tuning these best models improves performance on ESC-50 dataset. Surprisingly, however, the value of the architecture proposed persists showing us that it is capable of extracting meaningful representations from raw sound waves.

To summarize we provide our main results in the list:

- Our best transfer learning approach is not the one that keeps the acoustic features fixed obtained during pretraining. However, it is important to note that features from AclNet trained on AudioSet consistently outperform features obtained when training from scratch on AudioSet.
- Our architecture transfer well across audio-tagging and audio classification tasks even when weights are best fine-tuned. On a small fine-grained classification dataset like ESC-50, fine-tuning does provide the expected benefit over training from random initialization.
- When fine-tuning is introduced, AudioSet accuracy has a much stronger value, (accuracy = 95%) with a state-of-the-art performance in ESC-50 classification task, indicating a strong success of transference across both datasets.

Given the differences between AudioSet and ESC-50, it is not entirely surprising that features learned on one dataset benefit from some amount of adaptation when applied to another.

On the other hand, it is surprising that features learned from a large dataset cannot always be profitably adapted to datasets that are much smaller, as is the case when weights remain frozen. Importantly, AudioSet weights do provide an important starting point for features on ESC-50 classification. Moving forward, perhaps in the future what would be needed is new methods that change the way weights are adapted from one task to another. This issue is related to the problem of few-shot learning [56]. Finally, it remains to be seen whether methods can be developed to adapt representations learned fom AudioSet to obtain larger benefits across different sound datasets and tasks, especially through the use of self-supervision [37].

Chapter VI

Conclusions and Further Work

6.1 Conclusions

We have presented a novel end-to-end convolutional neural network architecture, AclNet, for audio classification. AclNet is a scalable architecture that achieved state-of-the-art, 85.65% accuracy with high compute and better than human level accuracy of 81.75% with only 155,000 parameters and 49.3 million mult-adds. To achieve the low complexity with high accuracy AclNet used depth-wise separable convolution blocks. The combination of mixup and data augmentation helped boost the accuracy by 5%, which had a major contribution to achieving one of the best results reported on the ESC-50 dataset.

Moreover, we proposed a learning method for deep sound recognition that includes regularization via mixup training and avoids overfitting through the use of sound data augmentation. Our method improved the performance on various scales, two datasets, and data augmentation schemes (strong and weak). Moreover, we achieved a performance that surpasses the human level by constructing a deeper network named AclNet and training with our learning methodologies. The learning approaches implemented here are simple and powerful allowing improvement to the problem of environmental sound classification.

Additionally, we proposed new model architectures called AclNet based on depthwise separable convolutions. Some of the important design decisions leading to an efficient model were discussed. In order to build smaller and faster AclNets we demonstrated the viability of experimenting with the width multiplier and resolution multiplier, the scaling hyperparameters, to find a reasonable tradeoff between cost and performance. We concluded by demonstrating AclNet's effectiveness when applied to transfer.

Finally, we assume that the core idea of raw feature extraction is generic and could contribute to the improvement of the performance of tasks that process sound without the need of computing arbitrary filterbanks or other manually engineered features.

We propose that the application range of our system is not limited to sounds; our system could be offer a solution to other signal processing tasks in the future, like seismographic or electrocradiogram data.

The main contributions of this work are:

- The introduction of a new end-to-end architecture that learns a optimized set of filters to extract a spectrogram-like representation of sounds, similar to a time-frequency representation but that is specifically tailored to the problem of acoustic detection.
- Previous methodologies used features taken from other problem domains such as speech recognition or so. And it was a long-standing question whether or not these features were the 'best' suited for environmental sound detection.
- We have beaten the state-of-the-art and specifically broken the barrier of the 90% accuracy. Well above it, with 95% five-fold accuracy thanks to the transfer learning approach we have implemented. Other similar transferences have been tried in the past ([42]) but ours is the one with a much higher performance, (see section 5.4).
- A thorough analysis of the tradeoffs between parameter count and accuracy performance was carried out so that a system designer can choose the desire scale of the model given the system contraints and task requirements. In this we followed the well known methodology of mobilenets, a form of convnets, but other possibilities are avaialable, see next section that would potentially give us better parameter efficiency, that is better performance with the same parameter count.

6.2 Further Work

In this section we delve into how further work can be carried out regarding the use of automatically evolved architectures and the use of a more recent scaling procedure [72]. Both approaches are novel and based on computational principled, instead of a hand-crafting of architectures or the grid-search that we used in this work.

Firstly, it is important to improve on the scaling methodology used and explained in Sect. 4.3.2. Convolutional Neural Networks are commonly developed at a fixed resource budget, and then scaled up for better accuracy when it is the case that more resources are available. There are multiple ways to scale a convolutional neural network. The method we used in this work was primarily based in the scaling of MobileNets [32]. Other approaches scale up the convnet by increasing the number of layers, [29], from ResNet-50 to ResNet-152. Others instead of going deeper go wider as in the case of [81]. Less common and/or popular is to scale up the resolution in order to enhance performance, see for example [34]. Tan et al. [72] proposed a balanced approach to all these three dimensions (resolution, width and depth) in order to obtain

better scaling curves, that is, better accuracy with less resources or, said differently, a graceful degradation of the performance. They called their method compound scaling and leads to better results if a ratio is mantained between the three parameters.Conversely scaling down is a way of trading accuracy and efficiency. Modern convolutional networks are overpamaterized and given the importance of applications in the mobile context as mentioned in Section 1.2 it is important to find reduced-computation models that are performant.

Secondly, there is a discussion of why convolutions are not entirely appropriate for sound processing. Sound events are transparent, they can be combined in a polyphonic way. Also there is no translation invariance in the frequency axis, there is only one in the time axis. Thus, 1-d convolutions are the only reasonable inductive bias, the other is actually invalid in terms of perceptual quality and hence damage performance. More specifically, the feature map obtained after the low-level feature extraction is not an spectrogram in the strict sense of the word. A spectrogram, provides a time-frequency representation that is ordered in time and frequency. Adjacent frames represent consecutive steps in time while frequencies increase in a fashion that is monotonic, f_i is less than or equal to f_{i+1}, for all values of the index i. The dimension defined by i is called feature axis.

A learnable low-level feature extractor that performs filtering by means of convolution preserves ordering along the temporal axis, as is the case in other handcrafted filterbanks. Nevertheless, the ordering along this axis is unconstrained when training a learnable convolutional filters. This present problems when downstream layers or operations require an input that is ordered along the feature axis. Chiefly among these kind of operations or layers are two-dimensional convolutions, which compute a feature map representation based on local time-frequency receptive fields in the signal. Other operations are of the data augmentation class, chiefly among these is the well-known SpecAugment [52]. A detailed comparison and evaluation need to be done to evaluate the impact of the lack of ordering or the impact of enforcing a ordering in the learned filterbank. We can make the assumption that filters that are ordered at the initialization tend to keep the same ordering throughout training. But the question remains if enforcing ordered filters has an effect on performance. It would be interesting as a followup of the current work to test this and similar hypotheses.

Finally, the way we came up with the network was explained in a previous section (Sect. 4.4) and it required a tedious grid search, that is computational expensive, time consuming and rather arbitrary. However, there is a principled computational procedure that can be carried out through the technique known as Neural Architecture Search or NAS. With NAS multiple architectures are modified and selected via a fitness function, usually a proxy of classification accuracy. Therefore, to go even further, neural architecture search [71], [11], [84] can replace the design of architecture to obtain a family of models that can achieve higher accuracy at a reduced computational cost than previous proposed architecture families, like envnet or mobilenets [32] for computer vision. There is some evidence [72], [40] that searched architecture families improve over manually designed ones in terms of transference. That is, these automatically designed networks transfer well from one task to another similar or between datasets of similar characteristics.

In particular, the study of Kornblith et al. [40] found that ImageNet architectures generalize well across different computer vision datasets, with small improvements in ImageNet accuracy spreading to improvements to other tasks and datasets. Nevertheless freezing the weights of pretrained networks did reduce accuracy of the features learned during pretraining, similar to the results presented in Sect. 5.4. In general, ours and the results of others including Kornblith et al. [40] show that transfer is not as good as is when fine-tuning the feature extraction module.

Neural architecture search on AudioSet will allow to increase AudioSet performance just like Imagenet can be increase through this procedure in [72] for example. Hopefully this way, sound recognition community progress as fast as other developed areas of machine learning like computer vision, in terms of efficiency and performance.

REFERENCES

[1] Enrique Alexandre, Lucas Cuadra, Manuel Rosa, and Francisco Lopez-Ferreras. Feature selection for sound classification in hearing aids through restricted search driven by genetic algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(8):2249–2256, 2007.

[2] Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

[3] Meysam Asgari, Izhak Shafran, and Alireza Bayestehtashk. Inferring social contexts from audio recordings using deep neural networks. In *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2014.

[4] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. In *Advances in neural information processing systems*, pages 892–900, 2016.

[5] Lamberto Ballan, Alessio Bazzica, Marco Bertini, Alberto Del Bimbo, and Giuseppe Serra. Deep networks for audio event classification in soccer videos. In *2009 IEEE International Conference on Multimedia and Expo*, pages 474–477. IEEE, 2009.

[6] Daniele Barchiesi, Dimitrios Giannoulis, Dan Stowell, and Mark D Plumbley. Acoustic scene classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, 32(3):16–34, 2015.

[7] Juan Pablo Bello, Charlie Mydlarz, and Justin Salamon. Sound analysis in smart cities. In *Computational Analysis of Sound Scenes and Events*, pages 373–397. Springer, 2018.

[8] Victor Bisot, Romain Serizel, Slim Essid, and Gaël Richard. Feature learning with matrix factorization applied to acoustic scene classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1216–1229, 2017.

[9] David Sanchez Blancas and Jordi Janer. Sound retrieval from voice imitation queries in collaborative databases. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.

[10] Michael Büchler, Silvia Allegro, Stefan Launer, and Norbert Dillier. Sound classification in hearing aids inspired by auditory scene analysis. *EURASIP Journal on Advances in Signal Processing*, 2005(18):1–12, 2005.

[11] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[12] Sachin Chachada and C-C Jay Kuo. Environmental sound recognition: A survey. *APSIPA Transactions on Signal and Information Processing*, 3, 2014.

[13] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In *Advances in neural information processing systems*, pages 416–422, 2001.

[14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[15] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[16] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

[17] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. A committee of neural networks for traffic sign classification. In *The 2011 international joint conference on neural networks*, pages 1918–1921. IEEE, 2011.

[18] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. Very deep convolutional neural networks for raw waveforms. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 421–425. IEEE, 2017.

[19] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.

[20] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.

[21] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR, 2014.

[22] Gustav Theodor Fechner, Davis H Howes, and Edwin Garrigues Boring. *Elements of psychophysics*, volume 1. Holt, Rinehart and Winston New York, 1966.

[23] Kunihiko Fukushima and Sei Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469, 1982.

[24] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[25] Brian R Glasberg and Brian CJ Moore. Derivation of auditory filter shapes from notched-noise data. *Hearing research*, 47(1-2):103–138, 1990.

[26] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[27] Donald D Greenwood. The mel scale's disqualifying bias and a consistency of pitch-difference equisections in 1956 with equal cochlear distances and equal frequency ratios. *Hearing research*, 103(1-2):199–224, 1997.

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.

[31] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson. Speech acoustic modeling from raw multichannel waveforms. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4624–4628. IEEE, 2015.

[32] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[33] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[34] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, pages 103–112, 2019.

[35] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[36] Navdeep Jaitly and Geoffrey Hinton. Learning a better representation of speech soundwaves using restricted boltzmann machines. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5887. IEEE, 2011.

[37] Aren Jansen, Manoj Plakal, Ratheet Pandya, Daniel PW Ellis, Shawn Hershey, Jiayang Liu, R Channing Moore, and Rif A Saurous. Unsupervised learning of semantic audio representations. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 126–130. IEEE, 2018.

[38] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *CoRR*, abs/1912.10211, 2019.

[39] Zvi Kons, Orith Toledo-Ronen, and M Carmel. Audio event classification using deep neural networks. In *Interspeech*, pages 1482–1486, 2013.

[40] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[42] Anurag Kumar, Maksim Khadkevich, and Christian Fügen. Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 326–330. IEEE, 2018.

[43] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[44] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[45] Ze-Nian Li, Mark S Drew, and Jiangchuan Liu. *Fundamentals of multimedia*. Springer, 2004.

[46] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[47] Kuba Łopatka, Paweł Zwan, and Andrzej Czyżewski. Dangerous sound event recognition using support vector machine classifiers. In *Advances in Multimedia and Network Information System Technologies*, pages 49–57. Springer, 2010.

[48] James G Lyons and Kuldip K Paliwal. Effect of compressing the dynamic range of the power spectrum in modulation filtering based speech enhancement. In *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[49] Nelson Mogran, Hervé Bourlard, and Hynek Hermansky. Automatic speech recognition: An auditory perspective. In *Speech processing in the auditory system*, pages 309–338. Springer, 2004.

[50] Douglas O'Shaughnessy. Speech communication, human and machine addison wesley. *Reading MA*, 1987.

[51] Dimitri Palaz, Ronan Collobert, and Mathew Magimai Doss. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. *arXiv preprint arXiv:1304.1018*, 2013.

[52] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[53] Karol J Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2015.

[54] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018. ACM, 2015.

[55] Mirco Ravanelli, Benjamin Elizalde, Karl Ni, and Gerald Friedland. Audio concept classification with hierarchical deep neural networks. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pages 606–610. IEEE, 2014.

[56] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[57] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[58] Hardik B Sailor, Dharmesh M Agrawal, and Hemant A Patil. Unsupervised filterbank learning using convolutional restricted boltzmann machine for environmental sound classification. In *INTERSPEECH*, pages 3107–3111, 2017.

[59] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. Learning the speech front-end with raw waveform cldnns. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[60] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[61] Ralf Schlüter, Ilja Bezrukov, Hermann Wagner, and Hermann Ney. Gammatone features and feature combination for large vocabulary speech recognition. In *ICASSP (4)*, pages 649–652. Citeseer, 2007.

[62] Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun. Very deep multilingual convolutional neural networks for lvcsr. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4955–4959. IEEE, 2016.

[63] Romain Serizel, Victor Bisot, Slim Essid, and Gaël Richard. Acoustic features for environmental sound analysis. In *Computational analysis of sound scenes and events*, pages 71–101. Springer, 2018.

[64] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3288–3291. IEEE, 2012.

[65] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[66] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[67] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[68] Stanley S Stevens and John Volkmann. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3):329–353, 1940.

[69] Dan Stowell, Michael D Wood, Hanna Pamuła, Yannis Stylianou, and Hervé Glotin. Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge. *Methods in Ecology and Evolution*, 10(3):368–380, 2019.

[70] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[71] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[72] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[73] Yuji Tokozume and Tatsuya Harada. Learning environmental sounds with end-to-end convolutional neural network. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2721–2725. IEEE, 2017.

[74] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from between-class examples for deep sound recognition. *arXiv preprint arXiv:1711.10282*, 2017.

[75] Zoltán Tüske, Pavel Golik, Ralf Schlüter, and Hermann Ney. Acoustic modeling with deep neural networks using raw time signal for lvcsr. In *Fifteenth annual conference of the international speech communication association*, 2014.

[76] Srinivasan Umesh, Leon Cohen, and D Nelson. Fitting the mel scale. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 1, pages 217–220. IEEE, 1999.

[77] Michel Vacher, Dan Istrate, Laurent Besacier, Jean-François Serignat, and Eric Castelli. Sound detection and classification for medical telesurvey. In *2nd Conference on Biomedical Engineering*, pages 395–398, 2004.

[78] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

[79] N Vapnik Vladimir and V Vapnik. Statistical learning theory. *Xu JH and Zhang XG. translation. Beijing: Publishing House of Electronics Industry, 2O04*, 1998.

[80] Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

[81] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[82] Matthew D Zeiler, M Ranzato, Rajat Monga, Min Mao, Kun Yang, Quoc Viet Le, Patrick Nguyen, Alan Senior, Vincent Vanhoucke, Jeffrey Dean, et al. On rectified linear units for speech processing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3517–3521. IEEE, 2013.

[83] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv e-prints*, pages arXiv–1710, 2017.

[84] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.