Industrial Machine Monitoring

Real-Time Anomalous Sound Event Detection on Low-Powered Devices

Anton Andersson,
Alexander Magnusson

Master's thesis
2023:E58

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Master's Theses in Mathematical Sciences 2023:E58
ISSN 1404-6342

LUTFMS-3485-2023

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/

Abstract

Traditionally fault detection in industrial machinery has been performed manually by experienced machine operators listening to the machines. However, it is desirable to automate this process to increase efficiency and improve the working environment of the operators. The main challenge in this thesis is to create a system that can accurately detect when an anomalous sound event occurs, at the same time the system may not report too many false alarms. Additionally, the system must perform the detection fast enough for the detected anomalies to still be relevant. This thesis therefore explores lightweight machine learning approaches to anomalous sound event detection such as Gaussian Mixture Models (GMM) and One-Class Support Vector Machines (OCSVM). The experiments evaluate how low-level descriptors from the time-, spectral- and cepstral-domain perform as features modeling the characteristics of a sound segment. Another set of experiments evaluates if it is possible to detect anomalies fast enough to achieve real-time anomaly detection. The results indicate that it is possible to detect anomalies of sufficient magnitude in relation to the expected signal. Furthermore, it is found that it is possible to detect anomalies at a speed fast enough to enable real-time anomaly detection on limited hardware such as a Raspberry Pi 4. Lastly, a real-time anomaly detection application based on the findings is presented together with the results for a few test runs.

Contents

Chapter 1

Introduction

Automation of industrial machinery is an essential problem on the road to the fourth industrial revolution. Automated fault monitoring is a key component in the quest to reach full automation, allowing for optimized service schedules and reduced maintenance downtime. One approach to fault monitoring is through anomaly detection. Anomaly detection is a process that aims to detect patterns in data that do not conform to some definition of normalness. Patterns that deviate from the normal are often of interest, as they may indicate that something is wrong. As an example, in the application of industrial machine monitoring, an anomaly might indicate that a machine is about to malfunction, which could stop production or potentially cause harm both to man and machine. It is therefore desirable to detect anomalies so that the machine can be serviced in a safe manner, reducing the impact on production.

In an industrial setting this type of anomaly detection has traditionally been done by machine operators who - through experience - have learned how a certain machine should or should not sound. It is however desirable to automate this process for many reasons, an automated system could run at all hours, removing the dependency of always having an experienced operator present. Furthermore, a lightweight, cheap and easily maneuvered anomaly detection system scales far better than having individual operators listening for anomalies, which affords the possibility to increase the number of machines covered at the same time. Lastly, offloading this responsibility from the operators may also improve the working environment by reducing their exposure to harmful noise, that could potentially damage their hearing. This thesis will therefore focus on sound anomaly detection in an industrial setting.

On a high level, it is rather trivial to define normal behavior and classify outlying data as anomalies. But in reality, it is much more challenging. One of the big challenges is to find labeled anomaly data. In the context of industrial machine monitoring, anomalies could represent events where a machine is about to malfunction, which can be assumed to be a very rare event, hence very little data of such events exist. Furthermore, the specific machines might be few enough and too expensive to intentionally break for data collection purposes. In contrast, data of normal behavior is often much easier to come by, as it is assumed that a machine operates in normal conditions most of the time.

Since data of normal behavior is often much more readily available than anomalous data, the problem of anomaly detection is often approached by creating a model only using normal data. The problem of determining whether new samples are anomalous or not is treated as a classification problem where samples that fall within some boundary around the normal behavior determined by the model are classified as normal, while samples outside the boundary are classified as anomalies [1].

Another challenge is that of the response time of the detection system. If it takes too long to analyze a segment of sound and flag potential anomalies, it might already be too late to take action when the warning is raised. It is therefore important that such a system can operate in real-time, detecting anomalies as they occur [2].

This report is divided up into two major parts, a pre-study, and an application. The pre-study aims to evaluate whether it is feasible to separate anomalies from normal sound, and if it can be done fast enough to achieve real-time anomaly detection. The pre-study compares two models, one based on Gaussian Mixture Models (GMM) and one based on Support Vector Machines (SVM). The performance of different feature sets is evaluated for both of the models in order to find a set of features that accurately model the detection problem. The second part proposes an application detecting anomalies and providing visual feedback in real-time using a pre-trained model running on a Raspberry Pi.

The general steps of an anomalous sound event detection workflow is presented in Figure 1.1. Let it serve as an overview to refer back to when reading the report.

1.1 Previous work

DCASE2020 Challenge Task 2: Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring presents a related problem. The task is to train an anomaly score calculator on ten second long normal sound segments. The calculator is trained so that a segment containing an anomaly is given a large score and a segment without any anomalies gets a low score. The calculator should also output a large anomaly score when parts of the ten second segment is anomalous, rather than only when the whole segment is considered to be anomalous. The goal in DCASE2020 Challenge Task 2 is thus to determine if there exists an anomaly somewhere in a ten second segment regardless of where in the segment it is located or how large part of the segment is considered to be anomalous [3].

In the past, sound event detection (SED) have often made use of GMMs and Hidden Markov models (HMM), using hand-crafted time and frequency domain features [4]. In order to improve on these models other models such as One-Class Support Vector Machines (OCSVM) and Deep Neural Networks (DNN) have gained popularity [2].

Models based on DNNs are often mentioned in previous work. While these kinds of models offer advantages when working with multivariate and high dimensional data and often boast impressive performance, they also suffer some disadvantages. DNNs require large amounts of data for training. Storing and processing enough data for these models may not be feasible depending on the hardware the models should run on. Anomaly detection is commonly performed in real-time

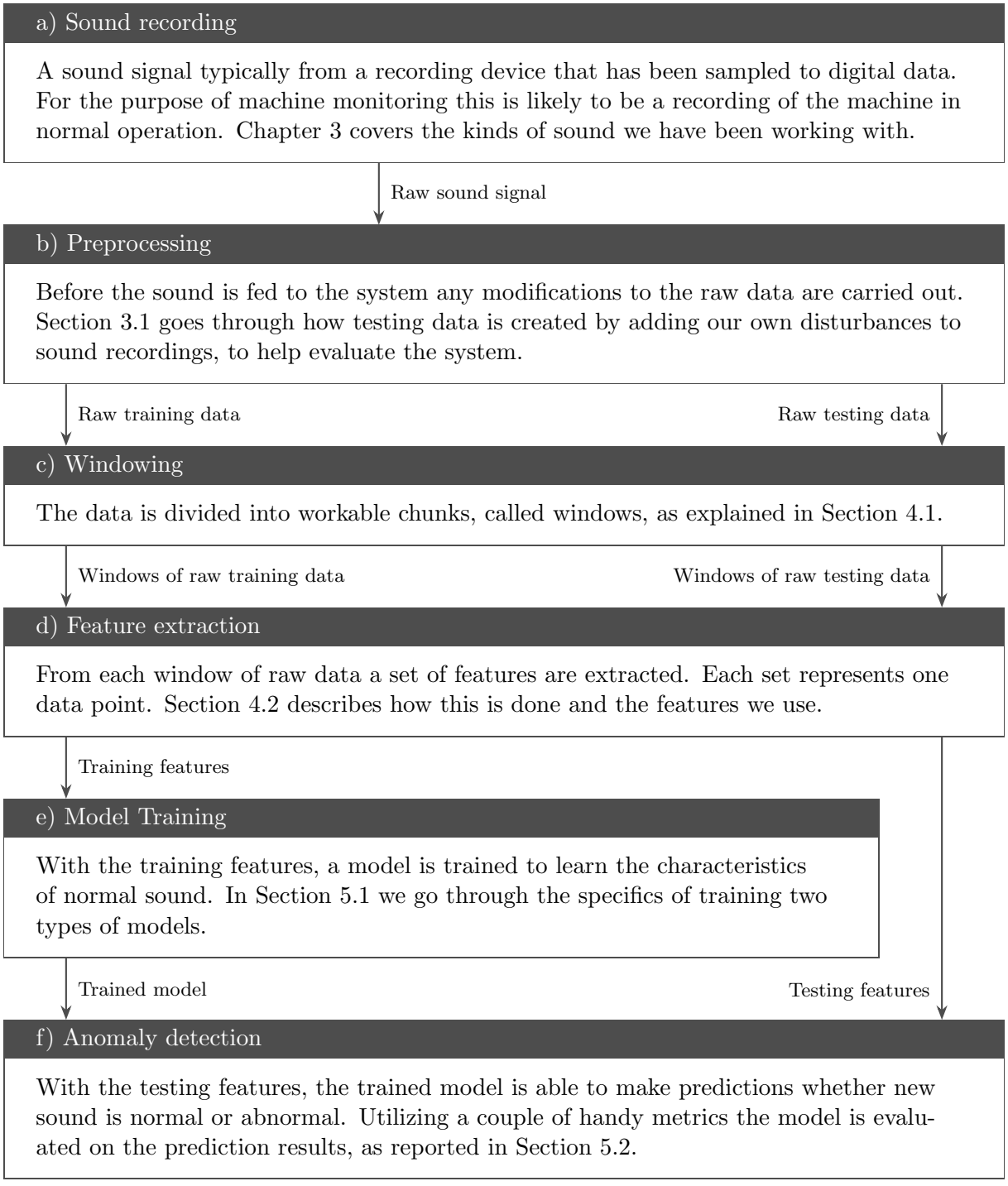


Figure 1.1: An anomalous sound event detection workflow.

which requires low latency. Estimating the parameters for a DNN may be too computationally expensive to be viable [5].

1.2 Problem statement

The goal of this project is to investigate whether it is feasible to detect sound anomalies in real-time, relieving machine operators of this duty. Since the idea is to replace human beings as detectors we have limited the scope of this thesis to only consider sound within the human hearing range. We want to run the application on the device that record the data in order to reduce the complexity and logistics required to operate the application. In addition, keeping collected data on the recorded devices simplifies the handling of data in compliance with privacy laws. The devices that record the audio often have limited hardware capabilities, comparable to a Raspberry Pi model 4B, we therefore consider lightweight models such as GMMs and SVMs.

In order to reach the goals of this project, the following questions were posed:

- Is it possible to train lightweight machine learning models such as GMMs and SVMs to detect anomalies in an environment with limited variability in background noise and types of anomalies?
- Is either a GMM or a SVM more suitable for the detection problem, both in terms of detection performance and speed?
- Can low-level descriptors such as time-domain, spectral-domain and cepstral-domain features accurately model the anomaly detection problem?
- How does characteristics such as duration, frequency and magnitude of an anomaly affect its detectability?
- Is it possible to create a fast enough application to perform anomaly detection in real-time on hardware with limited performance, such as a Raspberry Pi?

Chapter 2

Theory

The following section introduces the theory that the thesis builds upon. Section 2.1 introduces the digital signal processing concepts used to represent, analyze and visualize audio signals. Sections 2.2 and 2.3 introduces the machine learning models used to model the anomaly detection problem. Sections 2.4, 2.5 and 2.6 introduces the different metrics used to evaluate the proposed models performance for the anomaly detection task.

2.1 Spectrograms & Short-Time Fourier Transform (STFT)

The Fourier transform is a useful tool to extract the frequency components of a signal. Normally the Fourier transform analyzes a whole signal at once. However, in the application of detecting anomalies in sound it is of interest to analyze how the frequency representation changes over the duration of the signal.

In order to capture changes within the signal, the signal is divided up into shorter segments, known as windows. A partial example of the windowing process can be seen in Figure 2.1, where a signal is divided into shorter overlapping windows. The Fourier transform is then applied locally, to each of the M windows. Figure 2.2 illustrates a partial example transforming two, consecutive windowed segments from the time domain to the frequency domain via a Fourier transform. This process of windowing a signal and applying the Fourier transform to the individual windows is known as the short-time Fourier transform, or STFT. The result of the STFT can be seen as a time-frequency distribution, where time is represented by the frame number m and the k-th frequency bin, resulting in a N x M matrix. The STFT is defined as

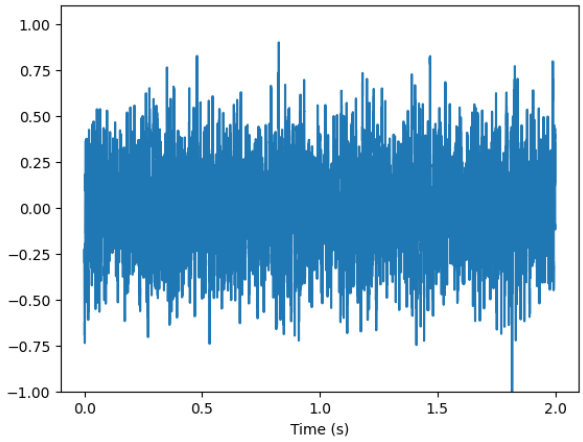
$$X(k,m) = \sum_{n=0}^{N-1} x(n+mH)w(n)e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, ..., N-1 \tag{2.1}$$

where N is the length of the Fourier transform, $x(n)$ is the signal, $w(n)$ is the window function of length N, m is the frame number, and H is the hop-size [6].

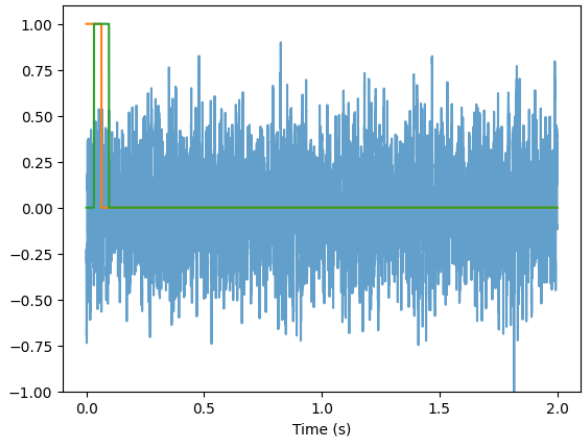
The complex-valued result of an STFT can be visualized in a more intuitive manner by taking the absolute value of the STFT, also known as a magnitude spectrogram in the following way,

$$S(k, m) = |X(k, m)|. \tag{2.2}$$

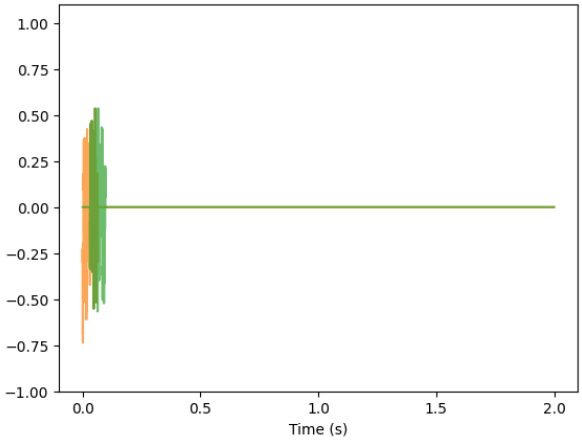
Plotting each of the M magnitudes results in a three-dimensional image, showing how the frequency content of the signal changes over the duration of the signal. Figure 2.3 depicts an example of a spectrogram for a ten second segment of industrial fan sound.



(a) A two second long audio signal.

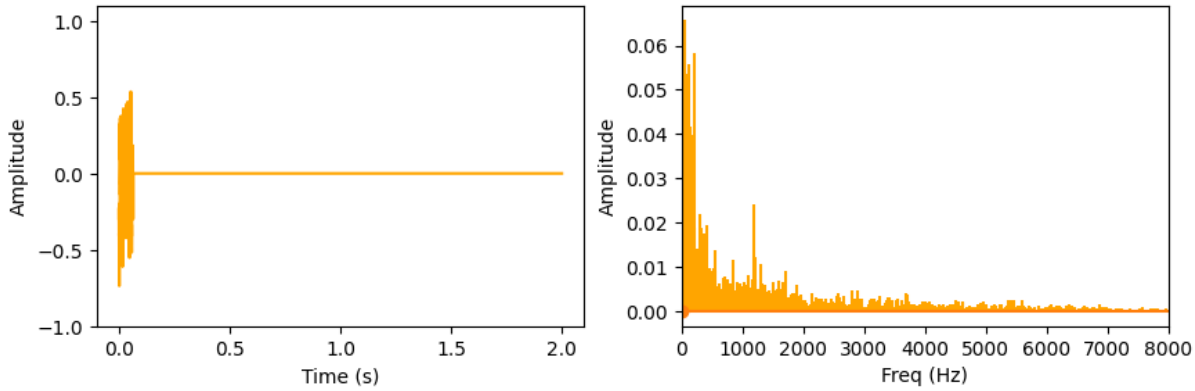


(b) The signal in Figure 2.1a and the first two consecutive windows.

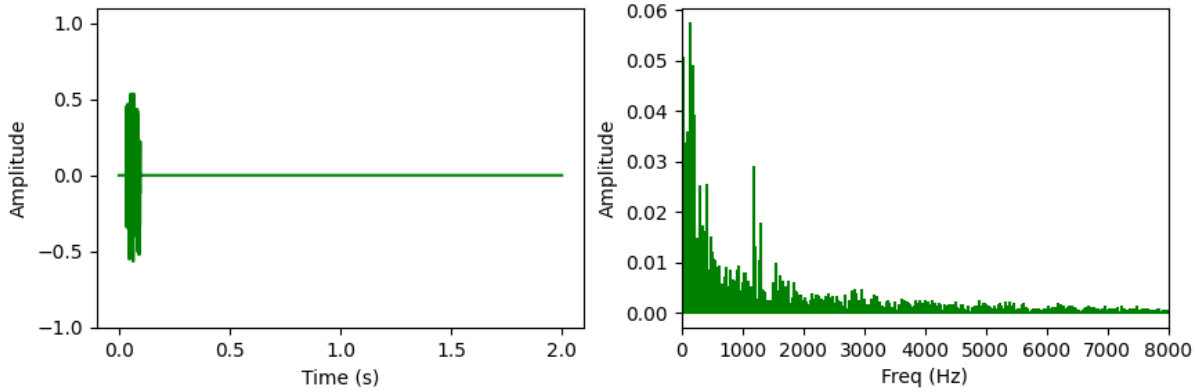


(c) The resulting windowed segments after windowing as in Figure 2.1b

Figure 2.1: A partial example demonstrating windowing a two second long signal. The result of extracting the first two overlapping windowed segments is shown.



(a) The first windowed signal from Figure 2.1c and its spectrum.



(b) The second windowed signal from Figure 2.1c and its spectrum.

Figure 2.2: A partial example showing how the windowed signals are Fourier transformed to show the frequency content of each window.

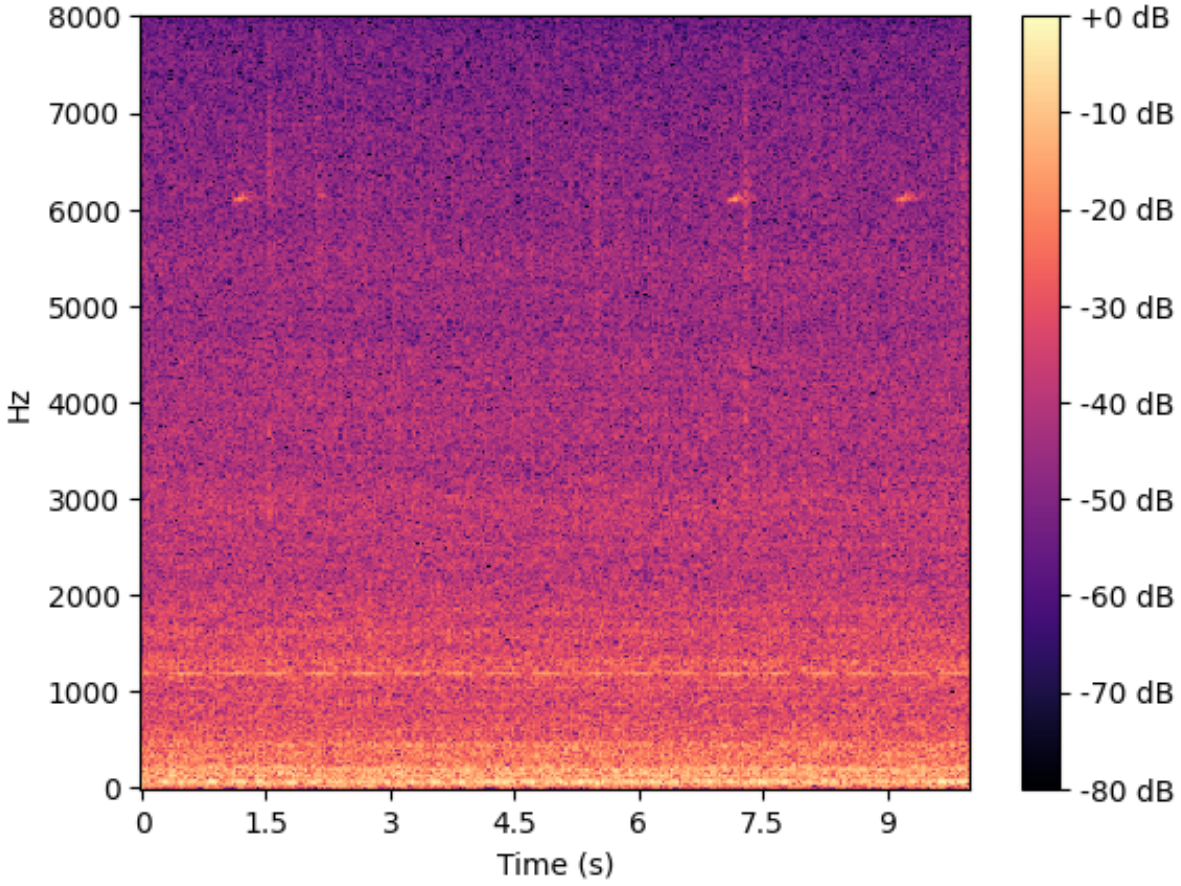


Figure 2.3: By repeating the windowing process throughout the whole signal in Figure 2.1c), Fourier transforming the windowed segments as in Figure 2.2 and aligning them on a time-axis results in the following spectrogram.

2.2 Gaussian Mixture Model & EM-algorithm

When modelling data it is common to assume that observations of a data set originate from a single distribution, for which we can estimate the parameters. However in a reality this is often not the case, as data is often much more complex, with data containing multiple regions with high probability mass. The concept of mixture models approaches this problem by assuming that the observations come from multiple unimodal distributions, or components mixed together [7].

A Gaussian mixture model (GMM) is a model combining K multivariate Gaussian distributions. Each Gaussian, called a component has its own mean μ_k and covariance $\mathbf{\Sigma}_k$ as following,

$$p\left(\mathbf{x}_i \mid \boldsymbol{\theta}\right) = \sum_{k=1}^{K} \pi_k p_k\left(\mathbf{x}_i \mid \boldsymbol{\theta}\right) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x}_i \mid \mu_k, \mathbf{\Sigma}_k\right) \tag{2.3}$$

where the mixture parameter π_k is the probability choosing the k-th component. Whereas the mixture component, $p_k(\mathbf{x}_i \mid \boldsymbol{\theta})$ represents the distribution of $\mathbf{x}_i$ given that it originates from the k-th component. In a dataset with I l-dimensional data points, $\mathbf{x}_i$, is a l-dimensional vector representing the i-th data point such that $\mathbf{x}_i = [x_{i,1}, x_{i,2}, ..., x_{i,l}]$ [7].

A visual example of a GMM can be seen in Figure 2.4 where the left plot shows two clusters of data points in two dimensions, and the right plot visualizes the result of a GMM with two Gaussian components. The respective centers of the two components are marked with an **X** each in the right plot. The right plot can be thought of as a density estimation of the two point clusters from the left plot. The darker colored regions represent a higher density of points, corresponding to a higher probablility of a data point laying in the region belonging to the cluster. Conversely the lighter regions represent lower density regions, corresponding to a lower probablility of a point laying in that region belonging to that cluster. When applying GMMs to anomaly detection the goal is to draw some boundary in the right plot, with points lying outside this boundary are to be classified as anomalies. In this example the points marked with a red star are classified as anomalies.

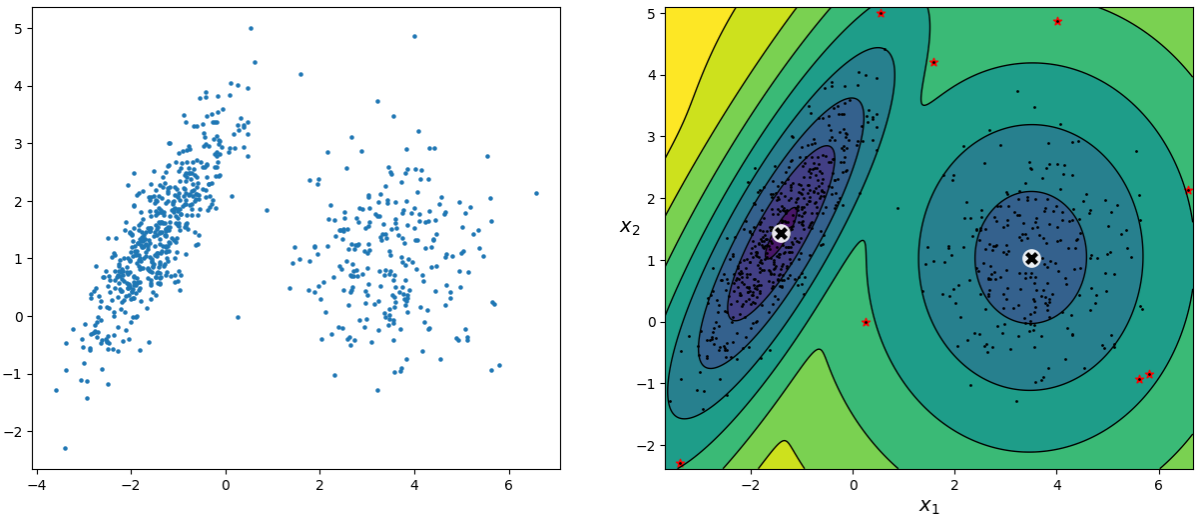


Figure 2.4: The left plot shows two clusters of points in two dimensions. The right plot visually explains how a GMM with two components is applied to the two clusters. The colored regions indicate the density of points in the region, the darker the color, the higher the probability of a point belonging to that cluster and vice versa. The red stars indicate points that are classified as anomalies.

The Expectation–maximization (EM) algorithm finds the maximum likelihood of the parameters in the model depending on the latent variables $\boldsymbol{z}$ by iterating the two steps, the expectation (E) step and the maximization (M) step. The E-step estimates a likelihood function w.r.t to the current parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and the M-step computes the maximum likelihood estimation of $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ using the likelihood function from the previous E-step. The likelihood function is as following,

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p\left(\mathbf{x}_i \mid \boldsymbol{\theta}\right) = \sum_{i=1}^{N} \log \left[\sum_{\mathbf{z}_i} p\left(\mathbf{x}_i, \mathbf{z}_i \mid \boldsymbol{\theta}\right) \right], \tag{2.4}$$

since $\boldsymbol{z}_i$ is unknown, it is not possible to compute the likelihood. Therefore the auxiliary function

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}) = \sum_i \sum_k r_{ik} \log \pi_k + \sum_i \sum_k r_{ik} \log p\left(\mathbf{x}_i \mid \boldsymbol{\theta}_k\right) \tag{2.5}$$

is introduced. $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)})$ makes use of the previous parameters $\boldsymbol{\theta}^{(t-1)}$ in the process of estimating the likelihood function. r_{ik} is the weight of point i in cluster k. The E-step is computed as following,

$$r_{ik} = \frac{\pi_k p\left(\mathbf{x}_i \mid \boldsymbol{\theta}_k^{(t-1)}\right)}{\sum_{k'} \pi_{k'} p\left(\mathbf{x}_i \mid \boldsymbol{\theta}_{k'}^{(t-1)}\right)}, \tag{2.6}$$

which relates to π_k by

$$\pi_k = \frac{1}{N} \sum_i r_{ik} = \frac{r_k}{N}, \tag{2.7}$$

where r_k is the weighted points belonging to cluster k [7].

The M-step optimizes Q with respect to, $\boldsymbol{\pi}$ and $\boldsymbol{\theta}_k$, leading to

$$\begin{aligned} \ell\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) &= \sum_k \sum_i r_{ik} \log p\left(\mathbf{x}_i \mid \boldsymbol{\theta}_k\right) = \\ &- \frac{1}{2} \sum_i r_{ik} \left[\log \left|\boldsymbol{\Sigma}_k\right| + \left(\mathbf{x}_i - \boldsymbol{\mu}_k\right)^{\mathsf{T}} \boldsymbol{\Sigma}_k^{-1} \left(\mathbf{x}_i - \boldsymbol{\mu}_k\right) \right] \end{aligned} \tag{2.8}$$

The maximum likelihood estimation lead to the following estimations of $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$

$$\boldsymbol{\mu}_k = \frac{\sum_i r_{ik} \mathbf{x}_i}{r_k}, \tag{2.9}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_i r_{ik} \left(\mathbf{x}_i - \boldsymbol{\mu}_k\right) \left(\mathbf{x}_i - \boldsymbol{\mu}_k\right)^{\mathsf{T}}}{r_k} = \frac{\sum_i r_{ik} \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}}}{r_k} - \boldsymbol{\mu}_k \boldsymbol{\mu}_k^{\mathsf{T}}. \tag{2.10}$$

2.3 Support Vector Machines (SVM)

The section will begin with an explanation of the original linear classifier from the 1960s and carry on to introduce the necessary extensions for the nonlinearly separable cases, and end with the one-class classification version. For accessibility the figures are bounded to the two dimensions of the paper, however, the properties of the SVM machinery carry over to any dimensionality.

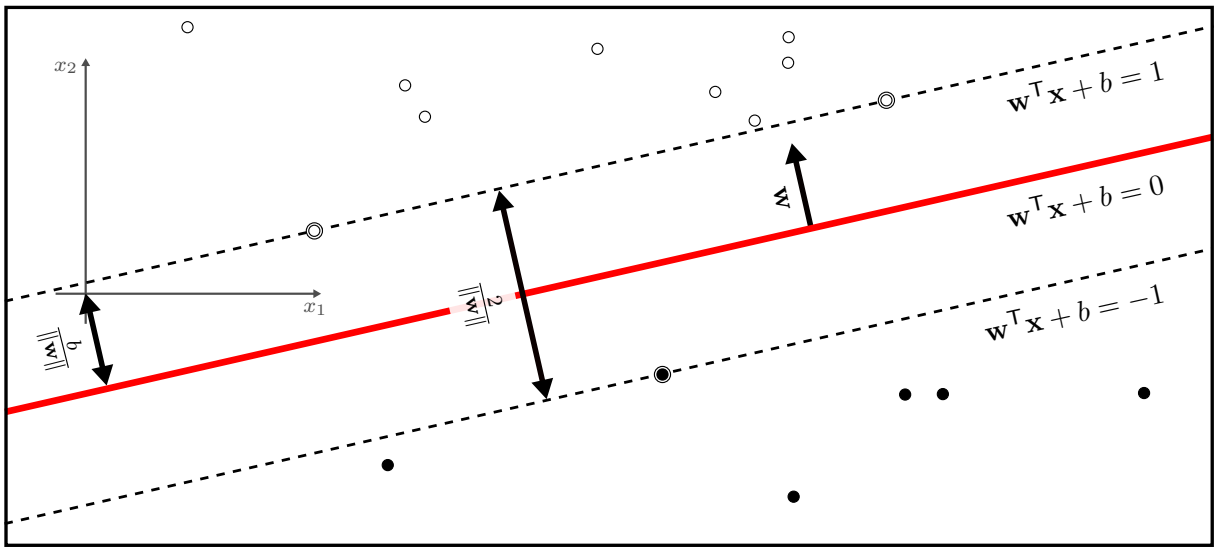


Figure 2.5: A maximum-margin hyperplane and margins for an SVM trained with samples from two classes. The marked samples lie on the margin boundaries and are called support vectors.

2.3.1 The linear classifier

SVM, based on the Vapnik-Chervonenkis theory, are non-probabilistic binary linear classifiers [8]. Given a set of labelled training samples, the SVM finds the parameters to the maximum-margin hyperplane,

$$\mathbf{w}^\mathsf{T}\mathbf{x} + b = 0, \tag{2.11}$$

that separates the two classes in the sample space. New points are classified according to what side of the hyperplane they belong using

$$\hat{y} = \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x} + b) \quad \hat{y} \in \{-1, 1\}, \tag{2.12}$$

following the convention that sign(a) equals 1 for $a \geq 0$ and -1 otherwise. Looking at Figure 2.5, the points are linearly separated by the hyperplane with the largest margin to the points on either side, minimizing the generalization error. The points that fall on the margin boundary, the dotted lines in the figure, are called support vectors and they completely determine the hyperplane. In this case, when the points are linearly separable, the margin is called a hard-margin, which means that no points are allowed to fall within the margin boundary.

The hard-margin duality optimization problem may be formulated as follows. Given a training set containing M data points,

$$(\mathbf{x}_i, y_i) \quad y_i \in \{-1, 1\} \quad i = 1, \ldots, M,$$

with $\mathbf{x_i}$ representing a sample vector and the corresponding y_i being the class of the sample. For the maximum-margin hyperplane the constraints

$$\mathbf{w}^\mathsf{T}\mathbf{x}_i + b \geq 1, \quad \text{if } y_i = 1 \tag{2.13}$$

and

$$\mathbf{w}^\mathsf{T}\mathbf{x}_i + b \leq -1, \quad \text{if } y_i = -1 \tag{2.14}$$

say that all sample points must lie on or on their side of the respective margin, as per the rules of the hard-margin. Maximizing the margin $\frac{2}{\|\mathbf{w}\|}$ under constraints (2.13) and (2.14) is equivalent with

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w}, \\ &\text{subject to} \quad y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \geq 1 \quad \forall i \in \{1, \ldots, M\}. \end{aligned} \tag{2.15}$$

In mathematical optimization theory (2.15) is called a primal problem and with the help of Lagrangian multipliers under Karush-Kuhn-Tucker conditions the dual perspective is

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{k=1}^{M} y_i y_k \alpha_i \alpha_k \mathbf{x}_i^\mathsf{T}\mathbf{x}_k, \\ &\text{subject to} \quad \sum_{i=1}^{M} \alpha_i y_i = 0, \text{ and } \alpha_i \geq 0 \quad \forall i \in \{1, \ldots, M\}, \end{aligned} \tag{2.16}$$

which is a convex quadratic programming problem with a standard optimization solution. α_i are called dual coefficients. From the dual formulation the relationship between $\boldsymbol{\alpha}$ and $\mathbf{w}$ is

$$\mathbf{w} = \sum_{i=1}^{M} \alpha_i y_i \mathbf{x}_i. \tag{2.17}$$

$\alpha_i = 0$ when the point is on the correct side of the margin, otherwise when $\alpha_i > 0$ the point is on the margin boundary, i.e. $\mathbf{x}_i$ is a support vector. Consequently, $\mathbf{w}$ is a linear combination of only the support vectors. b can be found by solving

$$y_{sv}(\mathbf{w}^\mathsf{T}\mathbf{x}_{sv} + b) = 1 \iff b = y_{sv} - \mathbf{w}^\mathsf{T}\mathbf{x}_{sv} \quad \forall sv : \alpha_{sv} > 0, \tag{2.18}$$

for any support vector $\mathbf{x}_{sv}$ with label y_{sv}. Note that if the points are not linearly separable the path going from (2.15) to (2.16) breaks down.

2.3.2 Nonlinearly separable data

Extending SVM to handle the situation when the points are nonlinearly separable is typically done by combining two complementing techniques. Figure 2.6 shows two types of point swarms. In the swarm on the left all but two outlier points called nonmargin support vectors are linearly separable, the points are *slightly* nonseparable. For this type of setup a hyperplane is used with a soft-margin, allowing but penalizing margin violations, thus avoids fitting noise. A soft-margin can be constructed using the hinge loss function

$$\max\left(0, 1 - y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b)\right). \tag{2.19}$$

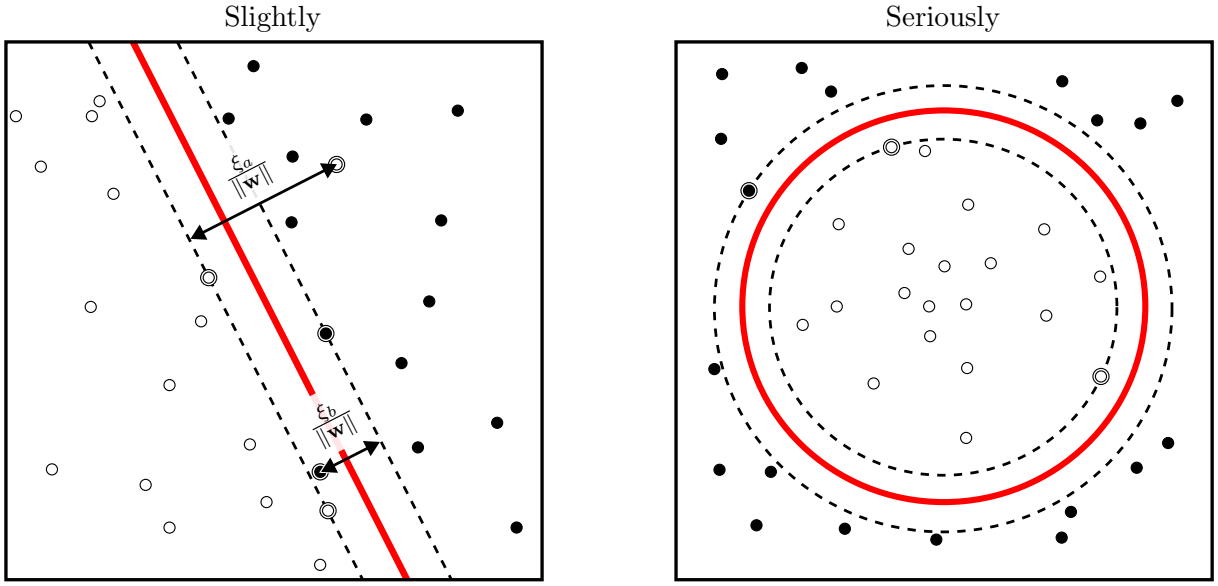


Figure 2.6: Two cases when points from two classes are not linearly separable. The red boundaries illustrating the hyperplanes achieved using a soft-margin for the slightly case and a kernel method for the seriously case. $\xi_a, \xi_b > 0$, is the amount of margin violation for each of the two offending points on the left.

Incorporating the hinge loss function using the slack variable $\boldsymbol{\xi}$ in the objective function results in the soft-margin primal,

$$\begin{aligned} \underset{\mathbf{w},b,\xi}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} + C\sum_{i=1}^{M}\xi_i, \\ \text{subject to} \quad & y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \geq 1 - \xi_i, \text{ and } \xi_i \geq 0 \quad \forall i \in \{1,\ldots,M\}. \end{aligned} \tag{2.20}$$

The penalty term C controls the strength of the penalty $\boldsymbol{\xi}$, and acts as an upper bound for α_i in the dual problem,

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{M}\alpha_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{k=1}^{M} y_i y_k \alpha_i \alpha_k \mathbf{x}_i^\mathsf{T}\mathbf{x}_k, \\ \text{subject to} \quad & \sum_{i=1}^{M}\alpha_i y_i = 0, \text{ and } 0 \leq \alpha_i \leq C \quad \forall i \in \{1,\ldots,M\}. \end{aligned} \tag{2.21}$$

Apart from the restriction in the equality constraint, the soft-margin (2.21) is identical to the hard-margin (2.16) dual optimization problem. After solving the quadratic programming problem of the Lagrangian, the hyperplane parameters are obtained from (2.17), and (2.18) with the added constraint $\forall sv : \alpha_{sv} < C$, singling out the margin support vectors. Classification is performed with (2.12).

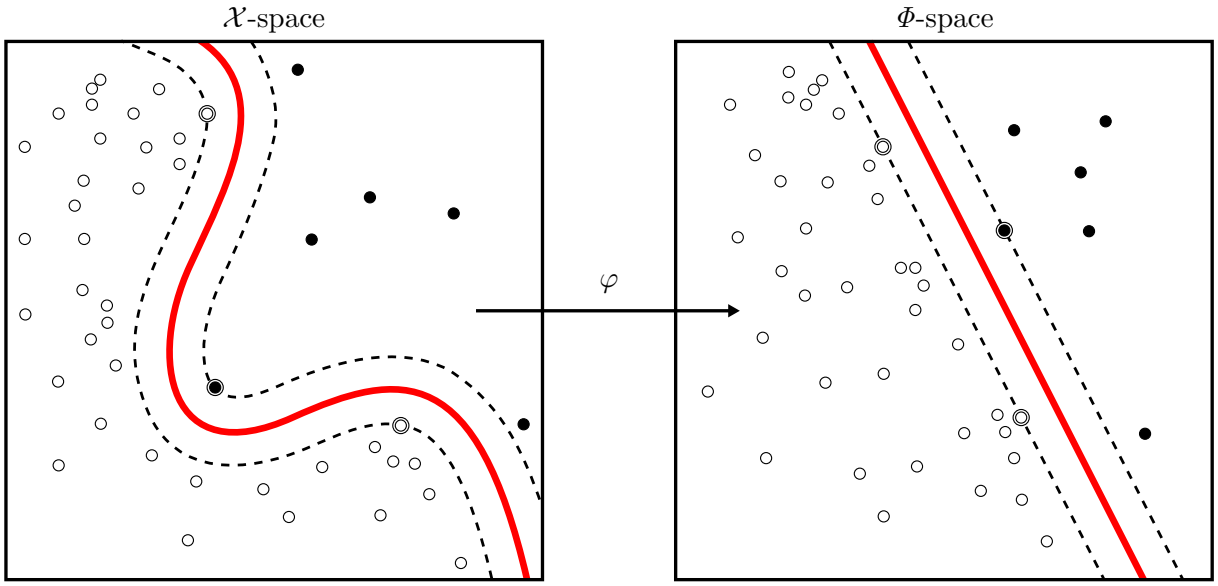


Figure 2.7: A kernel machine involves solving a nonlinear problem in the $\mathcal{X}$-space by utilizing a transformation φ to map the input to a high-dimensional Φ-space where a linear classifier is effective. Graphic adopted from Wikimedia Commons [9].

For the *seriously* swarm on the right in Figure 2.6 there is no good linear separation of the points. However, there is a clear circular separation between the two classes, which can be found by applying a kernel method. Figure 2.7 shows how a kernel machine can learn a nonlinear separation by transforming the input data to a high-dimensional Φ-space where the points are linearly separable. The computational cost of the transformation φ scales with respect to the dimensionality of Φ-space, however there is often no need to calculate φ as in the case for SVM. Due to exclusively needing the proper inner products of pairs of points from Φ-space it is possible to use the kernel trick. The kernel trick relies on the kernel function,

$$K(\mathbf{x}_i, \mathbf{x}_k) = \big\langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_k) \big\rangle = \varphi(\mathbf{x}_i)^\mathsf{T} \varphi(\mathbf{x}_k), \tag{2.22}$$

to take advantage of Φ-space while avoiding most of the expensive computational work of transforming the coordinates with φ. For SVM the computational cost is instead only relative to the number of training samples. By swapping out $\mathbf{x}_i$ with $\varphi(\mathbf{x}_i)$ in the soft-margin optimization problem (2.21), the Lagrangian dual turns into

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{k=1}^{M} y_i y_k \alpha_i \alpha_k K(\mathbf{x}_i, \mathbf{x}_k), \\ \text{subject to} \quad & \sum_{i=1}^{M} \alpha_i y_i = 0, \text{ and } 0 \leq \alpha_i \leq C \quad \forall i \in \{1, \ldots, M\}. \end{aligned} \tag{2.23}$$

Correspondingly with the weights,

$$\mathbf{w} = \sum_{i=1}^{M} \alpha_i y_i \varphi(\mathbf{x}_i), \tag{2.24}$$

and the bias,

$$b = y_{sv} - \mathbf{w}^\mathsf{T} \varphi(\mathbf{x}_{sv}) = y_{sv} - \sum_{i=1}^{M} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_{sv}) \quad \forall sv : 0 < \alpha_{sv} < C, \tag{2.25}$$

the classification function becomes

$$\hat{y} = \text{sign}(\mathbf{w}^\mathsf{T} \varphi(\mathbf{x}) + b) = \text{sign}\left(\sum_{i=1}^{M} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right). \tag{2.26}$$

Note that the support vectors only exist in the space that the support vector machinery is working in. For a kernel machine it is in Φ-space, which means $\mathbf{x}_{sv}$ are only pre-images of the support vectors $\varphi(\mathbf{x}_{sv})$ as in Figure 2.7.

One of the most popular kernels to use with SVM is the Radial Basis Function (RBF),

$$K_{RBF}(\mathbf{x}_i, \mathbf{x}_k) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_k\|^2), \tag{2.27}$$

which implicitly operates in an infinite dimensionality without paying the computational price for it. γ adjusts the reach of influence of a single training example and is dependent on the input data. A rule of thumb is to scale γ according to $\gamma = 1/(\text{Dimension}_{\mathbb{R}}(\mathcal{X}) * \text{Variance}(\mathbf{x}_1, \ldots, \mathbf{x}_M)$. Luckily, there is never a need to evaluate $\mathbf{w}$ in (2.24), an otherwise endless task when working with the RBF kernel [10].

2.3.3 One-Class Support Vector Machines (OCSVM)

In its bare form SVM is a binary classifier, meaning it is capable of working with points from two classes. Adaptations of SVM exist for Multi-Class classification, predicting for more than 2 labels, and One-Class classification, for outlier and novelty detection. Typically Multi-Class classification is accomplished by combining multiple SVMs in a one-vs-all or one-vs-one fashion for each combination of classes. In the One-Class version the points are separated from the origin [11]. The parameter ν is introduced and acts as an upper bound of the fraction of training errors and a lower bound of the fraction of support vectors. Figure 2.8 shows, for two values of ν, how the hyperplanes might look like for the same set of samples. With the new problem formulation the objective function looks like

$$\begin{aligned} \underset{\mathbf{w},\rho,\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} - \rho + \frac{1}{\nu M}\sum_{i=1}^{M} \boldsymbol{\xi}_i, \\ \text{subject to} \quad & \mathbf{w}^\mathsf{T}\varphi(\mathbf{x}_i) \geq \rho - \boldsymbol{\xi}_i, \text{ and } \boldsymbol{\xi}_i \geq 0 \quad \forall i \in \{1, \ldots, M\}, \end{aligned} \tag{2.28}$$

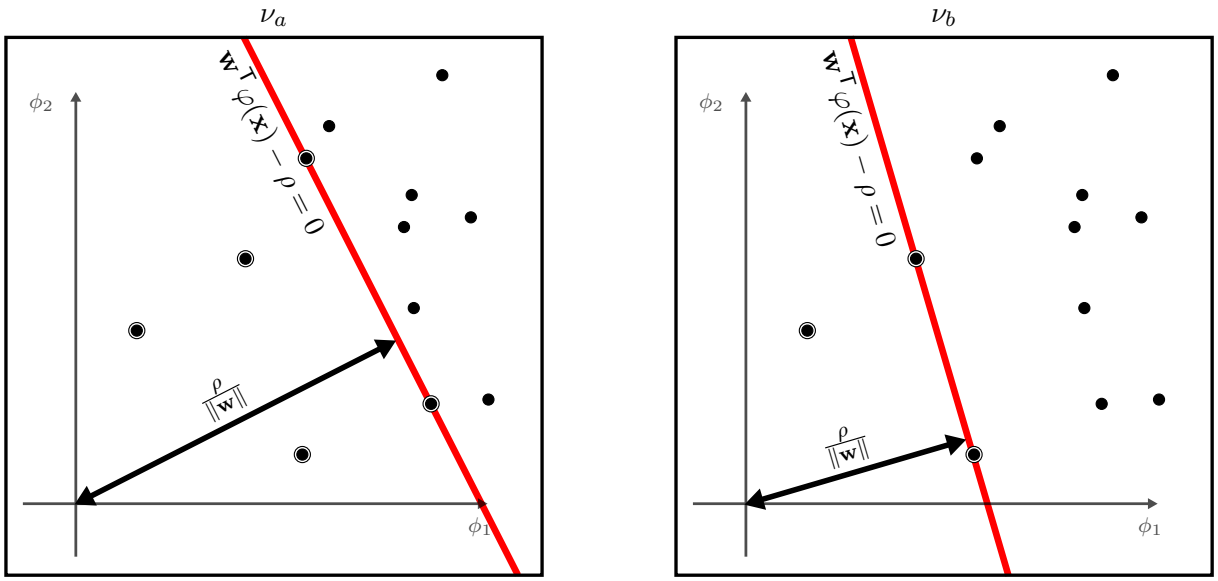


Figure 2.8: Two OCSVM hyperplanes separating the training samples from the origin illustrating the effect of the ν parameter. $0 < \nu_b < \nu_a \leq 1$, meaning ν_b allows less outliers and the use of less support vectors compared to ν_a.

and when solved for the dual Lagrangian the optimization problem emerge as

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & \mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{M}\sum_{k=1}^{M}\alpha_i\alpha_k K(\mathbf{x}_i, \mathbf{x}_k), \\ \text{subject to} \quad & \sum_{i=1}^{M}\alpha_i = 1, \text{ and } 0 \leq \alpha_i \leq \frac{1}{\nu M}, \quad \forall i \in \{1, \ldots, M\}. \end{aligned} \tag{2.29}$$

With the parameters of the hyperplane,

$$\mathbf{w} = \sum_{i=1}^{M}\alpha_i\varphi(\mathbf{x}_i), \tag{2.30}$$

and

$$\rho = \mathbf{w}^\mathsf{T}\varphi(\mathbf{x}_{sv}) = \sum_{i=1}^{M}\alpha_i K(\mathbf{x}_i, \mathbf{x}_{sv}) \quad \forall sv : 0 < \alpha_{sv} < \frac{1}{\nu M}, \tag{2.31}$$

novel samples are distinguished using

$$\hat{y} = \text{sign}(\mathbf{w}^\mathsf{T}\varphi(\mathbf{x}) - \rho) = \text{sign}\left(\sum_{i=1}^{M}\alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho\right). \tag{2.32}$$

2.3.4 Feature scaling

The SVM algorithm struggles with performance when the scaling and offset differs between features. This is because of how the SVM works by optimizing distances to the hyperplane. A value change from -100 to 100 could be as significant for one feature as a jump between 0 and 1 for another, but for the SVM the first change would have a stronger influence.

Therefore, scaling is applied to the samples prior to training and inference, after which the standardized features have a mean of 0 and a unit variance scaling.

2.4 Signal-to-disturbance ratio (SDR)

The concept of SDR is introduced in order to relate the magnitude of a signal to the magnitude of a disturbance affecting said signal, it is needed to compare how the relation between signal and disturbance affects the detectability of an anomaly. The SDR is defined as follows,

$$SDR = \frac{\frac{1}{T_s}\sum_{t=0}^{T_s} a_s(t)^2}{\frac{1}{T_d}\sum_{t=0}^{T_d} a_d(t)^2}, \tag{2.33}$$

where T_s is the duration of the signal s, $a_s(t_s)$ is the amplitude of s at time t. Similarly T_d is the duration of the disturbance d, $a_d(t_d)$ is the amplitude of d at time t. In this definition of SDR, a lower SDR value means that there is a lot of disturbance compared to signal, meaning that it sbould be easier to separate the disturbance of the signal compared to the case of a higher SDR value.

2.5 Classification metrics

The metrics Precision, Recall and F1-score were used in order to evaluate the anomaly detection performance of the models. They are defined as follows,

$$\text{Precision } = \frac{tp}{tp + fp} \tag{2.34}$$

$$\text{Recall } = \frac{tp}{tp + fn} \tag{2.35}$$

$$F_1 = 2 \cdot \frac{\text{precision } \cdot \text{ recall}}{\text{precision } + \text{ recall}}, \tag{2.36}$$

where tp is the true positives, fp is the false positives and fn is the false negatives. $0 \leq$ Precision, Recall, F1≤ 1, with a score closer to 1 being better and closer to 0 being worse.

2.6 Real-time factor (RTF)

In order to measure the speed of inferring whether a segment of sound is to be considered an anomaly or not the concept real-time factor (RTF) is introduced. RTF is a metric that measures the speed of a system by relating the processing time p_t of a signal s, to the duration, T_s of s. It is defined as follows,

$$RTF = \frac{p_t(s)}{T_s}. \tag{2.37}$$

It is useful when evaluating if a system is fast enough to process data in real-time. In order to process data in real-time it is required that $RTF \leq 1$.

Chapter 3

Data

The data used for the experiments is divided into two distinct sets. The first set is based on the normal data from DCASE 2020 Challenge Task 2 that is then augmented with simulated anomalies. The second dataset contains self produced recordings of fan sound in an office environment.

3.1 Augmented DCASE dataset (ADCASE)

The data used was based on the DCASE 2020 Challenge Task 2 Development dataset, containing ten second recordings of an industrial fan, with background noise from real factories mixed in, in order to make it more realistic [12]. The sound is sampled at 16000 Hz. The data is split into a training set and a test set. The training set only contains data of normal operating sound, while the test set contains recordings of normal operating sound, as well as recordings containing anomalies. In this thesis, the idea is to classify shorter segments than ten seconds and since the test data only annotates a whole ten second segment as containing an anomaly or not, with no information about how much of the segment is considered to be anomalous or where an anomaly would be located, the anomalous test data is not useful.

Instead, a dataset was constructed where the normal test data from the DCASE 2020 Challenge Task 2 dataset was augmented with simulated anomalies, henceforth known as the *ADCASE dataset*. It contains examples of anomalies such as a 440 Hz sine wave at a fixed length, position, and magnitude. This affords the possibility of labeling the augmented part as an anomaly and everything else in the file as normal. Simulated anomalies are used in the pre-study in order to examine whether the chosen models are able to separate an added disturbance from a normal signal at all.

The simulations made use of normal test samples from the DCASE dataset [12]. The test samples were modified by adding a tone in the form of a sine wave. In the resulting segment, each frame is labeled either as anomalous - for the frames that we modified - or normal for the frames that were unmodified. Figure 3.1 consists of four different plots (a-d), aiming to show how a disturbance can be visualized. Figure 3.1 a) shows a normal signal from the DCASE

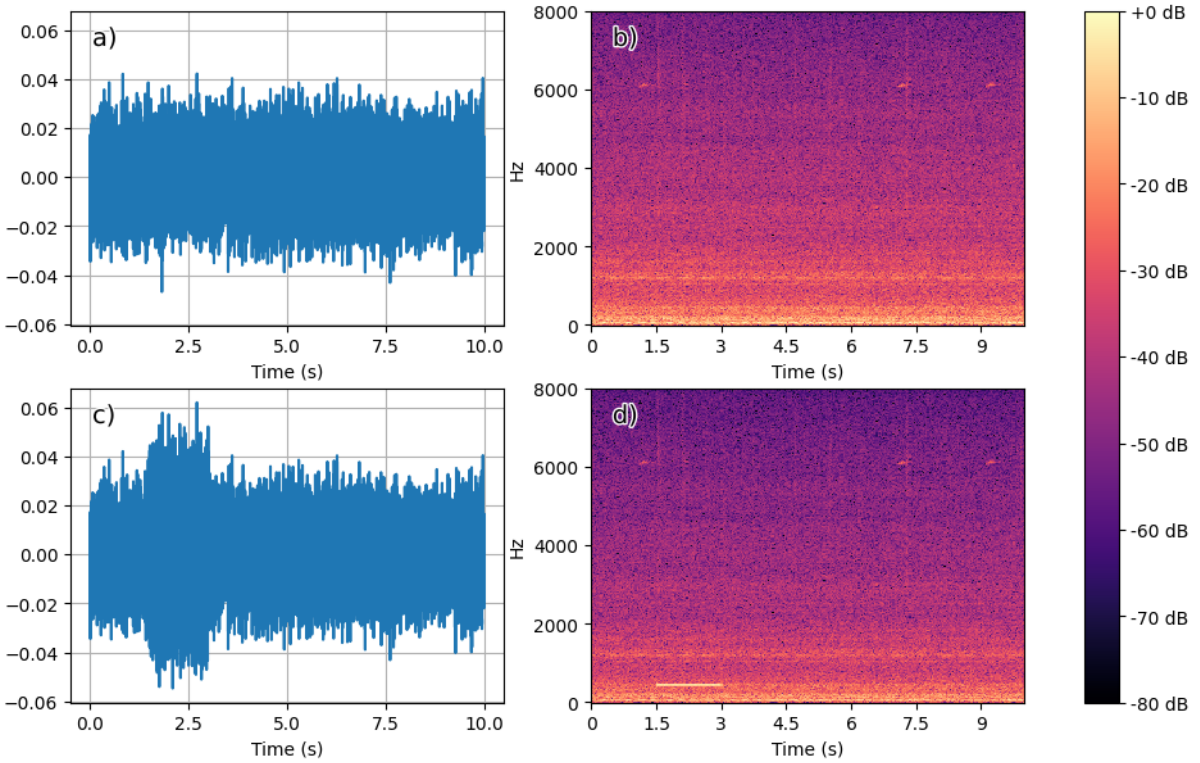


Figure 3.1: a) A signal without any anomalous disturbance. b) The spectrogram corresponding to the signal in a). c) Signal a) augmented with a simulated anomaly. d) The spectrogram corresponding to signal c). The anomaly is a 440 Hz sine wave in the interval $[1.5, 3]$ seconds with an SDR of 0.4.

dataset and its spectrogram in b). In c) and d) below, the same signal and its spectrogram are shown augmented with a disturbance between 1.5 and 3 seconds in the form of a 440 Hz sine wave with SDR 0.4. The bottom two plots in Figure 3.1 clearly visualize the added disturbance. In plot c), it can be seen as increasing amplitude values in the interval where the anomaly is added. The spectrogram in Figure 3.1 d) shows the disturbance as a larger intensity in the interval of $[1.5, 3]$ seconds, around the frequency of the disturbance.

3.2 Self produced recordings

There were two problems that motivated the creation of an additional new dataset. While the ADCASE dataset solved the problem of annotating where an anomaly occurred, it is not very realistic as it is unlikely that potential anomalies would take the shape of a pure sine tone. Furthermore, it was desired to test the models with chunks of sound shorter than the ten seconds of the ADCASE dataset. Since there is no way of knowing where an anomaly is located in one of the ten-second-long anomalous files, it was not possible to split them up into smaller chunks. Hence it was decided that it would be favorable to record sound where it was possible to choose how large chunks to process at a time and where anomalies are located. The new dataset was

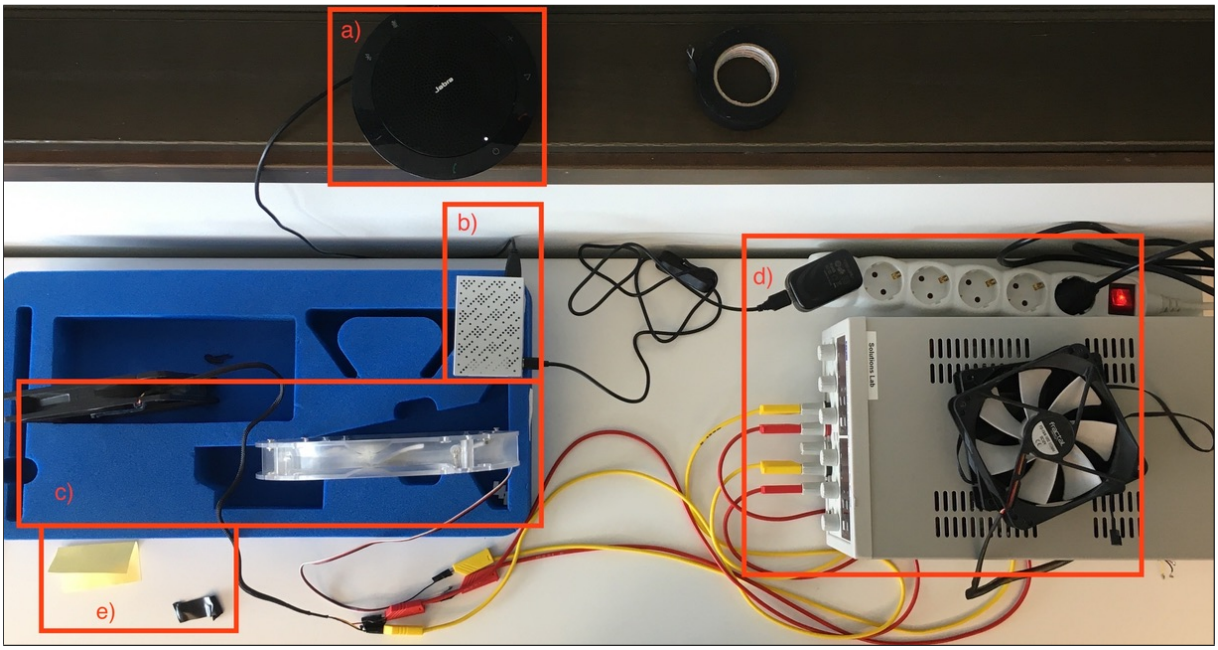


Figure 3.2: The experiment station for sound recording and performing live trials. a) Recording device, Jabra Speak 510. b) Computer, Raspberry Pi 4 model B 4GB. c) Computer case fans. d) Power supplies for driving the Raspberry Pi and the fans, and spare fans. e) Disturbance invoking tools.

created by recording sound on a Raspberry Pi 4B using a Jabra Speak 510 external microphone connected via USB, sampled at 16000 Hz. Figure 3.2 depicts the setup used to record sound. A computer case fan - visible in Figure 3.2 c) - was used to produce sound used as a signal. The recordings also include background noise from the office environment where the setup is located.

Chapter 4

Method

The method section describes how the theory and the data presented in the previous two chapters are used to create models for anomaly detection. Section 4.1 introduces the choice of the windowing function used when performing the STFT. Continuing on, Section 4.2 introduces the different feature sets that were evaluated in modeling the anomaly detection problem. Lastly, Section 4.3 introduces the method of thresholding used to decide if an anomaly score is large enough for the corresponding sample to be marked as an anomaly.

4.1 Windowing

The signal is split up into frames, each frame containing 1024 samples, corresponding to $N = 1024$ and each frame overlapping by 50%, corresponding to $H = 512$. The windowing function, $w(n)$ is the Hamming window defined as

$$w\left(n\right) = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \leq n \leq N-1, \tag{4.1}$$

where N is the number of samples or the width of the window [13].

4.2 Features

Six groups of features were extracted from the framed signals. The first three groups of features are the time domain, spectral domain, and Mel-frequency cepstral coefficients (MFCC). In addition to these three groups, there is the concept of delta (Δ) features. Delta features are defined as the difference between the value of a feature in two consecutive frames. The remaining three groups are constructed by adding the respective delta features to each of the three groups, time domain features, spectral domain features, and MFCC features. The resulting six groups of features, now known as Time, Time+Δ, Spectral, Spectral+Δ, MFCC, and MFCC+Δ. These six feature sets will be evaluated against each other in order to discern which set is most suitable for the anomaly detection problem.

4.2.1 Time-domain features

The time-domain feature set consists of three metrics, zero crossing rate, energy, and energy entropy. All of the time-domain features are extracted from the signal samples. Zero crossing rate is a metric that measures how many times a signal crosses the x-axis within a frame. The energy feature measures the normalized sum of square energy in a frame. Energy entropy is the entropy of the normalized energy of subsections of a frame, it is a measure of sudden changes in energy [14]. For each of the three metrics the delta features are also computed, corresponding to the difference between the value of the metric for two consecutive frames, leading to two groups, time-domain features and time-domain plus delta time-domain features respectively.

4.2.2 Spectral-domain features

The spectral-domain feature set contains five features all derived from the magnitude spectrum of the Fourier transform of a frame. The five features are spectral centroid, spectral spread, spectral entropy, spectral flux, and spectral roll-off. Spectral centroid is a measure of the center of mass of a spectrum. Spectral spread measures the spread of the spectrum around the spectral centroid. The spectral entropy of a frame is a measure of the entropy of the spectral energies over a set of subframes. Spectral flux is a measurement of the square magnitude difference between two consecutive frames. Spectral roll-off is the frequency where cut-off below which 90% of the magnitude is located [14]. In addition to the five features, the delta values of each feature, for all consecutive frames are computed, resulting in two groups, spectral-domain features and spectral-domain plus delta spectral-domain features respectively.

4.2.3 Mel-frequency cepstral coefficients (MFCC)

MFCCs are a set of features commonly used in speaker recognition. MFCCs are based on the mel-scale. The human perception of pitch is not linear, humans are better able to distinguish a difference in frequency in low frequencies compared to high frequencies. The mel-scale relates how humans perceive sound to its actual frequency in hertz, such that the perceived difference in pitch is the same for a fixed difference in mels. While MFCCs are optimized for human speech as opposed to industrial machines, they could still be useful as they capture an overview of the spectral characteristics of a frame [2].

The filterbank consists of $Q = 40$ filters with equal area in the frequency range $[133, 6854]$ Hz, where $Q = Q_{linear} + Q_{log}$. The first 13 filters corresponding to $Q_{linear} = 13$ filters are linearly spaced, with a step of 66.67 Hz and center frequencies (CF) in the range $[200, 1000]$ Hz. The following 27 filters are spaced on a logarithmic scale, with CFs in the range $[1071, 6400]$ Hz according to

$$logStep = \exp\left(\ln\left(\frac{f_{c_{40}}}{1000}\right)/Q_{log}\right), \tag{4.2}$$

where $f_{c_{40}}$ is the 40-th, and last center frequency. $Q_{log} = 27$ represents the 27 filters that are logarithmically spaced. The filterbank consists of triangular filters computed as

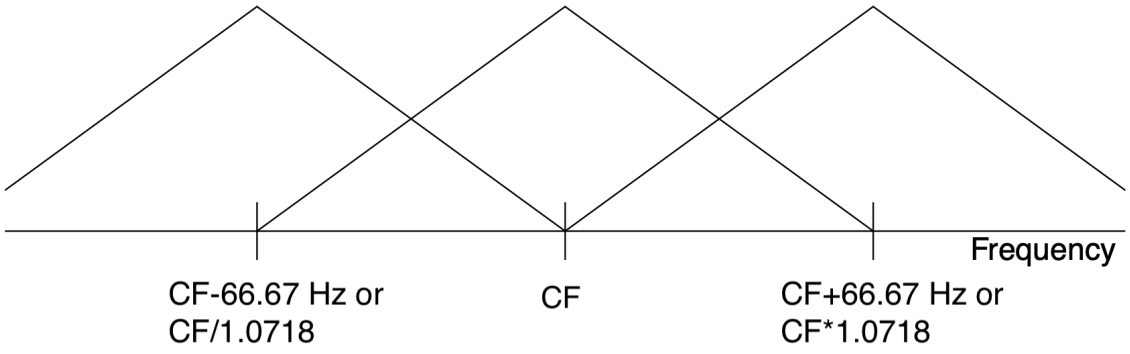


Figure 4.1: Triangular filter spacing [15].

$$H_i(k) = \begin{cases} 0 & \text{for} \quad k < f_{b_{i-1}} \\ \frac{2\left(k - f_{b_{i-1}}\right)}{\left(f_{b_i} - f_{b_{i-1}}\right)\left(f_{b_{i+1}} - f_{b_{i-1}}\right)} & \text{for} \quad f_{b_{i-1}} \leq k \leq f_{b_i} \\ \frac{2\left(f_{b_{i+1}} - k\right)}{\left(f_{b_{i+1}} - f_{b_i}\right)\left(f_{b_{i+1}} - f_{b_{i-1}}\right)} & \text{for} \quad f_{b_i} \leq k \leq f_{b_{i+1}} \\ 0 & \text{for} \quad k > f_{b_{i+1}} \end{cases} \tag{4.3}$$

Where i is the i-th of the 40 filters, f_{b_i} represents the boundary points where a triangular filter begins and ends. k represents the k-th frequency bin of the STFT. The factor, $2/\left(f_{b_{i+1}} - f_{b_{i-1}}\right)$, is included in 4.3 in order to ensure that all of the filters are of equal area. The resulting filter bank after normalization is depicted in Figure 4.2.

To get the log energies in each filter, the logarithm is computed on the result of applying the filterbank (4.3) on the magnitude (2.2) of the STFT. The last step to acquire the MFCC is to apply the Discrete Cosine Transform to the log energies as follows,

$$C_p = \sum_{i=1}^{Q} Y_i \cdot \cos\left(p \cdot (i - 1/2) \cdot \frac{\pi}{Q}\right), \quad p = 1, 2, \ldots, P, \tag{4.4}$$

where C_p is the p-th coefficient, Q is the number of filters in the filterbank and Y_i is the log-energy in the i-th filter computed as,

$$Y_i = \log_{10}\left(\sum_{k=0}^{N-1} S(k, m) \cdot H_i(k)\right), \quad i = 1, 2, \ldots, Q, \tag{4.5}$$

for all of the m framed segments [16].

The first 13 coefficients are kept as features, corresponding to $P = 13$ [14]. In addition to the 13 MFCC, the respective delta values for coefficients of consecutive frames are computed, resulting in two groups, MFCC features and MFCC plus delta MFCC features respectively.

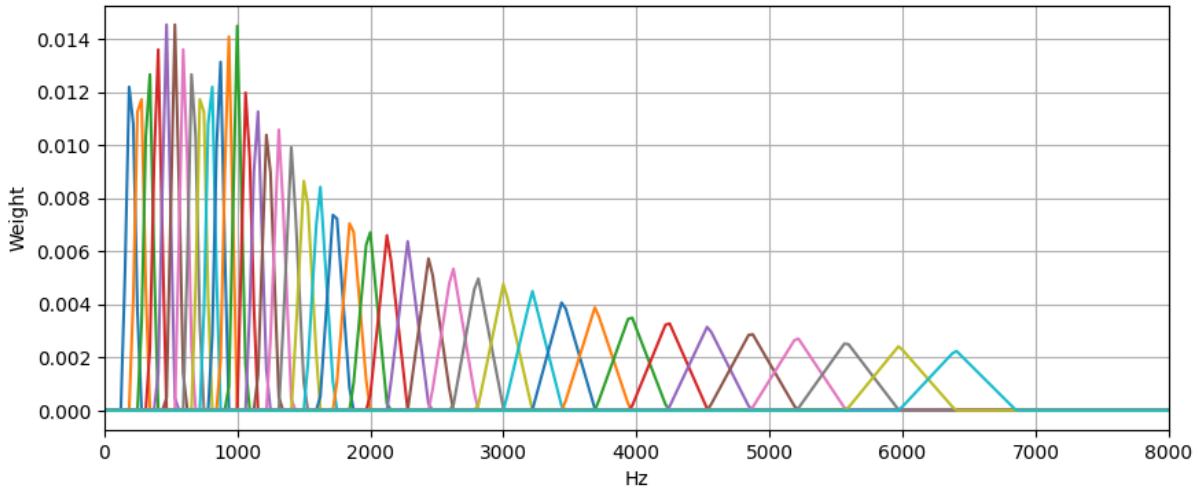


Figure 4.2: Equal area mel filter bank containing 40 filters.

4.3 Thresholding

The frames are classified as anomalies on a frame-by-frame basis for both models. Figure 4.3 show - from top to bottom - a normal signal, that same signal with a disturbance added, and the anomaly score of each frame together with the anomaly threshold. For the GMM-based model, the anomaly score is the negative log-likelihood of a sample frame, and the threshold is the 99-th percentile of the negative log-likelihood scores for the training data. For the SVM-based models the threshold is simply the hyperplane and the score is the negative output from the decision function, i.e. the negative of the signed Euclidean distance to the hyperplane, $-(\mathbf{w}^\mathsf{T}\varphi(x) - \rho)$. Every frame with a score above the threshold is classified as an anomaly. An example of thresholding for a GMM model can be seen in Figure 4.3 c), where the red dashed line represents the threshold.

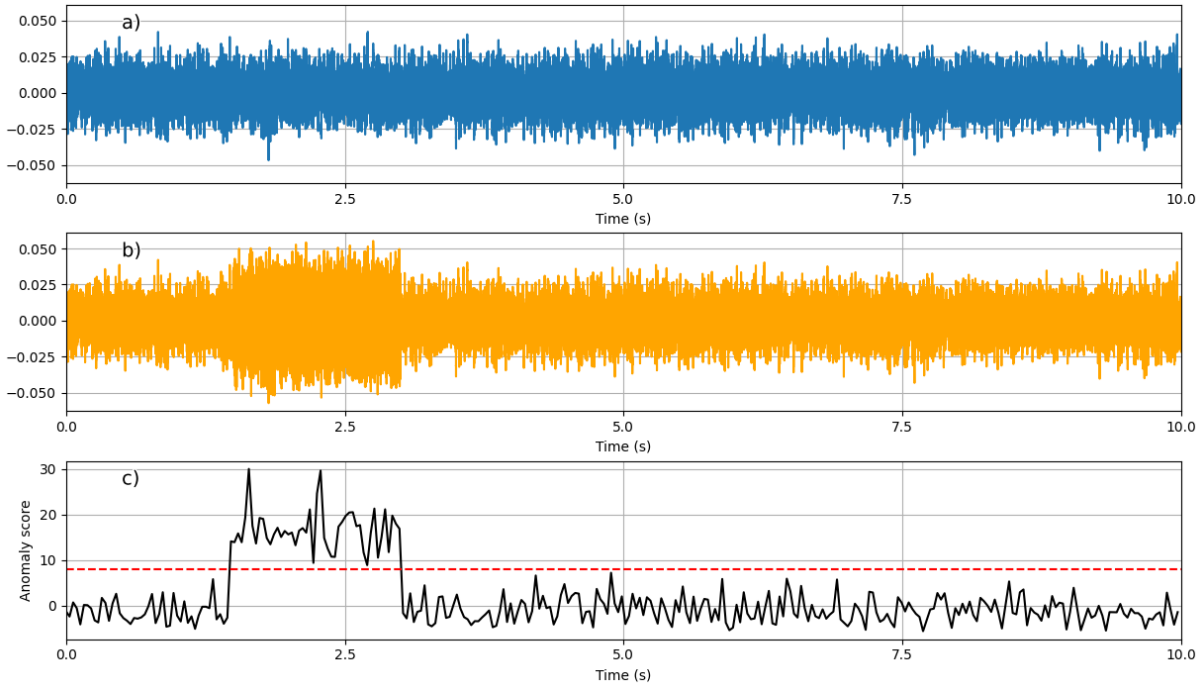


Figure 4.3: a) A ten-second audio signal. b) The signal in a) with an added disturbance with SDR 0.4. c) The negative log-likelihood scores of each frame in signal from plot b) and the anomaly threshold for a GMM-based model.

Chapter 5

Evaluation

This chapter focuses on evaluating the suitability of the two models, the GMM and the OCSVM, for the anomaly detection task. The evaluation is split into two parts. The first part aims to evaluate whether the anomaly detection performance is good enough to detect sound anomalies in a signal, such as sound from a fan. The second part of the evaluation focuses on the speed of the models, both in terms of RTF, but also in terms of training time.

The chapter starts out by introducing the different experiments and the conditions which the experiments were conducted under, it then goes on to present the results of said experiments.

5.1 Experiment setup

The experiments are divided up into two categories. The anomaly detection performance experiments investigate whether it is at all possible to distinguish anomalies from normal signals. These experiments use the ADCASE dataset. The second category of experiments concerns real-time anomaly detection using self-produced recordings of fan sound. These experiments first evaluate whether the prerequisites for running real-time anomaly detection are fulfilled.

5.1.1 Anomaly detection performance

This subsection consists of experiments analyzing how SDR, frequency, and duration of a disturbance affect the ability of a model to separate the disturbance from the signal. The experiments are performed on the ADCASE dataset, consisting of 100 normal ten-second segments of training data and a test dataset containing 100 ten-second segments containing anomalies. There are three main experiments in this section, using the MFCC feature set. The first experiment varies the SDR in the range $[0.01, 100.00]$ while fixing the frequency and duration to 440 Hz and 1.5 s respectively. The second experiment instead fixes the SDR at 1.00 and the frequency at 440 Hz, while comparing the results of a disturbance duration between 0.05 and 1.5 s. The third experiment fixes the SDR at 1.00 and the duration to 1.5 s, while letting the frequency take on the values 440, 110 and 55 Hz and observing the effect on the results. In addition to the three

experiments, an additional experiment was conducted for the time- and spectral feature sets with disturbances having 0.01 SDR, a duration of 1.5 s, and a frequency equal to 440 Hz.

The GMM-based models used diagonal covariance, 16 components, and 25 iterations. The threshold is set at the 99-th percentile, meaning that all test scores outside the 99-th percentile of normal scores are classified as anomalies by the model as seen in Figure 4.3.

The OCSVM model was trained with $\gamma = 0.1$, for the RBF kernel, and $\nu = 0.05$, dictating the upper bound fraction of outliers and the lower bound fraction of support vectors when finding the hyperplane. Additionally, the cache size was set to 2000MB. Before the data is fed to the OCSVM model it is scaled to zero mean and unit variance.

5.1.2 Training & inference speed

The real-time prerequisites are evaluated by measuring how the GMM and OCSVM models perform on limited hardware. The evaluation is done both in terms of RTF, as well as the time it takes to train the model. RTF and training time are compared both between models, but also for different lengths of training data.

The first experiment, measuring RTF listens to a segment of sound. After the segment of sound is recorded a timer starts recording the time it takes to compute the STFT of the segment, extract MFCC features, score the features according to the model, classify as an anomaly if the scores exceed the threshold, and lastly send the spectrogram, anomaly score, and threshold data to a client that can visualize it. The experiment measuring the RTF is left to run for ten minutes where all the RTF values are recorded. The segments used in the RTF measurements are 0.256 seconds long.

The second experiment, measures the training time of both models, including the time spent extracting MFCC features and computing the threshold.

The measurements for both RTF and training time are repeated for both the GMM and OCSVM, both trained on both 60 and 180 seconds of data.

5.2 Results

In this section, the results of the pre-study are presented. The section is divided up into two subsections, the first one evaluates the anomaly detection performance by measuring precision, recall, and F1-scores as in Equations 2.34, 2.35 and 2.36, for disturbances of varying magnitude, frequency and duration. The second subsection evaluates the training and inference speed of the system.

5.2.1 Anomaly detection performance

This subsection consists of three experiments each for both the GMM and OCSVM models, evaluating how the SDR, length, and frequency of a disturbance affect the possibility of detecting said disturbance using MFCC features. An additional experiment compares the performance of a couple of alternative feature sets, at an SDR where the performance of the MFCC features

starts to deteriorate. The evaluation metrics used are mean and standard deviation of precision, recall, and F1-scores for frames labeled as anomalies.

The experiment results in Table 5.1 showcase the impact of SDR values between 0.01 and 100.00. The frequency and disturbance duration are locked at 440 Hz and 1.5 s respectively.

	SDR	$\mu_{Precision}$	$\sigma_{Precision}$	μ_{Recall}	σ_{Recall}	μ_{F1}	σ_{F1}
GMM	0.01	0.8811	0.1600	1.0000	0.0000	0.9264	0.1246
	0.50	0.8778	0.1633	0.8698	0.1602	0.8496	0.1541
	1.00	0.8698	0.1736	0.6050	0.2638	0.6591	0.1972
	100.00	0.1125	0.1645	0.0447	0.1392	0.0393	0.0808
OCSVM	0.01	0.7788	0.1986	1.0000	0.0000	0.8590	0.1522
	0.50	0.7715	0.1974	0.9046	0.1187	0.8055	0.1328
	1.00	0.7332	0.1982	0.6733	0.2689	0.6361	0.1686
	100.00	0.1095	0.1160	0.0844	0.1711	0.0734	0.0997

Table 5.1: Test results for different SDR values using MFCC features. The frequency is fixed at 440 Hz and the duration of the signal is 1.5 s.

The second experiment, with results as seen in Table 5.2 evaluates the effect of changing the duration of the disturbances. The length of the disturbances is between 0.05 s and 1.50 s, with SDR and frequency fixed at 1.00 and 440 Hz respectively.

	$Duration$ [s]	$\mu_{Precision}$	$\sigma_{Precision}$	μ_{Recall}	σ_{Recall}	μ_{F1}	σ_{F1}
GMM	1.50	0.8698	0.1736	0.6050	0.2638	0.6591	0.1972
	0.50	0.7283	0.2623	0.5859	0.2650	0.5791	0.2097
	0.25	0.6424	0.2910	0.5922	0.2666	0.5310	0.2105
	0.05	0.3916	0.3284	0.5000	0.2345	0.3660	0.2318
OCSVM	1.50	0.7332	0.1982	0.6733	0.2689	0.6361	0.1686
	0.50	0.5209	0.2522	0.6524	0.2673	0.4881	0.1611
	0.25	0.3919	0.2720	0.6456	0.2846	0.3891	0.1770
	0.05	0.1984	0.2554	0.5467	0.2393	0.2200	0.1814

Table 5.2: Test results for different disturbance durations using MFCC features. The frequency and SDR of the disturbance is fixed at 440 Hz and 1.00 respectively.

Table 5.3 shows what kind of significance three different disturbance frequencies, 440, 110 and 55 Hz, have on the detection performance. The length the disturbances are fixed to 1.5 s and the SDRs are fixed to 1.00.

	Frequency [Hz]	$\mu_{Precision}$	$\sigma_{Precision}$	μ_{Recall}	σ_{Recall}	μ_{F1}	σ_{F1}
GMM	440	0.8698	0.1736	0.6050	0.2638	0.6591	0.1972
	110	0.7168	0.2650	0.2560	0.2666	0.2951	0.2201
	55	0.1098	0.1799	0.0453	0.1431	0.0380	0.0844
OCSVM	440	0.7332	0.1982	0.6733	0.2689	0.6361	0.1686
	110	0.6411	0.2085	0.4462	0.2938	0.4383	0.1987
	55	0.1094	0.1179	0.0842	0.1753	0.0714	0.1008

Table 5.3: Test results for disturbances of different frequencies using MFCC features. The duration and SDR of the disturbance are fixed at 1.5 s and 1.00 s respectively.

The results of the fourth experiment in Table 5.4 evaluate the performance of the six different feature sets mentioned in Section 4.2. The tested disturbances are of length 1.5 s, with an SDR of 1.00 and a frequency of 440 Hz. Each experiment is shown for both a GMM-based model and an OCSVM-based model.

	Feature set	$\mu_{Precision}$	$\sigma_{Precision}$	μ_{Recall}	σ_{Recall}	μ_{F1}	σ_{F1}
GMM	Time	0.2215	0.3283	0.0209	0.0403	0.0351	0.0614
	Time+Δ	0.0316	0.1270	0.0034	0.0099	0.0053	0.0159
	Spectral	0.1344	0.2792	0.0711	0.1930	0.0579	0.1228
	Spectral+Δ	0.0883	0.2288	0.0385	0.1473	0.0306	0.0881
	MFCC	0.8698	0.1736	0.6050	0.2638	0.6591	0.1972
	MFCC+Δ	0.7101	0.2405	0.2087	0.1953	0.2764	0.1924
OCSVM	Time	0.8234	0.2173	0.6827	0.1548	0.7283	0.1623
	Time+Δ	0.6363	0.2489	0.3483	0.1667	0.4256	0.1748
	Spectral	0.2977	0.2792	0.1702	0.2635	0.1517	0.1725
	Spectral+Δ	0.1633	0.1626	0.0992	0.1957	0.0885	0.1092
	MFCC	0.7332	0.1982	0.6733	0.2689	0.6361	0.1686
	MFCC+Δ	0.5624	0.1376	0.6190	0.2661	0.5433	0.1487

Table 5.4: Test results comparing the performance of the six different feature sets. The disturbance has a duration of 1.5 s, a frequency of 440 Hz, and an SDR value of 1.00 for all six measurements.

5.2.2 Training & inference speed

The result of measuring the RTF for ten minutes resulted in the maximum and mean RTF is shown in Table 5.5. The measurements are performed on an application running a GMM, as well as an algorithm running an OCSVM model, both using the MFCC feature set.

$Model$	$Uptime$ (s)	max_{RTF}	μ_{RTF}
GMM	600.64	0.29583	0.04618
OCSVM	601.59	0.41050	0.04873

Table 5.5: RTF measured running the real-time application. The RTF is recorded both using a GMM and an OCSVM, both using the MFCC feature set. The RTF was measured during an approximately ten-minute long run of each model.

Both models, the GMM and OCSVM were compared in terms of training time (TT) (Including extracting MFCC features), both for 60 and 180 seconds of training data (TDD), Table 5.6 shows the resulting maximum and mean TT, as well as the ratio between mean TT and TDD.

$Model$	TDD (s)	max_{TT} (s)	μ_{TT} (s)	$\frac{\mu_{TT}}{TDD}$
GMM	60	7.47	6.07	0.10
OCSVM	60	0.72	0.70	0.01
GMM	180	12.74	12.27	0.07
OCSVM	180	2.10	2.07	0.01

Table 5.6: Training time (TT) (including MFCC feature extraction) for GMM and OCSVM using 60 and 180 seconds of training data (TDD). Each model was trained ten times for each duration of training data.

Chapter 6

Real-time anomaly detection application

Based on the results from the evaluation chapter, an example of a real-time anomaly detection application was created. The application leverages a pre-trained model, either a GMM or OCSVM to compare an incoming segment of sound to the model by computing its anomaly score, classifying the segment as an anomaly if the anomaly score if it exceeds the threshold. The application runs on a headless Raspberry Pi 4 and also sends anomaly score, threshold, and STFT data to a client able to visualize it for monitoring. If the application is able to process a chunk before the next one arrives real-time anomaly detection is achieved.

The application consists of three Python programs.

- The first program records sound sampled at 16000 Hz, the duration of the recording is chosen by the operator. This program is when creating a training dataset using only normal operating sound, such as in Section 3.2.
- The second program trains either a GMM or OCSVM model on the previously recorded training data. The trained model is saved for use in the next step. Additionally, the threshold is computed from the training data.
- The third program is the actual real-time anomaly detection application. The application starts by loading the pre-trained model, it then starts a loop alternating between listening to 0.256 s long segments of sound, classifying them as anomalies or not, and then sending the data to a receiver for monitoring.

The hardware setup used for the application is shown in Figure 3.2, it is used both to record training data and for listening in real-time when running the application. The training data is recorded as described in Section 3.2. When running the application and monitoring the incoming sound, the application listens to 0.256 s long segments sampled at 16000 Hz.

Figure 6.1 shows a snapshot from a run of the real-time application trained on 180 seconds of fan sound. It consists of three graphs showing the anomaly score for the GMM, the anomaly score

for the OCSVM, and a spectrogram respectively. All three graphs depict the same time segment. The snapshot contains a few different disturbances composed from a desk situated roughly two meters away from the experiment station in Figure 3.2. From left to right in the spectrogram in Figure 6.1 c), at 10.24 seconds there is a short whistling sound, the anomaly score indicates that this is anomalous as it sharply increases for the duration of the whistle as seen in both Figures 6.1 a-b). The point of time, 8.192 seconds is surrounded by two disturbances, each 1 second in duration, with frequencies 6000, and 4000 Hz respectively. These disturbances cannot be distinguished from either of the plotted anomaly scores in Figures 6.1 a-b). Just to the right of 6.144 seconds, another disturbance caused by a loud bang is shown, at the corresponding time points in Figures 6.1 a-b) this is shown as a large spike in anomaly score. Just to the left of 4.096 seconds show another disturbance, this is the sound of pressing the button to turn off the fan on the power supply, the anomaly scores register a large peak, and as the fan winds down the anomaly score begin to trend up over the threshold.

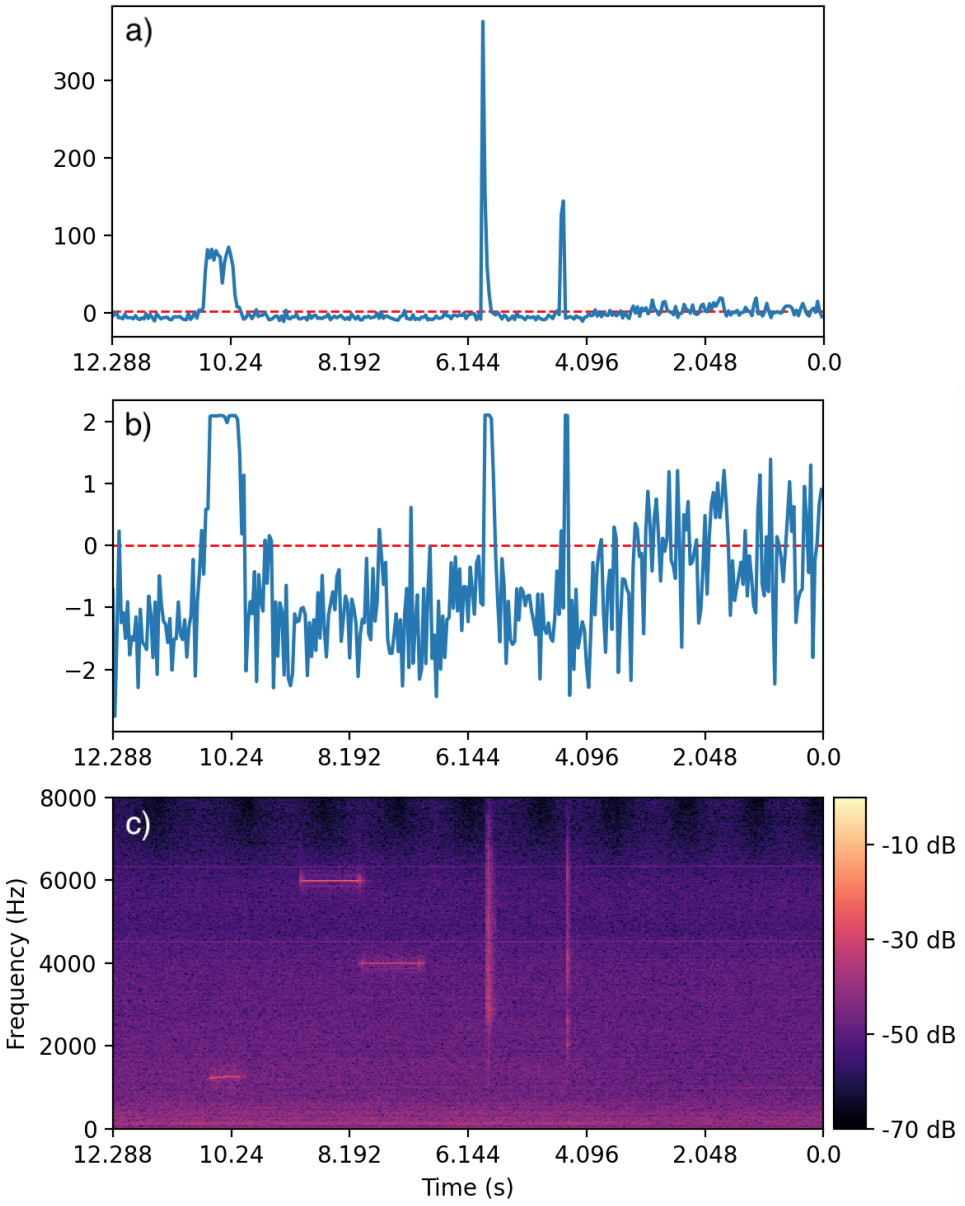


Figure 6.1: A snapshot of a real-time application run with models that were trained on 180 seconds of data. Plot a) shows the anomaly score in blue and the anomaly threshold for the GMM model. Plot b) shows the score and the threshold for the OCSVM model. Plot c) shows the live spectrogram of the sound.

Figure 6.2 depict the corresponding disturbances, but this time the applications are trained on 60 seconds of fan sound. Again a whistle is followed by a 6000 and a 4000 Hz tone, a loud bang, and turning off the fan. Similarly to the case in Figure 6.1, the whistle, the bang, and pressing the button to turn off the fan register large peaks in anomaly score for both models. The 6000

and 4000 Hz tones are barely noticeable in the anomaly score plots, and after turning off the fan the anomaly scores start trending up and oscillating around the threshold.

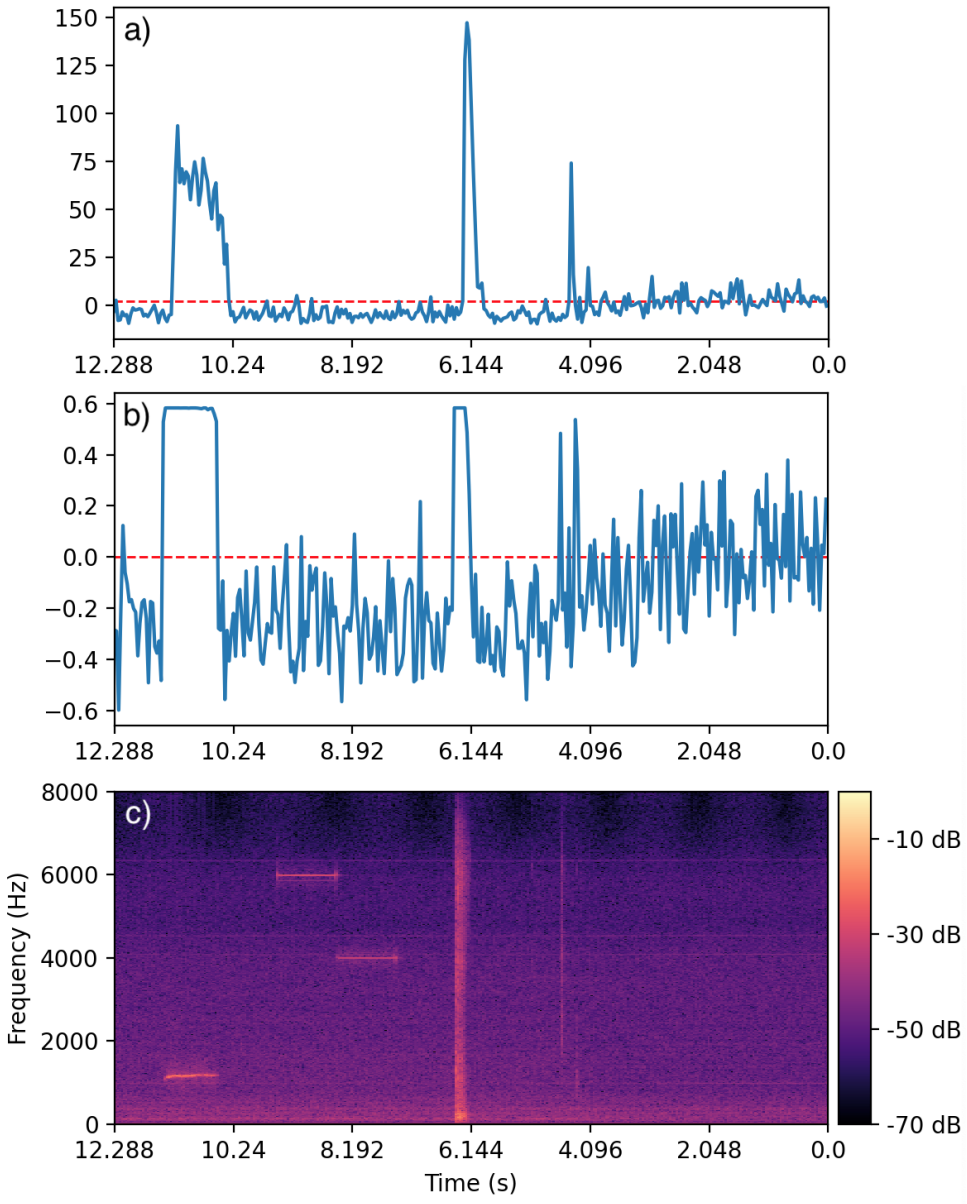


Figure 6.2: A snapshot of a real-time application run with models that were trained on 60 seconds of data. Plot a) shows the anomaly score in blue and the anomaly threshold for the GMM model. Plot b) shows the score and the threshold for the OCSVM model. Plot c) shows the live spectrogram of the sound.

Chapter 7

Discussion

This chapter will start out by interpreting the results of the evaluation section, it then goes on to discuss the findings from testing the real-time anomaly detection application introduced in Chapter 6.

7.1 Anomaly detection performance

The results presented in Table 5.1 clearly show that it is much easier to detect an anomaly for lower SDR values. This is unsurprising since a low SDR indicates that the segment contains a strong disturbance in relation to the signal.

Since both models perform very well, detecting all anomalies, with high precision, for very low values of SDR such as 0.01, it is interesting to evaluate how far we can push the SDR before the performance breaks down. The mean recall rate when going from an SDR of 0.50 to 1.00 drops significantly for both models. From 0.8698 to 0.6050 for the GMM model and from 0.9046 to 0.6733 for the OCSVM counterpart as shown in Table 5.1. The interpretation of this result is that an SDR of around 1.00 represents a breaking point and is thus extra interesting. Further experiments were therefore performed using 1.00 SDR.

According to Table 5.2, the length of the signal does not seem to have any significant effect on the ability to recall anomalies, although it does affect the precision. The lower precision for the shorter disturbances implies that shorter disturbances decrease the maximum possible number of true positives, meaning that frames that are falsely marked as anomalies, or false positives now make up a larger portion of the denominator in (2.34) lowering the precision. The drop in precision for shorter disturbances is most visible for the OCSVM model, likely due to having a bigger number of false positives, to begin with. One way to counteract the number of false positives would be to decrease the ν parameter when training the OCSVM.

An important aspect to keep in mind is that the position of a disturbance affects how many frames it is present in, and thus how many frames should be classified as an anomaly. The example in Figure 7.1 shows two anomalies of equal duration, but positioned differently and

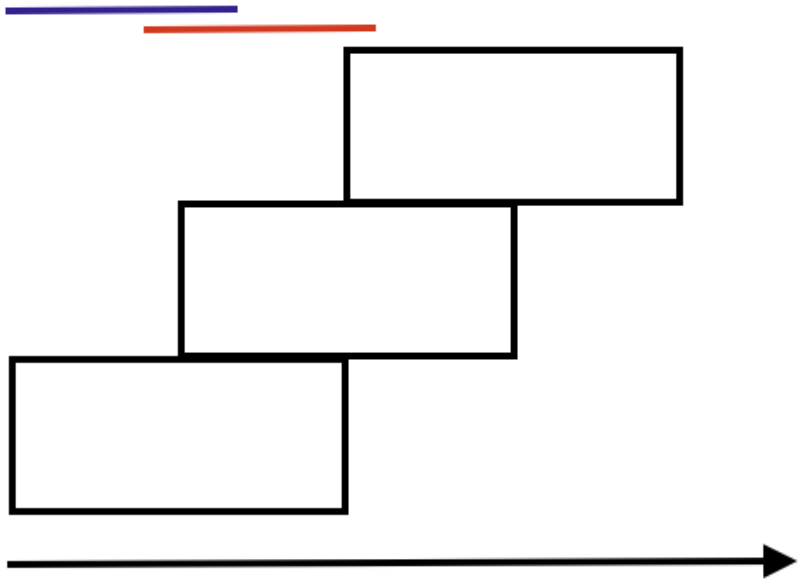


Figure 7.1: Shows two disturbances of equal duration, represented by the purple and the red lines. Underneath the lines, three frames of equal length and 50% overlap are depicted. As the image shows, the position of a disturbance affects the number of frames it covers. The purple line only covers two frames, while the red line cover three frames.

thus affecting a different amount of frames. The purple anomaly covers two frames, while the red anomaly covers three frames, thus each of the frames affected by the purple anomaly contains a higher concentration of anomalous samples than its red counterpart. This implies that it will be easier to detect the purple anomaly in comparison to the red one. In addition, when comparing classification metrics such as recall the result will be affected by the amount of frames an anomaly covers. If it was possible to recall the purple anomaly in the two frames the recall rate would be $\frac{2}{2} = 1$, while detecting the red anomaly, also in two frames would mean a recall rate of $\frac{2}{3}$. In all of the experiments the disturbance starts at 1.5 seconds, corresponding to the 960-th sample in the 46-th frame.

Table 5.3 presents the results of the experiments varying the frequency of the disturbance. When decreasing the frequency of the disturbance from 440 Hz to 110 Hz the precision deteriorates slightly for both models, while the recall rate is more than halved for the GMM model and reduced by 1/3 for the OCSVM model. When decreasing the frequency from 110 Hz to 55 Hz the precision and recall ends up at 0.1098 and 0.0453 respectively for the GMM model and at 0.1094 and 0.0842 respectively for the OCSVM.

The reason for this decline is that the effect of a disturbance with a certain frequency is dependent

on the frequency content distribution of the background signal. The intensity of a signal is not necessarily uniformly distributed. A disturbance with a certain SDR, duration, and frequency might be very easy to detect, while another disturbance with the same parameters except for a different frequency might be very hard to detect. This is because the SDR measurement does not take frequency distribution, or frequencies at all into account. A disturbance in a frequency band where the signal has a higher intensity will need to have a larger magnitude in order to stand out. While another disturbance of equal disturbance affecting the same signal located in a frequency band where the signal has lower intensity will stand out more. Figure 3.1 d) depicts a spectrogram containing a 440 Hz disturbance. The spectrogram contains a band with high-frequency content right below the disturbance, thus it would be harder to detect disturbances of 110 or 55 Hz compared to a disturbance of 440 Hz with the same SDR.

Another part of the anomaly detection performance evaluation was to compare how different feature sets compare in terms of precision, recall, and F1-score. Table 5.4 presents the results of the six different feature sets when the SDR is set to 1.00, the disturbance duration is 1.5 s and the disturbance frequency is 440 Hz. For the GMM MFCC is the clear winner in terms of mean precision, recall, and F1-score. Surprisingly, for the OCSVM the Time feature set is the best performer, the recall rate is similar to the MFCC feature set but the mean precision is much better at 0.8234 for the Time feature set vs 0.7332 for MFCCs. Though it is important to note that this is only measured at a value of SDR where the MFCC feature set is known to break down. It is possible that the Time feature set will quickly deteriorate and perform worse than MFCCs for other values of SDR.

It was expected that the MFCCs would perform best for both models, the reasoning being that they capture the spectral distribution of the sound in a way that neither the time nor spectral domain features do. Time domain features often look at the energy of a signal, but they do not know how that energy is distributed over the frequencies present in the signal. The spectral features do provide some information about where in the spectrum energy is located, but they are aggregated metrics such as spectral centroid and spectral spread. While this should in theory help, it may be the case that aggregating the spectral distribution into metrics such as spectral centroid and spread discards important characteristics of the sound.

In the comparison of the different feature sets, as expected MFCCs performed very well for both models. For the OCSVM the Time feature set surprisingly performed even better. Neither of the Δ-feature sets improved upon their non-Δ counterparts, on the contrary, the Δ-feature sets performed significantly worse.

7.2 Training & inference speed

The prerequisites for real-time anomaly detection is evaluated, both by measuring the RTF, which is presented in Table 5.5 and the model training time as shown in Table 5.6. As seen in the first row of Table 5.5 the RTF for a GMM is well below one. During a ten-minute run, the maximum RTF is 0.29583 and the mean RTF is 0.04618. While it is an absolute requirement that the RTF stays under one in order to enable real-time detection, in a real-world application the target device might run multiple other services, it is therefore desirable that our application

leaves headroom so that it can continue to perform real-time detection under heavier load.

The training time varied greatly between the two models, the GMM was much slower than the OCSVM, requiring on average 7-10 times longer time to complete feature extraction, training and threshold computation. The training time is important as it may be necessary to retrain the models if the normal behavior changes. It is possible that a machine that had a worn down part will sound different after replacing it with a new one, which would require retraining.

7.3 Real-time anomaly detection application

The real-time anomaly detection application was tested as seen in Figures 6.1 and 6.2, comparing training the GMM and OCSVM models on both 180 and 60 seconds of fan sound, represented by MFCC features. As the results showed, some of the disturbances registered large anomaly scores, over the threshold, and some did not. The disturbances caused by a whistling sound, a loud bang, and turning off the fan all registered large enough anomaly scores to be identified as anomalies. But the disturbances represented by a 6000 and a 4000 Hz tone do not clearly show an increase in anomaly score during the whole duration of either of the disturbances. This is likely due to the fact that they both only contain higher frequencies, as seen in Figure 4.2, when subsequent filters in the MFCC computation get wider, their weight (height) decay in order for the equal area constraint of the filters to hold. This means that they would need a larger magnitude in order to stand out compared to a lower frequency disturbance.

Chapter 8

Conclusion

This section will wrap up the thesis by presenting a conclusion of the results and the discussion in relation to the problem statement and research questions. Furthermore, a subsection on future work will propose a few possible next steps for anyone that wishes to continue where this thesis left off.

The model using a GMM performed best in terms of anomaly detection performance using the MFCC feature set, with the OCSVM being very close but slightly worse. When decreasing the duration of an added disturbance the recall of anomalous frames tapered off slowly, while the performance degraded significantly for shorter and shorter disturbances. The GMM performed better in terms of mean precision, while the OCSVM performed better in terms of recall, the better precision of the GMM outweighs the worse recall when it comes to F1-score where the GMM is the best. When varying the frequency of a disturbance the GMM wins in precision again, but this time it is not enough to outweigh the worse recall, meaning that the OCSVM wins in terms of F1-score. The low-frequency disturbances are harder to predict due to the fact that they line up with the frequency band of the signal. In an application, it is therefore important to keep the characteristics of the signal and anomalies in mind. When it comes to speed both models perform similarly in terms of mean RTF. Neither model exceeded a maximum RTF of one during the ten-minute measurement. The OCSVM is significantly quicker in feature extraction and training compared to the GMM.

When it comes to the different feature sets, as expected MFCC features performed well for both models. Surprisingly, for the OCSVM the Time feature set performed even better than the MFCCs. Since MFCC performs very well for both models, tested with different types of anomalies it seems that it is the most suitable feature set for now. Although it would be interesting to examine the Time features more thoroughly for the OCSVM and see if the performance generalizes for anomalies of varying SDR, duration and frequency.

To conclude, both the GMM and OCSVM manage to detect anomalies, the degree to which is dependent on SDR, duration, and the frequency distribution of the anomaly. Larger disturbances are unsurprisingly easier to detect. The duration of the disturbance does not seem to have a

noticeable effect on its detectability, but it does affect precision. The impact of the frequency of a disturbance depends on the characteristics of the signal that is affected by the disturbance. Disturbances that lie near the signal in frequency need to be stronger in order to stand out. It also seems like - at least for MFCC features - that disturbances with all of its frequency content above 4000 Hz are hard to detect due to the characteristics of MFCCs.

Under the tested conditions, the GMM seems to perform slightly better in terms of anomaly detection performance. Both models perform similarly in terms of mean RTF, but the GMM has a lower maximum RTF. But when it comes to training time the OCSVM is significantly faster. The answer to which model is more suitable is that it seems like the GMM is favored under the evaluated conditions.

While both models are able to perform real-time anomaly detection in the testing environment on a Raspberry Pi 4 it is beneficial to have headroom in terms of speed. A real-world application might be deployed on a device that has other services running simultaneously, allowing less of the performance to be allocated to the anomaly detection algorithm, or it might be the case that the algorithm is deployed on devices with even less performance than the Raspberry Pi 4.

8.1 Future work

It is possible that expected changes to a machine or changes to its environment result in sound that the model will claim as anomalous when in reality it is not. This concept is known as domain shift, a change in the characteristics of the sound which is not caused by an anomaly. Domain shift could be caused by a machine part slowly degrading over time or drastic changes to the background noise of the environment in which the machine operates, while different from the training conditions, still considered normal [3].

Models which are only trained once for a limited amount of time are susceptible to performance degradation caused by domain shifts not captured in training. A logical next step would therefore be to incorporate adaptability into the models so that the models can be retrained when new data reflecting changes in the environment arrives. An idea could be that every instance classified as an anomaly will be logged and machine operators would then be able to look at each anomaly and say if it really is an anomaly, or if it is a false positive. The false positives would then be incorporated into the normal training data of an updated model. In a scenario where models often are retrained on the fly, it would also be important to evaluate how much training data is required to retrain and how long time is required to retrain a model.

This thesis has focused on emulating the abilities of human hearing. Though, it is likely that our models miss valuable information outside of the human hearing range and in sounds not conforming to the characteristics of human hearing. Another path forward would be to investigate features that are not meant to mimic the attributes of our hearing, unlike the MFCCs, on sounds sampled with a frequency greater than 16 kHz.

Chapter 9

Thank you

A big thank you to our supervisors, for your guidance and valuable feedback during the course of our thesis project!

- Anders Sandberg, supervisor
- Rickard Jönsson, supervisor
- Maria Sandsten, principal supervisor

Bibliography

[1] H. Uematsu, Y. Koizumi, S. Saito, A. Nakagawa, and N. Harada, "Anomaly detection technique in sound to detect faulty equipment," *NTT Technical Review*, vol. 15, 08 2017.

[2] Z. Mnasri, S. Rovetta, and F. Masulli, "Anomalous sound event detection: A survey of machine learning based methods and applications." *Multimedia Tools and Applications: An International Journal*, pp. 1 – 50, 2021.

[3] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, November 2020, pp. 81–85. [Online]. Available: http://dcase.community/documents/workshop2020/proceedings/DCASE2020Workshop_Koizumi_3.pdf

[4] S. Ntalampiras, I. Potamitis, and N. Fakotakis, "Probabilistic novelty detection for acoustic surveillance under real-world conditions," *IEEE Transactions on Multimedia*, vol. 13, no. 4, pp. 713–719, 2011.

[5] Cloudera Fast Forward Labs. (2020) Deep learning for anomaly detection. [Online]. Available: https://ff12.fastforwardlabs.com/ff12-deep-learning-for-anomaly-detection.pdf

[6] A. Ang. (2020, December) Discrete short time fourier transform. [Online]. Available: https://angms.science/doc/SP/SP_STFT.pdf

[7] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[8] C. Cortes and V. Vapnik, "Support-vector networks." *Machine Learning*, vol. 20, no. 3, pp. 273 – 297, 1995.

[9] Wikimedia Commons. (2011) Kernel machine. [Online]. Available: https://commons.wikimedia.org/wiki/File:Kernel_Machine.svg

[10] Scikit-learn Developers. (2023) Kernel functions. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#kernel-functions

[11] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 07 2001. [Online]. Available: https://doi.org/10.1162/089976601750264965

[12] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Dcase 2020 challenge task 2 development dataset," Mar. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3678171

[13] NumPy Developers. (2022) numpy.hamming. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.hamming.html

[14] T. Giannakopoulos, "pyaudioanalysis: An open-source python library for audio signal analysis," *PloS one*, vol. 10, no. 12, 2015.

[15] M. Slaney, *Auditory Toolbox – Version 2*, 01 1998. [Online]. Available: https://engineering.purdue.edu/~malcolm/interval/1998-010/

[16] T. Ganchev, N. Fakotakis, and K. George, "Comparative evaluation of various mfcc implementations on the speaker verification task," *Proceedings of the SPECOM*, vol. 1, 01 2005.