

# Chapter 2

## The Machine Learning Approach for Analysis of Sound Scenes and Events

Toni Heittola, Emre Çakır, and Tuomas Virtanen

**Abstract** This chapter explains the basic concepts in computational methods used for analysis of sound scenes and events. Even though the analysis tasks in many applications seem different, the underlying computational methods are typically based on the same principles. We explain the commonalities between analysis tasks such as sound event detection, sound scene classification, or audio tagging. We focus on the machine learning approach, where the sound categories (i.e., classes) to be analyzed are defined in advance. We explain the typical components of an analysis system, including signal pre-processing, feature extraction, and pattern classification. We also present an example system based on multi-label deep neural networks, which has been found to be applicable in many analysis tasks discussed in this book. Finally, we explain the whole processing chain that involves developing computational audio analysis systems.

**Keywords** Audio analysis system • Sound classification • Sound event detection • Audio tagging • Machine learning • Supervised learning • Neural networks • Single-label classification • Multi-label classification • Acoustic feature extraction • System development process

### 2.1 Introduction

In each application related to computational sound scene and event analysis, the systems doing the computation need to solve very different types of tasks, for example, automatically detecting a baby crying, labeling videos with some predefined tags, or detecting whether a mobile phone is indoors or outdoors.

---

T. Heittola (✉) • E. Çakır

Tampere University of Technology, P.O. Box 527, FI-33101 Tampere, Finland  
e-mail: [toni.heittola@tut.fi](mailto:toni.heittola@tut.fi); [emre.cakir@tut.fi](mailto:emre.cakir@tut.fi); [tuomas.virtanen@tut.fi](mailto:tuomas.virtanen@tut.fi)

T. Virtanen

Laboratory of Signal Processing, Tampere University of Technology, Tampere, Finland

Even though the tasks appear to be very different, the computational methods used actually share several similarities, and follow the same kind of processing architecture.

Sounds present in natural environments have substantial diversity, and, for example, semantically similar sound events have generally different acoustic characteristics. Because of this, the manual development of computational indicators of sound scene or event presence is viable only in very simple cases, e.g., a gunshot might be possible to detect simply based on loudness of the sound event. However, in many practical computational analysis systems the target sounds have more diverse characteristics and the system is required to detect more than one type of sounds. Depending on the target application, the number of classes may vary quite much between different analysis systems. In the simplest case, a detection system uses only two classes of sounds: the target sound class vs. all the other sounds. Theoretically there is no upper limit for the number of classes, but in practice it is limited by the available data that is used to develop systems, the accuracy that can be reached, and computational and memory requirements. In the scenario where there are multiple target classes, systems can also be categorized depending on whether they are able to detect only one event at a time, or multiple temporally overlapping events (which are often present in natural environments). Analyzing a large variety of sounds requires calculating a larger number of parameters from sound signals, and using automatic methods like *machine learning* [3, 9, 13, 22] to differentiate between various types of sounds.

Most of the computational analysis systems dealing with realistic sounds are based on the *supervised machine learning* approach, where the system is trained using labeled examples of sounds from each of target sound type [3, p. 3]. The supervised learning approach requires that there is a set of possible scene (e.g., street, home, office) or event (e.g., car passing by, footsteps, dog barking) categories, *classes*, defined by the system developer, and that there is sufficient amount of labeled examples available to train the system. Other machine learning techniques such as *unsupervised learning* [9, p. 17] and *semi-supervised learning* [9, p. 18] are applicable, however, in this book we largely concentrate on the supervised learning approaches, as they are the most frequently studied and used for the analysis of sound scenes and events.

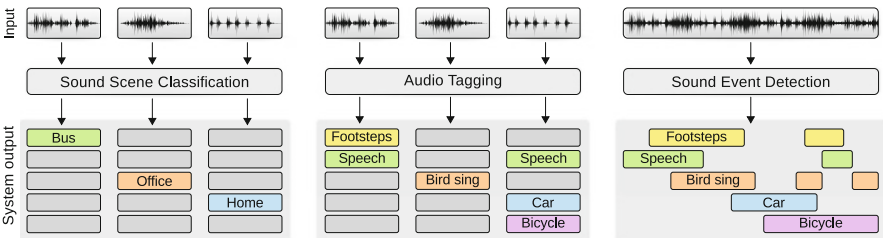
This chapter gives a general overview of the supervised machine learning approach to analysis of sound scenes and events. Section 2.2 starts by presenting the overview of audio analysis systems and introducing the main processing blocks on such systems. Section 2.3 deals with the acquisition of learning examples, and Sect. 2.4 introduces the processing pipeline to transform the audio signals into a compact representations suitable for machine learning. Basics of supervised learning, including acoustic models, generalization properties, and recognition process are discussed in Sect. 2.5, followed in Sect. 2.6 by an example approach based on neural networks. Lastly, Sect. 2.7 presents the development process of the audio analysis systems from problem definition to functional application.

## 2.2 Analysis Systems Overview

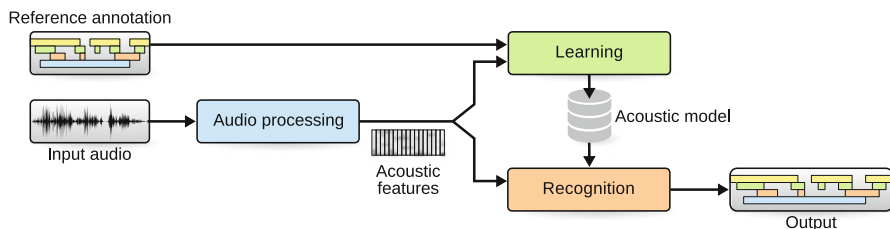
Analysis systems can be categorized into two types depending on whether or not they output temporal information of sounds analyzed. Systems which output information about the temporal activity of target sound classes are said to perform *detection*. In this case, various temporal resolutions can be used, depending on the requirements of the application. Detection can be performed for one or more sound classes at a time. In the case where temporal information is not outputted, a system only indicates whether the sound classes to be analyzed are present in the item subject to analysis (e.g., a video recording, an audio file). A system is said to do *classification* when it can output only one of the possible classes for an item to be analyzed, and it is said to do *tagging*, when it can output more than one class simultaneously for an item to be analyzed. In the machine learning terminology, tagging would be equivalent to multi-label classification. Different analysis systems types are illustrated in Fig. 2.1.

Figure 2.2 presents the block diagram of a typical computational sound scene or event analysis system based on machine learning. The system takes an audio signal as input, either in real-time, captured by a microphone, or offline, from an audio recording. The methods presented in this book assume discrete-time signals, obtained by using analog-to-digital converters. The *audio processing* block consists of different processing stages and outputs *acoustic features*, as the actual analysis of audio is rarely based on the audio signal itself, but rather on the compact signal representation with features. The purpose of the feature extraction is to obtain information sufficient for detecting or classifying the target sounds, making the subsequent modeling stage computationally cheaper and also easier to achieve with limited amount of development material.

At the development stage, the obtained acoustic features are used together with *reference annotations* of the audio training examples, to learn models for the sound classes of interest. Annotations contain information about the presence of target sound classes in the training data, and are used as a reference information to automatically learn a mapping between acoustic features and class labels. The mapping is represented by *acoustic models*. At the usage stage, the learned acoustic



**Fig. 2.1** System input and output characteristics for three analysis systems: sound scene classification, audio tagging, and sound event detection



**Fig. 2.2** Basic structure of an audio analysis system

models are used to do recognition (detection or classification), which predicts labels for the input audio. The recognition stage may also involve temporal models and post-processing of labels.

## 2.3 Data Acquisition

Machine learning approaches rely on data to learn parameters of the acoustic models, and to evaluate the performance of the learned acoustic models. The data includes both audio material and reference metadata associated with it (e.g., class labels). Data acquisition is an important stage of the development, as the performance of the developed system is highly dependent on the data used to develop it. As implementations of machine learning approaches are typically available, obtaining suitable training and development material is often one of the most time-consuming parts of the development cycle.

The defined target application dictates the type of acoustic data, recording conditions it is collected in, and type of metadata required. Essentially, the aim is to collect as realistic as possible acoustic signals in conditions which are as close as possible to the intended target application. Metadata should include a ground truth information which is often manually annotated during the data collection. Collected data should have sufficient amount of representative examples of all sound classes necessary for the target application to enable the acoustic models to *generalize* well [13, p. 107]. For preliminary feasibility studies, smaller datasets, containing only most typical examples can be collected. This type of dataset should not be used for the final system evaluation though, as there is higher danger that the acoustic models learned based on the dataset do not generalize well, and the system is optimized particularly for this small dataset. This section gives a brief overview of factors affecting the selection of the material that should be used, and discusses shortly potential ways to obtain material for development. Available data sources are discussed in more detail in Chap. 6 of this book.

### 2.3.1 Source Audio

The optimal performance of methods based on supervised classification is achieved when the material used to train and develop the system matches with the actual material encountered at the usage stage. Realistic sound sources commonly have internal variations in sound producing mechanism which can be heard as differences in the sound they produce. In classification tasks targeting such sounds, these variations cause intra-class variability which should be taken into account when collecting training material. The amount of examples required to sufficiently capture this variability depends highly on the degree of intra-class variability as well as similarity of target sound classes. As a general rule of thumb, easy classification cases may require tens of sound examples whereas more challenging scenarios can require easily hundreds or thousands of examples to capture intra-class variability sufficiently.

Depending on the application, there can be also variability in the sound signal caused by, e.g., characteristics of acoustic environment (e.g., size of room, type of reflective surfaces), relative placement of the source and the microphone, the used capture device, and interfering noise sources. In the ideal case, the above factors should be matched between actual usage stage and the training material to ensure optimal performance. However, in typical realistic audio analysis scenarios many of these variabilities cannot be fully controlled, leading to some level of mismatch between the material used to train and develop the system and the material encountered in the usage stage, and eventually to poor performance. These variations can be taken into account in the learning stage by making sure that the training material contains a representative set of signals captured under different conditions [20, p. 116]. This technique is called *multi-condition* or mixed condition training.

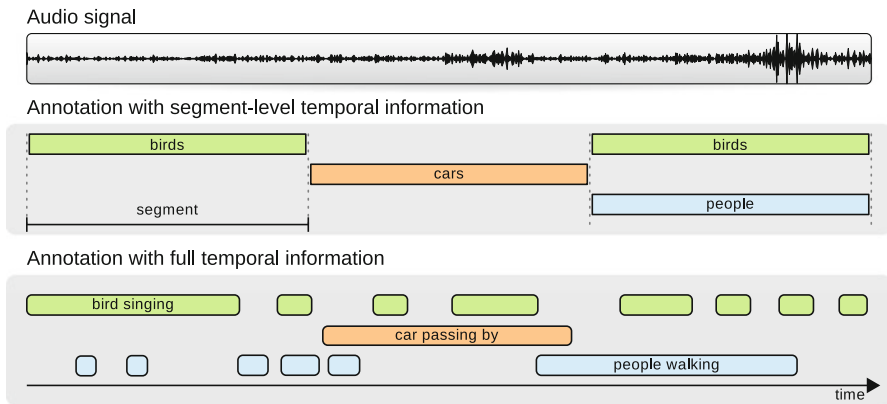
Most of the factors causing the variability (acoustic space, relative placement of the source and microphone, and capturing device) are reflected in the overall acoustic characteristics of the captured sound signal, called *impulse response* or in specific cases room impulse response [20, p. 206]. Different impulse responses can be artificially added to the signals, essentially easing the data collection process when using multi-condition training [41]. An effective strategy to achieve this is to obtain recordings of the target source with as little external effects as possible, and then simulate the effect of various impulse responses by convolving the signal with a collection of measured impulse responses from real acoustic environments. If measured impulse responses are not available, room simulation techniques can be used to generate room impulse responses for different type of acoustic environments [42, p. 191]. Similar strategies can be applied to interfering noise sources. If the noise source is known and stationary at the usage stage, the training material is relatively easy to collect under similar conditions. In the case where there are different types of noise sources at varying positions related to the microphone, the best resort is to use multi-condition training, i.e., include as many expected noise sources in the training material as possible. If it is feasible to obtain recordings of

target sound sources without any interfering noise and recordings with the noise sources alone, the best strategy is to simulate noisy source signals by artificially mixing these two types of recordings with various signal-to-noise ratios (SNR). This typically allows producing larger quantities of relevant training material in comparison to directly recording noisy material. On the other hand, the amount and diversity of the available recordings will influence and perhaps limit the variability of the produced material.

In order to start development quickly, source material can be obtained from external sources such as sound libraries (see Chap. 6 for more information). However, the availability of datasets that are collected for the development of supervised classification methods is limited, and the above discussed factors in the available datasets cannot be controlled properly. Therefore many audio analysis applications require collection of additional material for the development in order to achieve the best performance.

### 2.3.2 Reference Annotations

Supervised learning approaches that are discussed in this book require reference annotations, which indicate in which parts of the source audio each of the source classes is present. Depending on how the annotations are acquired, the annotations can be in different forms. Ideally the annotations will contain temporal information about each class, i.e., when a sound corresponding to the target class starts and when it ends. In practice, accurate temporal information can be difficult to obtain. Often the annotations are segment-level, i.e., each annotation indicates which classes are present in a segment of audio, but there is no temporal information about the class activities [19]. These two annotation types are illustrated in Fig. 2.3.



**Fig. 2.3** Annotation with segment-level temporal information and with full temporal information

Reference annotations can be obtained in various ways. The most generic way to obtain annotations is to do it manually, i.e., by having persons listening to the audio to be used and indicating the activities of each class. This is a very time-consuming process, and annotating a piece of audio accurately takes easily much more time than the length of the audio. On the other hand, human annotation is often the only option to obtain annotations for certain types of sound classes. Human annotation is also the most generic approach since human annotators can be trained to annotate various types of classes. In addition to being slow, human annotation can be inaccurate at least if the material to be annotated is very noisy. Human annotations can also be subjective, which should be taken into account when the annotations are used as a reference when measuring the performance of developed methods. Details on producing annotations and validating their quality can be found in Chap. 6.

Sometimes it is possible to use other sensors to acquire the reference annotations. For example, the activity of a machine can be measured based on the electric power used by the machine or presence of moving cars can be detected from a video signal. This type of extra information may be available only during the development stage, while in the actual usage scenario the system must rely only on audio capture.

When training material is obtained from sample libraries or databases, the database often contains information about its content that can be used to obtain the annotations. However, the descriptions of the database may not match one-to-one the target classes and may therefore require some screening to identify and exclude possible mismatches.

## 2.4 Audio Processing

Audio is prepared and processed for machine learning algorithms in the audio processing phase of the overall system design. This phase consists of two stages: *pre-processing*, in which the audio signal is processed to reduce the effect of noise or to emphasize the target sounds, and *acoustic feature extraction*, in which the audio signal is transformed into a compact representation.

### 2.4.1 Pre-processing

Pre-processing is applied to the audio signal before acoustic feature extraction if needed. The main role of this stage is to enhance certain characteristics of the incoming signal in order to maximize audio analysis performance in the later phases of the analysis system. This is achieved by reducing the effects of noise or by emphasizing the target sounds in the signal.

If the audio data is collected from various sources, it is most likely captured in non-uniform recording settings, with variations in the amount of captured audio channel, and used sampling frequency. These variations can be addressed by converting the audio signal into uniform format by *down-mixing* it into fixed number of channels and *re-sampling* it into fixed sampling frequency.

Knowledge about the recording conditions and characteristics of target sounds can be utilized in the pre-processing stage to enhance the signal. The energy of audio signal is concentrated on lower frequencies; however, for sound recognition higher frequencies contain also important information. This issue can be addressed by *pre-emphasis*—emphasizing high frequencies before feature extraction. In the case of noisy environments, *noise suppression* techniques can be used to reduce interference of environmental noise to the audio analysis [31], while interference of overlapping sounds can be minimized by using *sound source separation* techniques [16].

### 2.4.2 Feature Extraction

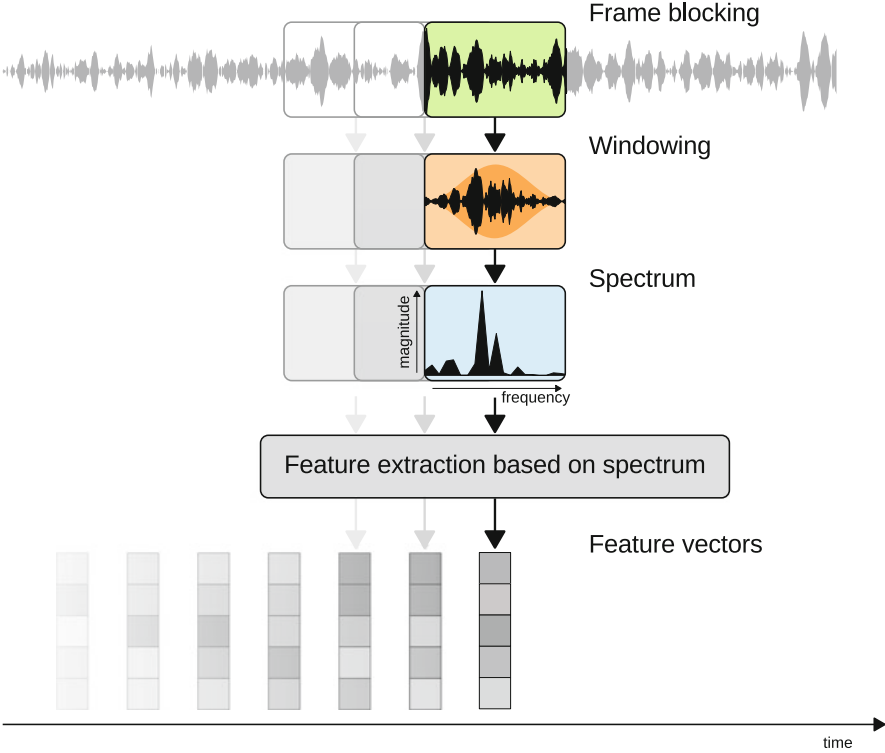
The audio analysis is commonly based on acoustic features extracted from audio signal to represent the audio in a compact and non-redundant way. For recognition algorithms, the necessary property of the acoustic features is low variability among features extracted from examples assigned to the same class, and at the same time high variability allowing distinction between features extracted from example assigned to different classes [12, p. 107]. The feature representations fulfilling this property usually make the learning problem easier. A compact feature representation also requires less amount of memory and computational power than direct use of audio signal in the analysis.

The role of feature extraction is to transform the signal into a representation which maximizes the sound recognition performance of the analysis system. The acoustic features provide a numerical representation of the signal content relevant for machine learning, characterizing the signal with values which have connection to its physical properties, for example, signal energy, its distribution in frequency, and change over time. The processing pipeline in feature extraction is similar for many types of acoustic features used in analysis and consists of *frame blocking*, *windowing*, *spectrum* calculation, and subsequent analysis, as illustrated in Fig. 2.4.

Digital audio signals are discretized in terms of both amplitude and time when captured. For audio analysis, a significant amount of information is contained in relative distribution of energy in frequency, suggesting use of frequency domain features or time-frequency representations. The most common transformation used for audio signals is the discrete Fourier transform (DFT), which represents the signal with a superposition of sinusoidal base functions, each base being characterized by a magnitude and phase [25]. Examples of other transformation methods used for audio signals are constant-Q transform (CQT) [4] and discrete wavelet transform (DWT) [35].

Audio signals are generally non-stationary as the signal statistics (i.e., magnitudes of the frequency components) change rapidly over time. Because of this, the feature extraction utilizes the short-time processing approach, where the analysis is done periodically in short-time segments referred to as *analysis frames*, to capture the signal in quasi-stationary state. In *frame blocking* the audio signal is sliced into fixed length analysis frames, shifted with a fixed timestep. Typical analysis frame





**Fig. 2.4** The processing pipeline of feature extraction

sizes are between 20 and 60 ms, and the frame shift is typically selected so that the consecutive frames are overlapping at least 50%. The analysis frames are smoothed with a *windowing function* to avoid abrupt changes at the frame boundaries that can cause distortions in the spectrum. The windowed frame is then transformed into spectrum for further feature extraction.

The most common acoustic features used to represent spectral content of audio signals are mel-band energies and mel-frequency cepstral coefficients (MFCCs) [7]. Their design is based on the observation that human auditory perception focuses only on magnitudes of frequency components. The perception of these magnitudes is highly non-linear, and, in addition, perception of frequencies is also non-linear. Following perception, these acoustic feature extraction techniques use non-linear representation for magnitudes (power spectra and logarithm) and non-linear frequency scaling (mel-frequency scaling). The non-linear frequency scaling is implemented using filter banks which integrate the spectrum at non-linearly spaced frequency ranges, with narrow band-pass filters at low frequencies and with larger bandwidth at higher frequencies.

Mel-band energies and MFCCs provide a compact and smooth representation of the local spectrum, but neglect temporal changes in the spectrum over time, which

are also required for the recognition of environmental sounds. Temporal information can be included by using delta features, which represent the local slope of the extracted feature values within a predefined time window. Another way to capture the temporal aspect of the features is to stack feature vectors of neighboring frames (e.g., five on each side of the current frame) into a new feature vector.

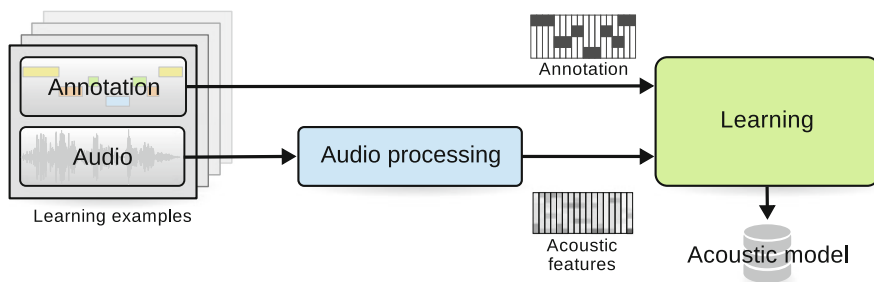
The feature design and parameters used in the extraction commonly rely on prior knowledge and assumptions about the content of the acoustic signal, which in some analysis tasks can lead to sub-optimal performance. For example, selected length of the analysis frame or number of mel-bands might be optimal only for a subset of sound classes involved in the analysis. Unsupervised feature learning can be used to learn better fitted feature representations for specific tasks [29, 39]. The feature learning can be also incorporated into the overall learning process through end-to-end learning, thus avoiding explicit learning of the feature representation. In this approach, the correspondence of the input signal (usually raw audio signal or spectrogram) and desired recognition output is learned directly [8, 14, 17, 40].

## 2.5 Supervised Learning and Recognition

After the data acquisition and feature extraction steps, acoustic features and reference annotations for each audio signal are available. The next step is to *learn* a mapping between these features and class labels for sound classes, where the labels are determined from the reference annotations. This is based on a computational algorithm that can analyze and learn the similarities/differences between acoustic features and the class labels for various sound classes. The learned acoustic model is then used to assign a class label for acoustic features without reference annotations in the usage stage. The study of developing such algorithms is called *supervised learning*.

In supervised learning, we are given a set of input–target output pairs, and the aim is to learn a general model that maps the inputs to target outputs. In the case of classification of sound classes, we have acoustic features  $\mathbf{o}_t$  extracted from  $t = 1, 2, \dots, T$  analysis frames and the reference annotations for each sound signal to be analyzed. Depending on the sound classification task at hand, there are several ways to define the input and the target output for the model (more details in Sect. 2.6). In this chapter, we define the input as  $\mathbf{o}_t \in \mathbb{R}^F$ , acoustic features extracted from a single analysis frame, where  $F$  is the number of features. The target output  $\mathbf{y}_t \in \mathbb{R}^C$  is a binary vector which includes the annotation of present sound classes in the analysis frame among  $C$  predefined class labels. If, according to the reference annotations, the class with the  $c$ th label is present in the analysis frame  $t$ , then  $y_{c,t}$  is set to 1 and 0 vice versa. Therefore, the acoustic features  $\mathbf{o}_t$  and the target outputs  $\mathbf{y}_t$  for each analysis frame correspond to a single input–target output pair, and each pair is called a *training example* for the model.

As illustrated in Fig. 2.5, the acoustic model is trained to learn the relationship between  $\mathbf{o}_t$ , the input feature vectors, and  $\mathbf{y}_t$ , the target outputs obtained from



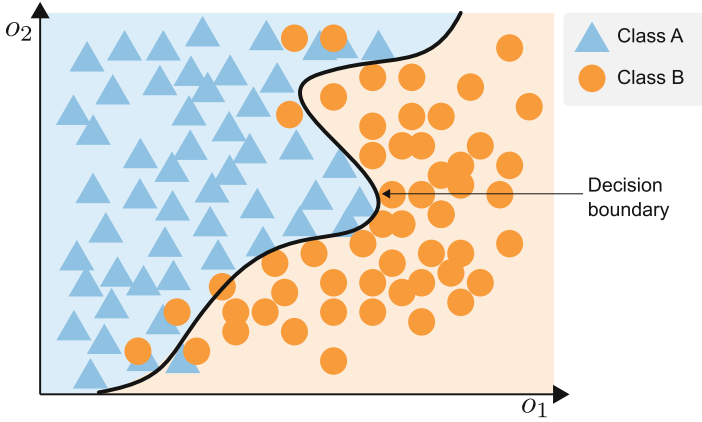
**Fig. 2.5** Overview of supervised learning process for audio analysis

reference annotations. When the target output is chosen in the range  $[0, 1]$ , the model's estimated output  $\hat{y}_{c,t} \in [0, 1]$  is expected to be either (a): close to 0 when class with  $c$ th label is *not* present, or (b): close to 1 when class with  $c$ th label is present. Therefore  $\hat{y}_t$  can be regarded as the *class presence probabilities*.

The type of the classification problem is an important factor in designing the model. If there can be at most one label present at a given frame, the task is regarded as *single-label* classification. Scene classification and sound event classification tasks are most often single label. The task of classifying multiple labels simultaneously present in a given frame is called *multi-label* classification. Sound event detection in real-life environments may belong to this category, since multiple sound events can occur simultaneously in daily life.

### 2.5.1 Learning

The learning process is about searching for the optimal model that would separate the examples from different classes on a given feature space. In Fig. 2.6, we illustrate a simple learning task involving examples with two features  $\{o_1, o_2\}$  from two different classes marked with blue triangles and orange circles. The curved line that divides the examples from different classes is called the *decision boundary*. It is composed of data points that the model estimates to be equally likely belong to one of the two classes. In the given figure, it can be observed that some of the examples end up in the wrong side of the decision boundary, so our model can be deemed imperfect. The performance of the model is calculated through a *loss* (can be also called error or cost) function that calculates the difference between the target and estimated outputs for the training examples, and the model is updated in order to decrease the loss through various optimization techniques. For instance, we can initialize our model parameters so that the decision boundary is a flat line roughly dividing the examples from two classes. Then, we can iteratively update the model parameters by minimizing the mean squared error between the target outputs and estimated outputs based on the decision boundary.



**Fig. 2.6** Examples from two different classes and the decision boundary estimated by the learned model

Supervised learning methods are often grouped into two main categories: *generative* and *discriminative* learning methods. In generative learning, the aim is to model the joint distribution  $p(x, y)$  for each class separately and then use Bayes' rule to find maximum posterior  $p(y|x)$ , i.e., from which class a given input  $x$  is most likely to be generated. Some of the established generative classifiers include Gaussian mixture models (GMM), hidden Markov models (HMM), and naive Bayes classifiers. On the other hand, in discriminative learning, the aim is to model the boundaries between the classes rather than the classes themselves and find a direct mapping between the input and the target output pairs [23]. Neural networks, decision trees, and support vector machines are some of the established discriminative learning methods. When it comes to classification of sound classes, discriminative learning has recently been the widely chosen method [5, 11, 26, 28]. This is due to the fact that high intra-class variability among the samples makes it hard to model the individual class accurately and also there is only little benefit for classification in doing so. For example, if our task is to classify an audio recording as either a cat meow or a dog bark, there is no need to model the cat meow and dog bark sounds individually, as long as we can distinguish these two classes from each other.

## 2.5.2 Generalization

Supervised learning methods aim to learn a model that can map the inputs to their target outputs for a given set of training examples. For the usage stage, the learned model is used to estimate the outputs for a different set of examples, which have not been used during learning stage. These examples are often called *test* (or *unseen*)

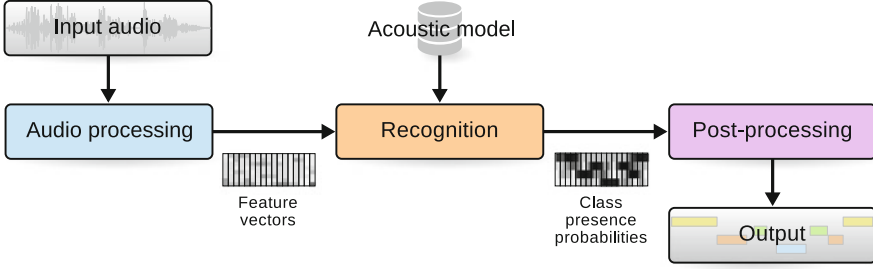
examples. The underlying assumption in the usage stage is that the test examples from a sound class have similar inputs compared to the inputs for the training examples for the same class. Therefore, if the mapping between the input acoustic features and the class label has been learned by the model during the learning stage, then the learned model should be able to estimate the correct outputs for test examples. However, in practice, the performance of the learned model may differ between the training examples and the test examples. In machine learning, the ability to perform well on unseen examples is called *generalization* [13, p. 107].

For sound classification tasks, there are several factors that make it challenging to reach a good degree of generalization. Due to high levels of environmental noise and multi-source nature of the recordings, there can be a large amount of variance among the examples from the same class, i.e., intra-class variability. Besides, class labels are often broadly defined to include a wide range of sound sources with high variation in their acoustic characteristics, such as *door bell* or *bird singing* (see Chap. 7 on taxonomy of sound events).

Modern supervised learning techniques, such as deep learning, are known for their ability to express highly non-linear relationships between the input and the output, given the high depth and large number of parameters [13, p. 163]. However, high expressional capability may also cause *overfitting* [15]. Overfitting is the term used when the difference between loss for training and test examples is large, i.e., the trained model can effectively model the examples for the training set but fails to generalize for the examples in the test set. A learned model with high accuracy in training examples and low accuracy in test examples may indicate that the model has learned the peculiarities of the training examples for better performance on the training set rather than to learn the general acoustic characteristics of the sound classes. Therefore, a sufficiently large number of examples that can provide the general characteristics of the classes and reflect the intra-class variability are required in the training set. Overfitting can also be reduced by using simpler approximation functions and regularization techniques such as L1/L2 weight regularization [24], dropout [32], and batch normalization [18]. On the other hand, the model should be complex enough to provide a good representation of the classes and low loss on the training set to avoid *underfitting* [37]. To summarize, learning is about finding the optimal model on the fine line between overfitting and underfitting.

### 2.5.3 Recognition

After an acoustic model for classification is obtained through the learning stage, the model is ready to be used in an actual usage scenario. An overview of the recognition process is shown in Fig. 2.7. First, acoustic features  $\mathbf{o}_t$  from the test examples are extracted. Then, frame-level class presence probabilities  $\hat{\mathbf{y}}_t$  for the acoustic features are obtained through the learned model. Frame-level class presence probabilities  $\hat{\mathbf{y}}_t$  can be obtained both from acoustic features  $\mathbf{o}_t$  in each timestep, or one can use a memory-based model such as recurrent neural networks to calculate  $\hat{\mathbf{y}}_t$  from



**Fig. 2.7** Basic structure of recognition process

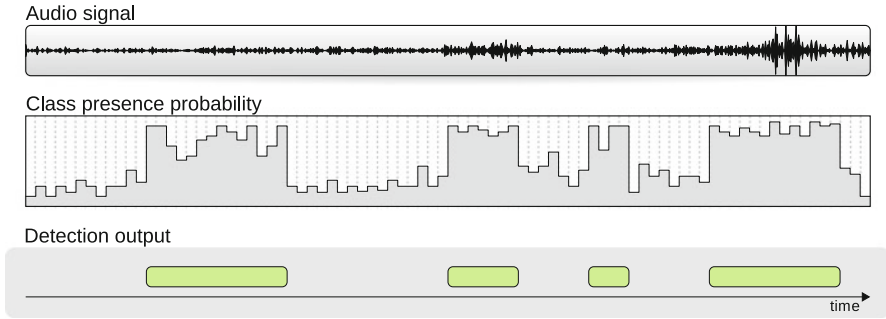
$\mathbf{o}_{(t-T_M):t}$ , where  $T_M$  represents the memory capacity in timesteps. After obtaining  $\hat{\mathbf{y}}_t$ , there are several ways to obtain item-level *binary estimates*  $\mathbf{z} \in \mathbb{R}^C$ , which is a binary vector with only the assigned label elements as 1 and the rest 0, depending on the type of the classification task and the item to be analyzed.

**Classification** For a single-label sound classification task, the item to be analyzed (e.g., audio file or video recording) consists of multiple short analysis frames. In order to combine the class presence probabilities of multiple analysis frames in a single classification output, one can assign each frame the label with highest class presence probability. This way, one would obtain 1-hot *frame-level binary estimates*  $\mathbf{z}_t \in \mathbb{R}^C$ . Item-level binary estimates  $\mathbf{z}$  can be obtained, e.g., by performing a majority voting over the frame-level binary estimates of all the frames for the item, i.e., the item would be assigned the label with the highest number of occurrences among the estimated labels. Another way to obtain  $\mathbf{z}$  would be to sum  $\hat{\mathbf{y}}_t \in \mathbb{R}^C$  class-wise among the frames of the item, and then assign the item the label with highest combined probability.

For a multi-label sound classification task, such as tagging, the number of present sound classes in each item is most often unknown, so a similar majority voting approach cannot be applied. In that case, frame-level class presence probabilities  $\hat{\mathbf{y}}_t$  can be converted to item-level class presence probabilities  $\hat{\mathbf{y}}$ , e.g., by taking the average or the maximum  $\hat{\mathbf{y}}_t$  for each class among all the frames of the item. Taking the maximum  $\hat{\mathbf{y}}_t$  among all the frames would help to correctly classify the classes with rare activity (and therefore low average presence probability over the frames). On the other hand, taking the average  $\hat{\mathbf{y}}_t$  would be useful for the cases when a class is mistakenly assigned a high presence probability in a small portion of frames (since the average probability over the frames would be low in this case).

Then, binary estimates  $\mathbf{z}$  for the item can be obtained by converting  $\hat{\mathbf{y}}$  into a binary vector over a certain binarization rule. A simple binarization rule would be thresholding over a constant  $\sigma$  subject to  $0 < \sigma < 1$ .

**Detection and Temporal Post-processing** In order to obtain the temporal activity information of the sound classes in the usage stage, the acoustic features  $\{\mathbf{o}_r\}_{r=1}^T$  are presented to the acoustic model in a time sequential form. The features are extracted



**Fig. 2.8** Temporal activity information for each class can be obtained from the class presence probabilities for consecutive analysis frames

from the consecutive analysis frames of the item to be analyzed and frame-level class presence probabilities are obtained through the learned model. Detection of the temporal activity for a single class from frame-level class presence probabilities is visualized in Fig. 2.8.

The simplest way of obtaining discrete decisions about source activities from frame-level class presence probabilities  $\hat{\mathbf{y}}_t$  is to first convert  $\hat{\mathbf{y}}_t$  to frame-level binary estimates  $\mathbf{z}_t$  over a certain binarization rule, such as the thresholding that was presented above. Having a frame-level binary estimate  $z_{c,t}$  for class  $c$  in consecutive frames allows us to estimate the onset and offset information for this class. This way, the temporal position of each sound class can be *detected* among the audio signal.

When spectral domain acoustic features are used, the typical values selected for the analysis frame length are quite small (often ranging from 20 to 50 ms). On the other hand, the duration of an individual sound event is most often longer than the analysis frame length, and a single event can span several consecutive frames. For a given acoustic model, this may result in a correlation between classification outputs for consecutive analysis frames. In order to make use of this correlation in the detection tasks, temporal post-processing techniques can be applied over either frame-level class presence probabilities  $\hat{\mathbf{y}}_t$  or binary estimates  $\mathbf{z}_t$ . There are several temporal post-processing techniques, and next, we will shortly describe two of them.

Sound signals may have short, intermittent periods which do not reflect the acoustic characteristics of the sound class that they have been labeled with. For instance, due to the overlap between the feature distributions over different sound classes, acoustic features for an analysis frame for a sound class may be very similar to the features from another class. Therefore, processing the audio signal in short, consecutive analysis frames may introduce some noise in the detection outputs. One simple way to filter this noise and smoothen the detection outputs is to use *median filtering*. For each frame, the post-processed frame-level binary estimate  $\tilde{\mathbf{z}}_t$  is obtained by taking the median of the binary estimates in a window of frames [5]. The method is continued by sliding this window by one frame when every new frame is classified through the model.

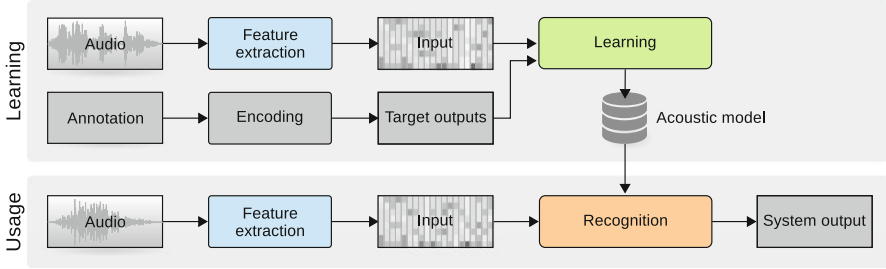
Hidden Markov model (HMM) is an established generative learning method which can be used for temporal post-processing over class presence probabilities [10, 11]. HMM can be used for (a): smoothing  $\hat{\mathbf{y}}_t$  using class presence probabilities from previous analysis frames, and (b): producing an estimate of a hidden state variable for each state which allows segmenting a sequence of features from consecutive frames to various sound classes, provided that HMM states are class-specific. More detailed information about HMMs can be found in Chap. 5.

## 2.6 An Example Approach Based on Neural Networks

This section introduces a basic deep neural network (DNN) [13, 30] based approach for general audio analysis. DNNs are discriminative classifiers that can model the highly non-linear relationships between the inputs and outputs, and which can be easily adapted to output multiple classes at a time (multi-label classification) [5, 26]. This is especially useful for real-life environmental audio, as sounds are very likely to overlap in time. For example, a recording in street environment may include sounds such as car horns, speech, car engines, and footsteps occurring at the same time. DNNs also enable good scalability. Depending on the computational resources available and performance requirements of the target application, the network size can be adjusted accordingly. With larger network sizes, DNNs can take advantage of large sets of examples in the learning process, thus covering a high variability of sounds. This usually leads to better generalization of the acoustic model and better overall performance. With smaller network sizes, the approach can meet the computational limits of the application without compromising the performance too much. DNN-based audio analysis systems have recently shown superior performance over more traditional machine learning approaches (e.g., GMM and support vector machines) given that there is sufficiently large amount of learning examples available [5, 26]. The presented basic system architecture is followed in many current state-of-the-art systems with various extensions, for example [1, 5, 11, 26]. The architecture is here differentiated for two target applications: audio tagging and sound event detection. Even though these applications may seem at first quite dissimilar, the system architectures for them are highly similar, allowing easy switching between applications during the research and development as will be explained. The basic system architecture is illustrated in Fig. 2.9. The system uses DNNs to classify input audio in analysis frames, and using the frame-wise classification results to get a system output matching the requirements of the target application. Collected data, audio signals, and associated reference annotations are divided into non-overlapping training and test sets.

**Learning Stage** In the learning stage, the training set is used to learn the acoustic model. Training examples consist of acoustic features extracted in short analysis frames from audio signals and target outputs defined for each frame based on reference annotations. Acoustic features are extracted in 40 ms analysis frames with





**Fig. 2.9** Framework of the training and testing procedure for the example system. Grey processing blocks are adapted according to the target application for the system

50% overlap over the audio signal. For each analysis frame, the magnitude spectrum is calculated as the magnitude of the discrete Fourier transform (DFT) of the frame. The bins are accumulated into 40 mel bands using non-linear frequency scaling spanning from 0 to 22.05 kHz (assuming audio signal is sampled with 44.1 kHz) and logarithm is applied to get the acoustic features—the *log mel band energies* [33, 34]. To include a temporal aspect of the features into the final feature vector, *frame stacking* can be used: the acoustic features extracted from the current frame are concatenated with features from neighboring frames, e.g., the previous four frames, to create a single feature vector. The target output vectors for each analysis frame are obtained by binary encoding of the reference annotations. In this process, classes annotated to be active within the current analysis frame are marked with 1 and non-active classes are marked with 0. The target outputs for the training set examples will be used in training the acoustic model. A DNN acoustic model is used to learn a mapping between acoustic features and the target outputs for the training set examples [13, p. 163].

The DNN consists of multiple layers of inter-connected elements called *neurons*. Each neuron outputs an activation through a weighted sum of previous layer activations and a non-linear activation function. The first layer takes as input the acoustic features and the following layers take as input the previous layer activations. The class presence probabilities are obtained through the activations of the final layer. The network parameters (weights of the neurons) are learned through an iterative process where parameters are updated using an optimization algorithm (e.g., gradient descent) and a loss function (e.g., cross-entropy) [13, p. 171]. Part of the training examples are left out from the actual learning process, for *validation*, being used to evaluate the performance of the system between the learning iterations and to decide when the learning process should be stopped to avoid overfitting [13, p. 239]. A comprehensive description of DNNs can be found in [13] and a review of the DNN-based techniques can be found in [30]. After the network parameters are trained, the system is ready to be used for test examples.

**Usage Stage** The usage stage shares the acoustic feature extraction part (same parameters) with the learning stage. The same acoustic features are extracted from

the input audio, and the previously learned acoustic model is used to get the class presence probabilities for each analysis frame. The class presence probabilities are acquired by a single forward-pass through the learned network. Frame-wise class presence probabilities are then processed to obtain the output in correct format for the target application as discussed in next subsections.

### 2.6.1 Sound Classification

The previously presented system structure for audio analysis can be adapted for classification applications through specific use of training examples, output layer for the network, and post-processing of the frame-wise class presence probabilities. In the classification task, a segment of audio is classified into a single predefined class for *single-label classification*, or into multiple predefined classes for *multi-label classification*, depending on the target application. Audio analysis systems performing multi-label classification are also referred to as *audio tagging* systems. Illustrative examples of system inputs and outputs for these applications are shown in Fig. 2.1.

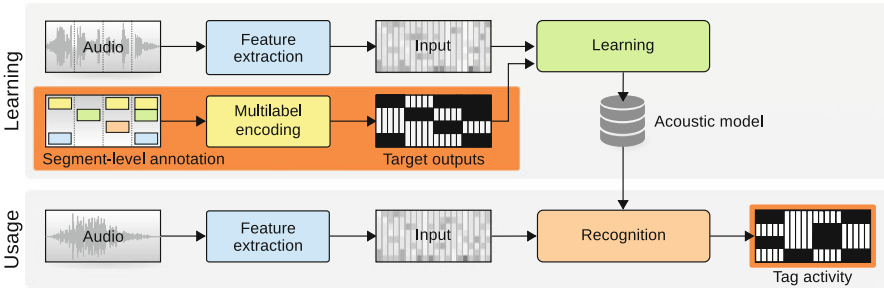
**Single-Label Classification** For single-label classification, the learning examples are audio segments with a single class annotated throughout. The annotations are encoded into target outputs which are used in the learning stage together with audio signals. In this case, classes are mutually exclusive. This condition is included into the neural network architecture by using output layer with *softmax* activation function, which will normalize outputted frame-level class presence probabilities to sum up to one [13, p. 78]. In the usage stage, the frame-level class presence probabilities within the classified item (e.g., audio segment) are first calculated. These probabilities can be used to get the overall classification output in two different ways: by doing classification at frame-level and combining results, or by combining frame-level information and doing final classification at item level. In the frame-level approach, classification is done first for each frame by assigning each frame the label with the highest class presence probability, and then majority voting is used among these frame-level results to get the final classification result. In the item-level approach, the frame-level class presence probabilities are summed up class-wise and the final classification is done by assigning the item the label with highest combined probability. This type of system architecture has been utilized for both acoustic scene classification [2, 27, 36] and sound event classification tasks [21, 28].

**Multi-Label Classification** For multi-label classification or audio tagging, the learning examples contain audio annotated similarly as for single-label classification, only this time multiple classes can be active at same time in the annotations. In this case, the neural network architecture is using an output layer with *sigmoid* activation, which will output class presence probabilities independently in the range (0, 1) [13, p. 65]. In the usage stage, the frame-level class presence

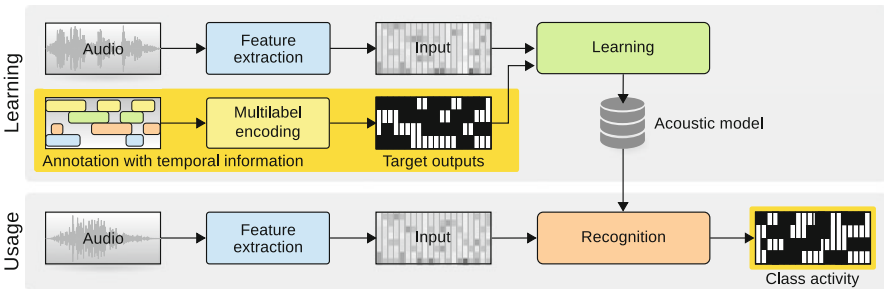
probabilities within the classified item (e.g., audio segment) are calculated and collected over the item. The final class-wise activity estimation is done, for example, based on the average frame-level class presence probability and binarization with a threshold  $\sigma$ . Since the average frame-level class presence probability is in the range  $[0,1]$ , an unbiased selection for  $\sigma$  would be 0.5 [5]. The threshold  $\sigma$  can be adjusted if there is any bias towards less false-alarms (false positives) or less misses (false negatives) in the usage stage. The same overall system architecture has been used in many published works [6, 28, 38]. A system architecture for multi-label sound classification is shown in Fig. 2.10, where highlighted blocks are modified compared to the basic architecture (see Fig. 2.9 for comparison).

### 2.6.2 Sound Event Detection

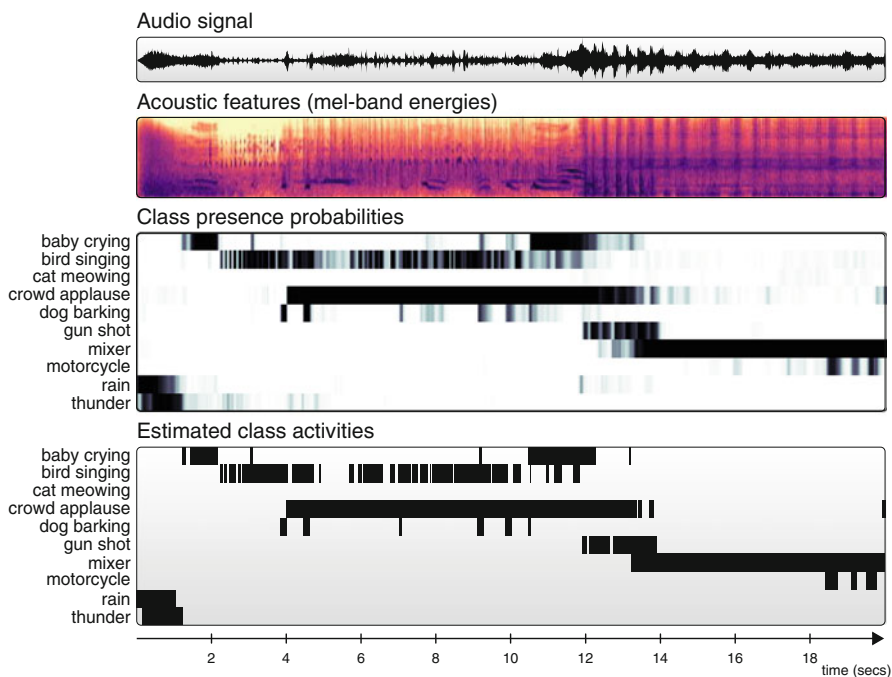
The basic system structure can also be adapted for detection applications. In the detection task, temporal activity is estimated along with actual class labels for the events. A system architecture for sound event detection is shown in Fig. 2.11. The highlighted blocks are the ones different compared to the basic architecture from Fig. 2.9.



**Fig. 2.10** Multi-label sound classification. Adapted blocks compared to the basic system architecture shown in Fig. 2.9 are highlighted



**Fig. 2.11** Sound event detection. Adapted blocks compared to the basic system architecture shown in Fig. 2.9 are highlighted



**Fig. 2.12** Illustration of acoustic features, class presence probabilities, and estimated class activities for a multi-label sound event detection task

Essentially this is the same architecture as the multi-label classification one, the main difference being the temporal resolution of the acoustic modeling during the learning and usage stages. Annotation of sound events provides detailed temporal information about class presence at any given time, which is transformed into a frame-level binary activity indicator that serves as target for the network learning stage. In the usage stage, binarization of the class presence probabilities output by the network is done at frame-level, resulting in estimated class activities that are likely to be noisy. This procedure is illustrated in Fig. 2.12. Post-processing is commonly applied to this binarized output by imposing minimum lengths (e.g., 300 ms) for event instances and the gaps between them (e.g., 200 ms), to clean up activity estimates. This system architecture has been used successfully for sound event detection in recent years [1, 5, 11].

## 2.7 Development Process of Audio Analysis Systems

The previous sections presented the main building blocks of the audio analysis system, and introduced basic system architectures for various target applications. A majority of systems are nowadays based on machine learning methods, and to get

these type of systems to work at the optimal level, rigorous development process is required. The crucial steps required for successful system development are identified and briefly introduced in this section, in order to give a general overview of the development process.

In academia, the audio analysis systems are often developed from a research perspective. The aim of this process is to push the technology in this field further, including through academic publications. Once the technology reaches a sufficient level, measured either in analysis performance or user experience, the development focuses on refining the technology and making it work reliably in real use cases, and possibly building a real product around it. Often in this stage the development starts to shift towards industry either through joint projects between academia and industry, or by moving intellectual properties to industrial partners. As the development progresses, the aim of the development is to deploy a product into actual use (either into commercial or non-commercial market).

The development of a audio analysis system consists of two main phases: *technological research*, where research and development is done in a laboratory environment, and *product demonstration* with a prototype system in a real operating environment. This book concentrates mostly on topics related to the first phase. The technological research phase uses fundamental research from different fields, and applies them to the target application. Interesting supporting research fields for sound scene and event analysis are, for example, human perception and machine learning. The aim of technological research is to produce a *proof-of-concept* system that shows the capability and feasibility of the analysis system, with the system evaluated in laboratory environment with audio datasets. In the product demonstration phase, the proof-of-concept system is further developed into a *prototype* having all the key features planned for the final system. The prototype is tested in realistic use cases, and demonstrated with real users, while continuing to develop and refine the system for eventually being suitable for deployment into use.

### 2.7.1 Technological Research

Before entering the active research and development of the audio analysis system, one has to identify the target application for the system and main characteristics of this application, as these will dictate system design choices later on. By having the target application identified, the research problem becomes easier to define. Sometimes in academic research, one cannot identify a clear target application for the system, especially in such early stage of the research. In these cases it is still a good practice to envision a speculative application for the system to steer the research and development.

In the research problem definition, the analysis system type is identified (detection vs. classification), the used sound classes are defined, and amount of classes needed to be recognized at the same time is defined (e.g., classification vs. tagging). For example, if our target application is the recognition of the sound scene in

5 s intervals from a predefined set of scene classes, then the problem definition would be the classification of input signal into a single class within predefined set of 15 sound scene classes. Another example would have as target application to recognize certain target sound events in office environment (e.g., mouse click); in this case the problem definition would be the detection of occurrences of the one predefined sound event in a predefined environment. Based on the defined research problem, the system development moves into active research and development. In this phase, three main stages can be identified: data collection, system design and implementation, and system evaluation.

**Data Collection** The audio data and associated metadata is used to learn parameters of the acoustic models in the system design and implementation stage. Furthermore, the data is used to evaluate performance in the system evaluation stage. The defined target application dictates the type of acoustic data, recording conditions it is collected in and type of metadata required. In general, the aim should be to collect as realistic as possible acoustic signals in conditions which are as close as possible to the target application. Details of the data collection and the annotation process are discussed in Chap. 6.

**System Design and Implementation** The main goal of the system design and implementation stage is to create a proof-of-concept system which solves the defined problem. The stage starts with the design of the overall system architecture and implementation of the initial version of the system. This initial version usually contains basic approaches and it is used as comparison point (*baseline system*) when developing the system further. The actual system design is a gradual process, where different steps of the system are developed either independently or jointly, and integrated into the overall system. Main steps involved in the processing chain for system design are audio processing (containing, e.g., pre-processing and acoustic feature extraction), machine learning (containing, e.g., learning, and recognition or detection), and post-processing. To some extent, it is possible to isolate each step in the development and optimize the corresponding parameters of the system while keeping the other parts fixed. This allows to take into account the effect of each processing step on the overall system performance, and will enable identification of the error sources and their contribution to the overall performance. The system integration stage uses the knowledge acquired in the development process to maximize performance of the overall system. In particular cases, some steps can be designed, implemented, and evaluated independently outside the system for optimal performance: for example, a noise suppression method can be first evaluated using specific data and metrics, before including it into the developed system as a pre-processing step.

**System Evaluation** The evaluation of the system is based on the test data and reference annotations assigned to it, and using a metric related to the target application. Ideally the system should be evaluated in scenarios which match the target application as much as possible, to get a realistic performance estimation. During the development the evaluation is commonly done in steps by gradually

increasing how closely the scenario matches the target application to better isolate factors affecting the performance. At the beginning of the development, highly controlled laboratory scenarios are preferred, and as the development progresses evaluation switches to more realistic scenarios. An example of a highly controlled scenario is the *offline* scenario where pre-recorded audio data is read directly from audio files and the analysis can be done efficiently for large datasets repeatedly while the core algorithms are developed. Depending on the target application, the system should be evaluated also in *online* use case, where the system is capturing audio in real-time. For example, the same pre-recorded audio data used in offline case can be played back in a laboratory environment to verify that the system performs similarly as in offline case and then move to more realistic usage environment. The evaluation metrics are chosen to reflect performance that should be maximized or errors that should be minimized in the system development. In some cases, subjective evaluation of the system can be performed based on user opinions or user satisfaction with system output, avoiding the need for reference annotations. For objective evaluation, a part of the data is assigned to the *training* of the system and a part is assigned to the *testing* of the system's performance. For sound scene recognition systems the most commonly used metric is accuracy, a percentage of correctly classified items. For sound event detection systems the most commonly used metrics are F-score (balanced mean of precision and recall) and error rate (ER). They are both calculated based on correctly classified items and errors made by the system, but emphasize different characteristics of the systems, namely the ability of correctly detecting as much as possible of the content versus the ability of making as small amount of mistakes as possible. The details of evaluation metrics are discussed in Chap. 6. The performance of the system is compared against other solutions to the same problem, for example, state-of-the-art methods or well-established (*baseline*) approaches. The analysis of these alternative solutions and comparison against developed system is necessary to put the obtained evaluation scores into larger context while doing the research.

### 2.7.2 Product Demonstrations

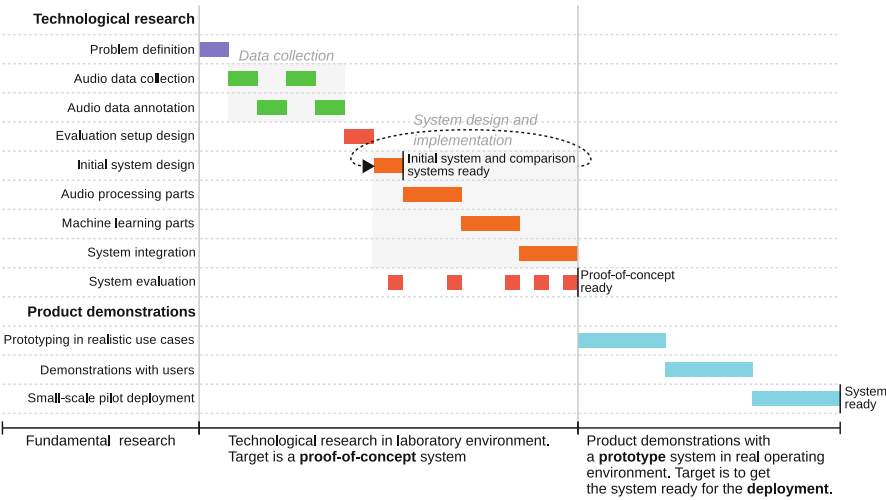
Once the proof-of-concept system is ready, the development moves to the product demonstration phase. In this phase, the desired use cases are first identified, and acceptable error types and level of performance in these use cases are defined. These factors are closely related to the end users' requirements and perception on good performing system, and thus they should be considered as early as possible in the development of the product. The proof-of-concept system developed in the technological research phase is used as a starting point for the development targeting a prototype having all the key features planned for the final system. When the prototype is ready, the technology is validated by testing it in real operating environment with realistic use cases, and demonstrated with real users. User experience studies can be used in this stage to get quantified feedback, and these results can be further used to refine the overall system design.

In the actual development project, there are often setbacks which force the development to return back to the technological research phase and to reiterate stages within it. Testing the system with realistic use cases can reveal problems in the core system design which were not identified earlier in the development. For example, the system could have poor user experience due to a wrongly selected recognition approach which is producing results with too high latency, or the system could have low recognition performance because of low noise robustness. These type of problems are usually such fundamental design flaws that they have to be addressed in the technological research phase.

Once the prototype is validated and achieves sufficient recognition performance with good user experience, the system is developed into a final system which is demonstrated with a small group of real users in actual operating environments. Usually in this stage the core system is considered ready and development concentrates mainly on polishing the possible user interface and communication interfaces with other applications. After the successful demonstrations, the system can be deployed to the market with small scale pilots first, and finally in full scale.

2.7.3 Development Process

The previously introduced development stages often overlap, and are executed multiple times during the overall development process. An example of one possible development process is shown in Fig. 2.13. The figure shows main stages for both



**Fig. 2.13** Process graph for audio analysis system development. Fundamental research is happening outside this graph. The actual development is divided into two phases, technological research phase and product demonstration phase



the technological research and the product demonstration phases to give a comprehensive view for the whole process. However, as the book mainly concentrates on the technological research phase, in the following only the stages from technological research phase are fully explained.

The technological research phase starts from the problem definition stage, where the target application for the system is identified and the system characteristics are defined. After this, the actual active development starts by data collection and manual annotation of this collected data. The annotation stage is usually one of the most time-consuming parts of the development, and in some cases the amount of data is gradually increased as the annotation progresses during the development. It is also a good practice to interlace the data collection and annotation, to get some data ready for the development as early as possible. Once the dataset is complete, the evaluation setup is designed by selecting the evaluation metrics, defining the cross-validation setup, and selecting appropriate comparison systems. Before entering the full system development, the overall system architecture has to be designed. As each part of the system is developed in separate development stages but evaluated as a part of entire system, the components of the whole system have to be defined in general terms. The initial version of the entire system is implemented based on this design by using basic approaches, and usually is also used as a baseline system.

After the initial version of the system is ready, the main development can start. Individual parts of the system are designed, implemented, and evaluated in the order which follows the logical signal path through the system: first audio processing parts (pre-processing, acoustic feature extraction), followed by machine learning parts, and finally the post-processing parts. The evaluation results are used to guide the design choices within each part of the system. When the system performance reaches the desired level or saturates, the development moves to the next part. In the system integration stage, all parts of the system are optimized to get maximum overall performance. The end result of this stage is a complete proof-of-the-concept system which can be moved for further development to the product demonstration phase.

## 2.8 Conclusions

This chapter introduced the basic concepts in computational methods used for audio analysis, concentrating on supervised machine learning approaches. The focus is on classification and detection applications such as scene classification, audio tagging, and sound event detection. Systems for audio analysis have very similar architecture, with building blocks such as data acquisition, feature extraction, and learning often being identical between different systems. The learning process mirrors the selected target application in its association between labels and features, guiding the mapping between class labels and features performed during learning. An approach based on deep neural networks was presented, illustrating a very general system architecture that can be adapted for various sound classification and detection problems.

The last section of this chapter brought into discussion the larger picture of system development, from the definition of the problem all the way to the product developed for the market. System development is often a lengthy and iterative process involving stages of various difficulty and duration. Mid-way of this process is the link between academic research and industry, where the focus switches from proof-of-concept to the commercialization of a product. Industrial research concentrates on the prototyping and product demonstrations with the goal of refining and improving the user experience with the product. Specific solutions for this will be presented in more detail in Chap. 12.

## References

1. Adavanne, S., Parascandolo, G., Pertila, P., Heittola, T., Virtanen, T.: Sound event detection in multichannel audio using spatial and harmonic features. In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pp. 6–10 (2016)
2. Bae, S.H., Choi, I., Kim, N.S.: Acoustic scene classification using parallel combination of LSTM and CNN. In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pp. 11–15 (2016)
3. Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer, New York (2006)
4. Brown, J.C.: Calculation of a constant  $q$  spectral transform. *J. Acoust. Soc. Am.* **89**(1), 425–434 (1991)
5. Çakır, E., Heittola, T., Huttunen, H., Virtanen, T.: Polyphonic sound event detection using multi label deep neural networks. In: *The International Joint Conference on Neural Networks 2015 (IJCNN 2015)* (2015)
6. Çakır, E., Heittola, T., Virtanen, T.: Domestic audio tagging with convolutional neural networks. Technical report, DCASE2016 Challenge (2016)
7. Davis, S.B., Mermelstein, P.: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoust. Speech Signal Process.* **28**, 357–366 (1980)
8. Dieleman, S., Schrauwen, B.: End-to-end learning for music audio. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6964–6968. IEEE (2014). doi:10.1109/ICASSP.2014.6854950
9. Du, K.L., Swamy, M.N.: *Neural Networks and Statistical Learning*. Springer Publishing Company, Incorporated, New York (2013)
10. Espi, M., Fujimoto, M., Kubo, Y., Nakatani, T.: Spectrogram patch based acoustic event detection and classification in speech overlapping conditions. In: *4th Joint Workshop on Hands-free Speech Communication and Microphone Arrays (HSCMA)*, pp. 117–121 (2014)
11. Espi, M., Fujimoto, M., Kinoshita, K., Nakatani, T.: Exploiting spectro-temporal locality in deep learning based acoustic event detection. *EURASIP J. Audio Speech Music Process.* **2015**(1), 26 (2015)
12. Gold, B., Morgan, N., Ellis, D.: *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, 2nd edn. Wiley-Interscience, New York, NY (2011)
13. Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. MIT Press, Cambridge (2016)
14. Graves, A., Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks. In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, ICML'14, vol. 14, pp. 1764–1772. JMLR Workshop and Conference Proceedings (2014)
15. Hawkins, D.M.: The problem of overfitting. *J. Chem. Inf. Comput. Sci.* **44**(1), 1–12 (2004)
16. Heittola, T., Mesaros, A., Virtanen, T., Gabbouj, M.: Supervised model training for overlapping sound events based on unsupervised source separation. In: *38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, pp. 8677–8681 (2013)

17. Hertel, L., Phan, H., Mertins, A.: Comparing time and frequency domain for audio event recognition using deep learning. In: *Proceedings IEEE International Joint Conference on Neural Networks (IJCNN 2016)*, pp. 3407–3411 (2016)
18. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. *CoRR* abs/1502.03167 (2015)
19. Kumar, A., Raj, B.: Audio event detection using weakly labeled data. In: *Proceedings of the 2016 ACM on Multimedia Conference, MM'16*, pp. 1038–1047. ACM, New York (2016)
20. Li, J., Deng, L., Haeb-Umbach, R., Gong, Y.: *Robust Automatic Speech Recognition — A Bridge to Practical Applications*, 1st edn., 306 pp. Elsevier, Amsterdam (2015)
21. Lim, H., Kim, M.J., Kim, H.: Cross-acoustic transfer learning for sound event classification. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2504–2508 (2016)
22. Murphy, K.P.: *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge (2012)
23. Ng, A., Jordan, A.: On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. *Adv. Neural Inf. Proces. Syst.* **14**, 841 (2002)
24. Nowlan, S.J., Hinton, G.E.: Simplifying neural networks by soft weight-sharing. *Neural Comput.* **4**(4), 473–493 (1992)
25. Oppenheim, A.V., Schaffer, R.W., Buck, J.R.: *Discrete-Time Signal Processing*. Prentice Hall, Upper Saddle River, NJ (1999)
26. Parascandolo, G., Huttunen, H., Virtanen, T.: Recurrent neural networks for polyphonic sound event detection in real life recordings. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6440–6444 (2016)
27. Petetin, Y., Laroche, C., Mayo, A.: Deep neural networks for audio scene recognition. In: *23rd European Signal Processing Conference (EUSIPCO)*, pp. 125–129. IEEE, New York (2015)
28. Piczak, K.J.: Environmental sound classification with convolutional neural networks. In: *IEEE International Workshop on Machine Learning for Signal Processing* (2015)
29. Salomon, J., Bello, J.P.: Unsupervised feature learning for urban sound classification. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 171–175 (2015)
30. Schmidhuber, J.: Deep learning in neural networks: an overview. *Neural Netw.* **61**, 85–117 (2015)
31. Schröder, J., Moritz, N., Schädler, M.R., Cauchi, B., Adiloglu, K., Anemüller, J., Doclo, S., Kollmeier, B., Goetze, S.: On the use of spectro-temporal features for the IEEE AASP challenge 'detection and classification of acoustic scenes and events'. In: *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 1–4 (2013)
32. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014)
33. Stevens, S.S., Volkman, J.: The relation of pitch to frequency: a revised scale. *Am. J. Psychol.* **53**, 329–353 (1940)
34. Stevens, S.S., Volkman, J., Newman, E.B.: A scale for the measurement of the psychological magnitude pitch. *J. Acoust. Soc. Am.* **8**(3), 185–190 (1937)
35. Tzanetakis, G., Essl, G., Cook, P.R.: Audio analysis using the discrete wavelet transform. In: *Proceedings of the WSES International Conference Acoustics and Music: Theory and Applications (AMTA 2001)*, Skiathos, pp. 318–323 (2001)
36. Valenti, M., Diment, A., Parascandolo, G., Squartini, S., Virtanen, T.: DCASE 2016 acoustic scene classification using convolutional neural networks. In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pp. 95–99 (2016)
37. Van der Aalst, W.M., Rubin, V., Verbeek, H., van Dongen, B.F., Kindler, E., Günther, C.W.: Process mining: a two-step approach to balance between underfitting and overfitting. *Softw. Syst. Model.* **9**(1), 87–111 (2010)

38. Xu, Y., Huang, Q., Wang, W., Jackson, P.J.B., Plumbley, M.D.: Fully DNN-based multi-label regression for audio tagging. In: *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pp. 105–109 (2016)
39. Xu, Y., Huang, Q., Wang, W., Foster, P., Sigtia, S., Jackson, P.J.B., Plumbley, M.D.: Unsupervised feature learning based on deep models for environmental audio tagging. *IEEE/ACM Trans. Audio Speech Lang. Process.* **25**, 1230–1241 (2017)
40. Yuji Tokozume, T.H.: Learning environmental sounds with end-to-end convolutional neural network. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2712–2725 (2017)
41. Zhao, X., Wang, Y., Wang, D.: Robust speaker identification in noisy and reverberant conditions. *IEEE/ACM Trans. Audio Speech Lang. Process.* **22**(4), 836–845 (2014)
42. Zölzer, U. (ed.): *Digital Audio Signal Processing*, 2nd edn. Wiley, New York (2008)

Computational Analysis of Sound Scenes and Events

Virtanen, T.; Plumbley, M.D.; Ellis, D. (Eds.)

2018, X, 422 p. 81 illus., 54 illus. in color., Hardcover

ISBN: 978-3-319-63449-4