Magnus Stave Lysø
Håkon Skjetne Kvalnes

Comparison of Wavelet Transforms and STFTs in Classification of Outdoor Noise

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Tor Andre Myrvoll
June 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Magnus Stave Lysø
Håkon Skjetne Kvalnes

Comparison of Wavelet Transforms and STFTs in Classification of Outdoor Noise

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Tor Andre Myrvoll
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Abstract

This thesis presents a comparison between two time-frequency transforms by using them in a classification problem with a convolutional neural network(CNN). The transforms in question are the wavelet transform and the well known and popular Short-time Fourier transform(STFT). The data used to conduct the comparison is a collection of eleven different sounds, or noises, traditionally found on construction sites. To be able to do a comparison, the two-dimensional arrays known as spectograms and scaleograms are extracted from the transforms as features before a CNN is trained with them. The resulting metrics, associated with classification problems, are then compared for the different features using a test set of the data. This will indicate if one of the transforms outperform the other, based on different configuration parameters for each transforms. The results indicates that neither of the two transforms can outperform the other overall, but there is still interesting results and differences found by comparing different classification metrics isolated. Both transforms have several parameters that can be tuned, and the results are heavily dependant on choosing the optimal configurations for the feature extraction. The results indicate that the two transforms both have a favourable resolution trade-off in time and frequency for different classes. The choice of transform can therefore be argued to be dependent on the data itself and not the notion that one, in general, is superior to the other.

Sammendrag

Denne masteroppgaven presenterer en sammenligning mellom to tid-frekvens transformasjoner ved å bruke dem i et klassifikasjonsproblem med et konvolusjonalt neuralt nettverk (CNN). De aktuelle transformasjonene er wavelet transformasjonen og den godt kjente og populære Short-time Fourier transformasjonen (STFT). Datasettet som brukes for å gjøre denne sammenligningen består av elleve forskjellige lyder, eller støy, som vanligvis finnes på byggeplasser. For å gjennomføre sammenligningen ekstraheres de to-dimensjonale matrisene, kjent som spektrogram og scaleograms, ved å bruke transformasjonene før et CNN blir trent opp med dem. Så blir forskjellige beregninger assosiert med dataklassifikasjon kalkulert med et testsett. Det er disse beregningene som til slutt sammenlignes. Sammenligning av resultater, basert på forskjellige konfigurasjoner av hver transformasjon vil gi en indikasjon på om en av transformasjonene kan utkonkurrere den andre. Resultatene indikerer at ingen av transformasjonene utkonkurrerer den andre direkte, men at det fortsatt er interessante resultater og forskjeller som kan observeres ved å se på isolerte beregninger. Begge transformasjonene har flere parametere som kan endres, og resultatene blir på grunn av dette helt avhengig av å velge passende konfigurasjoner for ekstraksjonene. Resultatene indikerer også at begge transformasjonene har gode oppløsningsutbytter mellom tid og frekvens for forskjellige klasser. Valget av korrekt transformasjon kan derfor argumenteres for å være høyst avhengig av hvilket datasett som brukes og at ingen transformasjon kan sies å være generelt bedre enn den andre.

ACKNOWLEDGMENTS

We would like to express our greatest gratitude towards our supervisors Tor Andre Myrvoll, Herold Olsen and Izzie Yi Liu for guiding us through the process of making this thesis. A special thanks to Herold for his patience and weekly follow-ups.

We would like to thank NTNU and the faculty of Information Technology and Electrical Engineering for five wonderful years and for providing us with the necessary aids and knowledge through our time as students. A thanks also goes to SINTEF Digital and Norsonic for providing the dataset used for this thesis.

Even though this semester was affected by the 2020 pandemic lockdown, the brilliance of todays technology made the flow of information run smoothly. Thank you to August Erlend Gardarsson and Nikolai Wanderås for providing a safe work space through this crisis.

List of Figures

2.1 *Illustration of how time and frequency resolution differs in a time series (a), Fourier transform (b), STFT (c) and wavelet transform (d). This figure is merely a sketch and provides an overall understanding of the resolution in time and frequency.* . . . 5

2.2 *Comparison of a chirp signal in a spectrogram (a) with a 10-point FFT and a scaleogram (b) with 10 scales. The chirp signal ranges from ∼0 Hz to 11 kHz over 10 seconds.* . . . 6

2.3 *A level 3 decomposition using low-pass filters G_0 and high-pass filters H_0. This demonstrates a filter-bank implementation of DWT.* . . . 7

2.4 *Morlet mother wavelet of order 5 and 14.* . . . 8

2.5 *Gaussian derivative mother wavelet of order 3 and 8.* . . . 8

2.6 *Mexican hat mother wavelet.* . . . 8

2.7 *Example of the mother wavelet of Shannon and Meyer.* . . . 9

2.8 *Figure showing how the resolution cell area remain the same for different scales but with same mother wavelet (here morlet of order 5). There are two wavelets present with different scales, a_1 and a_2.* . . . 10

2.9 *The relation of bandwidth and center frequency in the frequency range 10 Hz to 9 kHz for three different wavelets. (a), (b), (c) and (d) contain frequency responses from different scaled wavelets, while (e) compares the bandwidths of (a), (b), (c) and (d)* . . . 12

2.10 *Simple illustration of an ANN with four neurons as input layer, two hidden layers with five neurons and an output layer of three neurons. The layers are all connected* 13

2.11 *The figure shows an input of 5x5 values convoluted with a kernel of 3x3. The kernel is moved over the input by shifting one element at a time.* . . . 16

2.12 *Illustration showing how a larger array can be minimized by utilizing max pooling to get only the most distinct features* . . . 16

2.13 *Simple confusion matrix showing how outputs of a classification algorithm can be organized to find useful statistics.* . . . 18

2.14 *Confusion matrix including precision and recall metrics.* . . . 19

2.15 *Figure showing curves formed by precision and recall values at different probabilistic classification thresholds for a trained and a random binary classifier. The areas under the curves (AUC) are also given. It is important to note that a badly trained classifier can perform worse than the random one as well.* . . . 20

3.1 *Image showing the average frequency spectrum of each class and the deviation from this average in each class.* 23
3.2 *The process of merging two scaleograms created with different sampling frequency* . 27
3.3 *Comparison between STFT (a), CWT with morlet (5) (b) and CWT with morlet (5) and two sampling frequencies (c).* 28
3.4 *Display of how the truncation factor affects the resulting scaleogram picture. These scaleograms were produced with morlet mother wavelet and 200 scales.* 29
3.5 *Display of how the amount of scales affect the resulting scaleogram picture. These scaleograms were produced with a morlet mother wavelet and no truncation.* 30
3.6 *An example of the scaleograms for mexican hat (a), gaussian derivative of order 5 (b) and morlet of order 5 (c). The mother wavelet is shown on the left side.* 32
3.7 *An example of the scaleograms for morlet of order 11 (a), 14 (b) and 17 (c). The mother wavelet is shown on the left side.* 33
3.8 *Visualisation of the final network architecture. The number of neurons in is layer are not exact, except from in the output layer.* 34
3.9 *ReLU activation function.* 35

4.1 *Figure showing how three of the different scoring metrics where affected, for each mother wavelet, by the two truncation factors in use.* 38
4.2 *Figure showing how three of the different scoring metrics where affected, for each mother wavelet, by the number of scales in use.* 39
4.3 *Figure showing how three of the different scoring metrics where affected, for each mother wavelet, by the number of scales in use.* 40
4.4 *Figure showing how four of the different scoring metrics where effected by the number of oscillations in the morlet. The original used mother wavelet is marked by a red circle.* 48

List of Tables

3.1 *Overview of classes present in the dataset, a description of them and the count of each.* 22
3.2 *Overview of different wavelet families for the PyWavelets library* 24
3.3 *The features for CWT and STFT that were used in the making of the results.* 31

4.1 *Table showing metrics for a CNN trained on different features obtained with different configurations of STFTs and CWTs. It should be mentioned that the time resolution for the CWTs is the highest attainable resolution based on the truncation factor.* 42
4.2 *Color chart for the worst (red) to the best (green) results.* 42
4.3 *Table showing accuracy for the different mexican hat wavelet features.* 43
4.4 *Table showing accuracy for the different gaussian(5) wavelet features.* 44
4.5 *Table showing accuracy for the different morlet wavelet features.* 45
4.6 *Table showing accuracy for some different STFT features.* 46
4.7 *Table showing accuracy for the rest of the STFT features.* 47
4.8 *Table showing the metric scores for increased oscillations in the morlet.* 49
4.9 *Table showing the accuracy score for increased oscillations in the morlet.* 50

5.1 *Table showing the total data size of all features extracted from the 2960 sound files.* 55

Abbreviations

ANN	Artificial Neural Networks
AUPRC	Area Under Precission Recall Curve
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CWT	Continous Wavelet Transform
DWT	Discrete Wavelet Transform
FN	False Negative
FP	False Positive
gaus	Gaussian derivative (wavelet)
gaus5	Gaussian derivative of order 5 (wavelet)
GPU	Graphics Processing Unit
mexh	Mexican Hat (wavelet)
morl	Morlet of order 5 (wavelet)
MRA	Multi Resolution Analysis
QMF	Quadrature Mirror Filters
STFT	Short-Time Fourier Transform
SWT	Stationary Wavelet Transform
TN	True Negative
TP	True Positive

Contents

Abstract i

Sammendrag ii

Acknowledgements iii

List of Figures iv

List of Tables vi

Abbreviations vii

1 Introduction 1

2 Theory 3

1 Wavelets 3

1.1 Historical overview 3

1.2 Basic principles 4

1.3 Wavelet transform vs. STFT 5

1.4 Continuous wavelet transform (CWT) 6

1.5 Discrete wavelet transform (DWT) 7

1.6 Wavelet families 7

1.7 The trade-off between frequency and time resolution in CWT 9

1.8 Bandwidth and center frequency 11

2 Convolutional neural networks 13

2.1 Historical overview 13

2.2 Basic principles of CNNs 14

2.2.1 Convolutional layers 14

2.2.2 Pooling layers 16

2.2.3 Forward layers 17

2.3 Metrics for data classification 17

2.3.1 Confusion matrix 18

2.3.2 Accuracy 18

2.3.3 Precision and Recall 19

2.3.4 F1-score 20

2.3.5 Macro vs. Micro averaging 20

3 Experimental setup **21**

1 The Dataset 22

2 Assumptions and limitations 24

2.1 Data length 24

2.2 Wavelet transform 24

2.3 STFT 25

2.4 Frequency range 25

2.5 Distribution of frequency components 25

2.6 Truncation of time 25

2.7 The problem of false frequency components 26

2.8 Visual evaluation of time truncation 29

2.9 Visual evaluation of distribution of scales 30

2.10 The final features and data size 31

3 CNN 34

3.1 Creating the CNN 34

3.2 Finding useful statistics 35

4 Results **37**

1 F1-score and AUPRC 38

2 Accuracy score for mexican hat 43

3 Accuracy score for gaussian(5) 44

4 Accuracy score for morlet 45

5 Accuracy score for STFT 46

6 Increased oscillations for morlet wavelet 48

5 Discussion **51**

1 Evaluation metrics 51

2 CWT vs. STFT 52

3 Increasing wavelet oscillations 53

4 Distribution of scales 54

5 Thoughts on the low frequency issue in scaleograms 54

6 Dataset and extraction time 54

7 Further work 56

6 Conclusion **57**

Bibliography **59**

Appendix A **61**

Chapter 1

Introduction

Analysis in time and frequency are two fundamental building blocks in modern signal processing. Looking at these two properties together is known as time-frequency analysis and has its strengths for analysing signals with varying statistics over time e.g non-stationary signals. Since the first implementations of time-frequency analysis, researchers have been aware of the fundamental limit to the maximal obtainable resolution in both frequency and time simultaneously. This is known as the Gabor limit, or the uncertainty principle of signal processing [1]. The Gabor limit states, as Heisenberg's uncertainty principle states in quantum mechanics for position and momentum; that an exact frequency can not be found at an exact time. There will always be a trade-off between the frequency and time localisation, and the resolution obtainable in both domains. The de facto standard for time-frequency analysis can be said to have been the Short-time Fourier transform (STFT) in the later years. Other ways of time-frequency analysis have however been shown to have favorable properties in the way the trade-off between time and frequency resolution is shifted. One such method is the wavelet transform which have been shown on multiple occasions to have favorable properties for certain signals. Where the STFT is bound by its window size, giving a fixed resolution, either good in time or frequency, the wavelet transform in general gives a good time resolution for high frequency occurrences in the signal and a good frequency resolution for low frequency occurrences in the signal, a combination that may be suited for a lot of real-life signals such as in seismics or in brain activity[2].

The later years advancements in the field of machine learning have made the analysis of complex structures and huge amounts of data less time consuming, due to the ability to let complex machine learning algorithms work with the data. The popularity for such algorithms have seen an enormous surge through the 2010s and it is applied everywhere from the medical industry to in autonomous cars. In many cases the use of machine learning can enable pattern and structure analysis on a level and at a speed a human never could match. This pattern recognition ability is especially prominent in convolutional neural networks (CNNs) used widely and successfully in image recognition and for other applications.

The idea of this thesis is to investigate if the wavelet transform can outperform the STFT in a time-frequency analysis based classification of different construction site recordings using a CNN. The analysis is done by looking at the resulting spectograms, or scaleograms in the case of the wavelet transform. These two-dimensional arrays form pictures, which makes CNNs a natural choice to perform the classification due to their pattern recognition ability. The data to be used is a combination of eleven different classes with outdoor sound recordings of typical construction site related noise and sounds. The main hypothesis is to investigate if the wavelet transforms multiresolution trade-off in time and frequency may perform better in a classification problem than the STFT, for such data.

It is worth noting that this thesis expects the reader to be somewhat familiar with time-frequency analysis, general concepts of signal processing and artificial neural networks. The thesis will not go in mathematical depth of general neural networks and their different learning process, but there will be provided a short summary of this. A deeper description of the concept of CNNs and their specialities are however provided. The main focus will be on the wavelet transform compared to the more traditional STFT.

The thesis is structured to first give a theoretical overview of the wavelet transform, STFT and CNN. The chapter "Experimental setup" will contain the main ideas, concerns, limitations and assumptions regarding the task which then will be discussed and forged into an implementation. The results are created by comparing classification metrics from inputs in the form of spectrograms and scaleograms. A discussion section provides a more in-depth view on the results before the thesis comes to an end as the conclusions are drawn.

Chapter 2

Theory

This chapter contains theory needed to better understand the implementations and results presented in this thesis. It will give an introduction to wavelets, the different wavelet transforms and their properties. The focus will lie on the continuous wavelet transform (CWT), as this is the most important wavelet transform in time-frequency analysis. There will also be an explanation of the Short-time Fourier transform (STFT), convolutional neural networks (CNN) and different metrics for data classification such as accuracy, F1-score and AUPRC.

1 Wavelets

1.1 Historical overview

J. Morlet, a geophysical engineer, was in the late 1970s faced with a problem of analyzing signals which contained short bursts of high frequency components and longer time spans of low frequency components. STFT is able to analyze either high frequency components using narrow time windows (wideband frequency analysis), or low frequency components using wide time windows (narrowband frequency analysis). Morlet therefore came up with a new window function to analyze different frequency bands. He met a lot of criticism for his work, but continued on to formalize a mathematical transformation and its inverse. Little did he know, however, that the wavelet transform he had developed was merely a rediscovery of a slightly different interpretation of Alberto Calderón's work on harmonic analysis in 1964. The similarities in Calderón's and Morlet's work was discovered in 1984 by Yves Meyer, a French mathematician. Meyer continued working on developing better localization properties for the wavelet. In 1985 he constructed orthogonal wavelet basis functions with very good time and frequency localization[3].

In the mean time, Ingrid Daubechies developed the frames for discretization of time and frequency parameters for the wavelet transform. This allowed for more liberty in the choice of basis functions, but at the expense of some redundancy. Daubechies, along with Stéphane Mallat, is therefore credited with developing the transition from continuous to discrete signal analysis. In 1986 Mallat developed the idea of multiresolution analysis (MRA) for discrete wavelet transform (DWT) with Meyer. The idea was decomposing a discrete signal into frequency bands by a series of lowpass and highpass filters to compute its DWT from the approximations at these various frequency components. This idea had been familiar to electrical engineers for about twenty years under the name of quadrature mirror filters (QMF) and subband filtering, which were developed by A. Croisier, D. Esteban and C. Galand in 1976. Mallat's work constituted a natural extension of time localization to the well-established frequency localization idea of QMF and subband coding. Also in 1988, with the development of Daubechies' "*Orthonormal bases of compactly supported wavelets*" [4], the foundations of the modern wavelet theory were laid [3].

1.2 Basic principles

The families of function $\Psi_{a,b}$,

$$\Psi_{a,b}(x) = |a|^{-1/2}\Psi(\frac{x-b}{a}) \tag{2.1}$$

generated from a single function Ψ by scaling a, tranlation b and sample x is well stated by Ingrid Daubechies; "*... we shall call such families 'wavelets'.* " [4].

$\Psi_{a,b}(x)$ is what can be recognized as the wavelet itself. Ψ is called the mother wavelet and is what decides the general shape of the wavelet. a is the scaling factor which will alter the width of the wavelet in both time and frequency. A smaller scale correlates to a more compressed wavelet and vice verca. Using lower scales will capture more rapid oscillations (higher frequencies), while higher scales will capture more slowly varying oscillations (lower frequencies). b, translation, is a parameter that shifts the wavelet along the samples x. This can also be described as a convolution. The wavelet transform can be found by,

$$< f, \Psi_{a,b} >= \int_{-\infty}^{\infty} |a|^{-1/2}\Psi(\frac{x-b}{a})g(x)dx \tag{2.2}$$

where $g(x)$ is the target signal of the transform. By varying scale and translation, a set of coefficients $\Psi_{a,b}$ and frequencies f corresponding to the scales are extracted. These coefficients can be used to create an image of the signal $g(x)$ which is called a scaleogram.

In general, there are three criteria a wavelet must satisfy. It has to have:

- Zero mean
- Finite energy
- A non-zero frequency component

[5]

1.3 Wavelet transform vs. STFT

Figure 2.1 demonstrates how the frequency and time resolution differs in a time series (a), Fourier transform (b), Short-Time Fourier Transform (c) and wavelet transform (d).

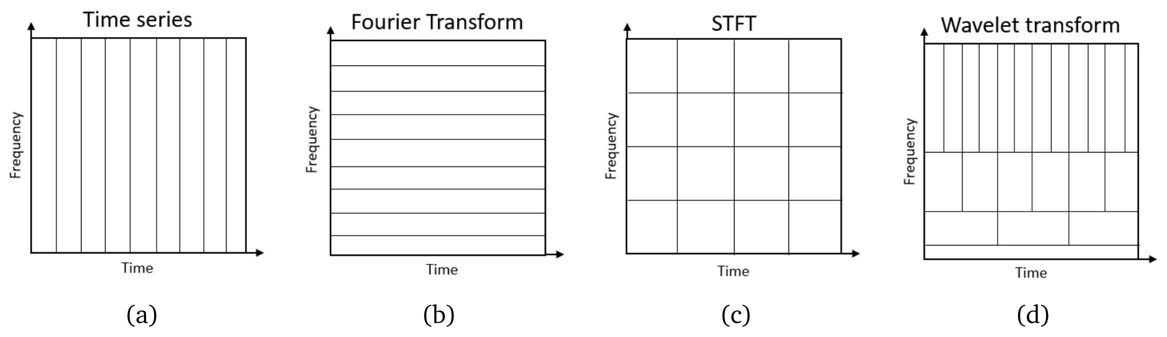


Figure 2.1: *Illustration of how time and frequency resolution differs in a time series (a), Fourier transform (b), STFT (c) and wavelet transform (d). This figure is merely a sketch and provides an overall understanding of the resolution in time and frequency.*

It is quite obvious that a time series offers excellent time resolution, but zero frequency resolution. To provide resolution in the frequency domain a Fourier transform is performed on a time series. It can now precisely be seen which frequencies the signal contains, but there is no knowledge of when they occurred in time. This is where the STFT comes to play. By computing several Short-Time Fourier Transforms throughout the time series, both frequency and time resolution are present to create what is called a spectrogram. The time which each Fourier transform is calculated over, is called the length of the window. A window function can also be applied so to weight the samples in the window differently. This has been shown to have favorable properties in the frequency domain. Having no window function is the same as having a rectangular window function. One of the most popular and most used window function can be argued to be the Hanning window function. By varying the length of the windows the trade-off between time and frequency can be decided. It is also common practise to have a overlap between each window to capture information that would else wise have fallen between two windows. To capture low-frequency signals within a spectrogram a long window is required. This causes the time resolution to suffer. The wavelet transform will allow for these low-frequency components to be captured while still having excellent time-resolution for high frequencies.

In figure 2.2 the concept of analysis with a continuous wavelet transform (CWT) is shown in comparison to the more traditional STFT implementation. In the figure, a logarithmic chirp is used for both the CWT and STFT where a 10-point frequency analysis is executed. For the CWT it can easily be seen that for high frequencies, the time resolution increases, while the frequency resolution gets worse. For the STFT, the resolution remains constant.

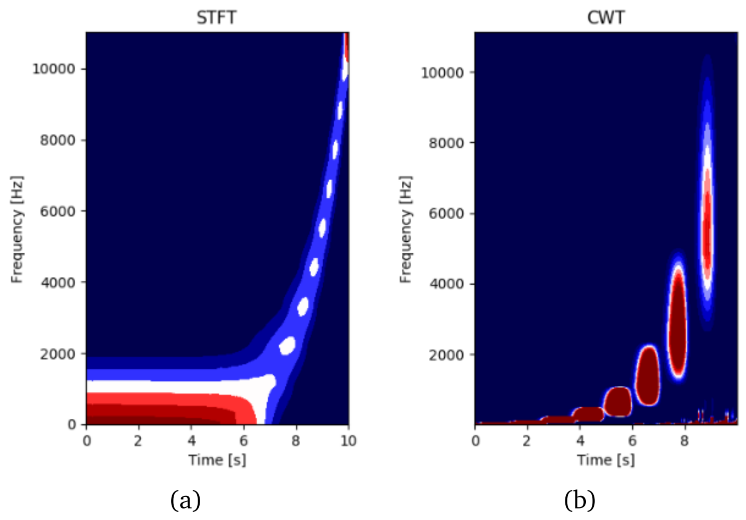


Figure 2.2: *Comparison of a chirp signal in a spectrogram (a) with a 10-point FFT and a scaleogram (b) with 10 scales. The chirp signal ranges from ∼0 Hz to 11 kHz over 10 seconds.*

1.4 Continuous wavelet transform (CWT)

Continuous wavelet transform is, as mentioned, the first developed wavelet transform with roots from as early as 1964 by Alberto Calderón. Although there has been some development since then, the main principles and application of the transform remains the same[3].

The CWT is used to construct a time-frequency representation of a signal that offer different qualities than for instance the STFT. The transform can be described as a convolution of the data sequence $g(x)$ with a scaled and translated version of the mother wavelet Ψ. The convolution can be computed directly or with an FFT-based fast convolution. One of the main advantages of using CWT in time-frequency analysis is the ability to have scales in a continuous sequence. This indicates that an exact scale can be found for an exact corresponding center frequency and the other way around.

The main application of the CWT is to perform time-frequency analysis. It is an excellent tool for mapping the changing properties of non-stationary signals and is ideal for determining whether or not a signal is stationary. Its good low-frequency resolution combined with its good time resolution for higher frequencies is desired in a lot of applications. The CWT is today employed in a broad span of disciplines; from astronomy to zoology. There has been released papers with CWT in fields like quantum mechanics, meat processing, river sedimentation and driver reaction monitoring to mention a few [5]. CWT has also been used for studies in geophysics like; tropical convection, the El Niño-southern Oscillation and atmospheric cold fronts etc. [6]. Another use for it is in medicine for MRS, and music to capture short bursts of repeating and alternating notes [7]. So, CWT is used in a lot of different fields, serving the same purpose; time-frequency analysis. It is reported to provide a "new view" compared to the more traditional STFT.

1.5 Discrete wavelet transform (DWT)

In the 1980's the discrete wavelet transform DWT was brought to light by Ingrid Daubechies and Stépḥane Mallat. This transform was limited by discretization of time and scale[3]. The scales are discretized in the power of 2 as follows;

$$a = 2^k, \qquad k = 1, 2, 3, 4... \tag{2.3}$$

while the time, corresponding to the translation are limited by

$$b = 2^k m, \qquad m = 1, 2, 3, 4... \tag{2.4}$$

These limitations lead to a loss in time,- and frequency resolution compared to CWT, but there will be a massive decrease in memory needed to store all the coefficients. For this reason, the DWT is very viable in denoising of signals and data compression like in JPEG2000, but will have little to no place in time-frequency analysis.

DWT can be implemented as a composition of high-pass filters, low-pass filters and decimations as illustrated in figure 2.3.

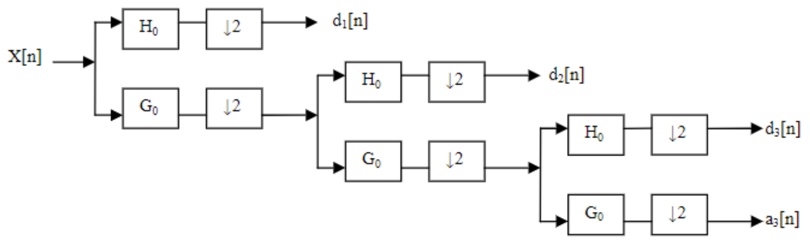


Figure 2.3: *A level 3 decomposition using low-pass filters G_0 and high-pass filters H_0. This demonstrates a filter-bank implementation of DWT.*

1.6 Wavelet families

There exist a wast number of wavelet families - both for continuous and discrete wavelets. As the main focus will lie on continuous wavelets, some of the most known continuous wavelet families will be presented [8].

The order, n, of the different wavelet families allows for a varying trade-off between time and frequency resolution. An increased order will correlate to a over-all higher frequency resolution but at the cost of a worse time resolution, and vice verca. The frequency range of interest and the requirements for resolution are factors for deciding the order and wavelet family.

A popular and well known wavelet is the morlet wavelet, also called Gabor wavelet. It is composed of a complex exponential multiplied by a gaussian window. The formula for the mother wavelet of a morlet is: $\Psi_n(x) = c_n \pi^{-\frac{1}{4}} e^{-\frac{x^2}{2}} (e^{inx} - e^{-\frac{1}{2}n^2})$ where $c_n = (1 + e^{-n^2 - 2e^{-\frac{3}{4}n^2}})^{-\frac{1}{2}}$. This wavelet has properties closely related to human perception, both in hearing and vision.

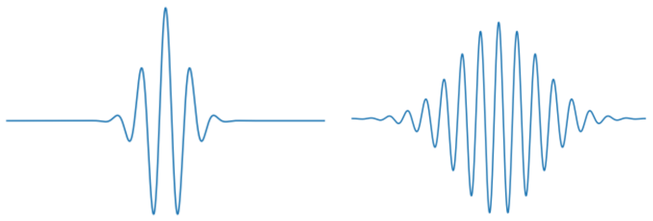


Figure 2.4: *Morlet mother wavelet of order 5 and 14.*

Figure 2.4 shows a morlet of order 5 and 14. One can clearly see the increase in oscillations when increasing the order of the wavelet.

The gaussian derivative wavelet is presented in figure 2.5. This family is based on the function $\Psi_n(x) = C_n e^{-x^2}$ which makes every odd order (n) be asymmetric, while even orders give symmetric implementations which resemble a morlet. Figure 2.5 shows a gaussian derivative wavelet of order 3 and 8. The similarities between the morlet and the gaussian can clearly be seen when comparing the morlet of order 5 and the gaussian of order 8.

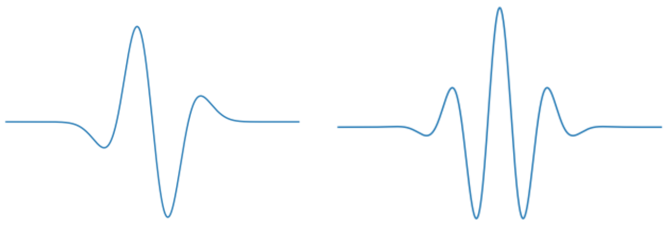


Figure 2.5: *Gaussian derivative mother wavelet of order 3 and 8.*

The mexican hat in figure 2.5 is the negative normalized second derivative of a gaussian function and is a special case of a family of continuous wavelets called Hermitian wavelets. It can be implemented by $\Psi(x) = \frac{2}{\sqrt{3}\sqrt[4]{\pi}} e^{-\frac{x^2}{2^2}}(1-x^2)$. This wavelet has gotten its name because of its similarity to a sombrero [9]. Also, the mexican hat does not have the option for changing its order, and thus limits its adaptive ability for time-frequency trade-off.

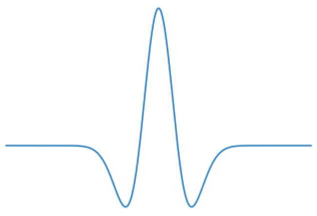
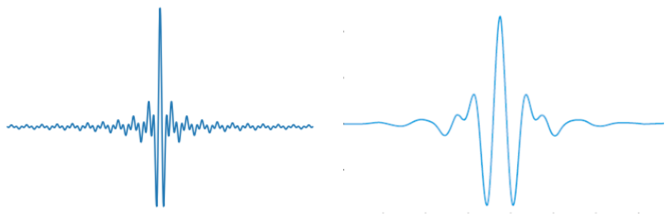


Figure 2.6: *Mexican hat mother wavelet.*

Two other popular wavelets are the Shannon and Meyer families. They can be seen in figure 2.7. Shannon and Meyer will not be included in this thesis.



(a) Shannon (b) Meyer

Figure 2.7: *Example of the mother wavelet of Shannon and Meyer.*

The wavelet families all have slightly different properties, which makes them able to detect different patterns in signals. They do however have in common that wavelets with large amounts of oscillations give a high resolution in frequency, while having to trade off resolution in time.

1.7 The trade-off between frequency and time resolution in CWT

As mentioned in the introduction there will always be a trade-off between the frequency and time resolution that can be obtained simultaneously. In signal processing this is often referred to as the Gabor limit, or Heisenberg-Gabor limit. This limit can be formulated as:

$$\sigma_t \cdot \sigma_\omega \geq \frac{1}{2} \tag{2.5}$$

or:

$$\sigma_t \cdot \sigma_f \geq \frac{1}{4\pi} \tag{2.6}$$

where σ_t and σ_f representes the standard deviation of the time and frequency estimates. The proof is omitted here, but can be found in "*Wavelet theory and its applications*" by R.K Young[1].

As the time and frequency resolution varies based on the scales, the resolution cell, or the combined time and frequency resolution remain the same. By replacing the x in the general case in equation 2.1 with time t, the resolutions in time and frequency can be written as[10]:

$$\sigma_t^2(a,b) = \int_{-\infty}^{+\infty} (t - u_{a,b})^2 |\Psi_{a,b}(t)|^2 dt \tag{2.7}$$

$$\sigma_\omega^2(a,b) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} (\omega - \xi_{a,b})^2 |\hat{\Psi}_{a,b}(\omega)|^2 d\omega \tag{2.8}$$

where:

$$u_{a,b} = \frac{1}{||\Psi_{a,b}||} \int_{-\infty}^{+\infty} t |\Psi_{a,b}(t)|^2 dt \tag{2.9}$$

$$\xi_{a,b} = \frac{1}{2\pi ||\Psi_{a,b}||} \int_{-\infty}^{+\infty} \omega |\hat{\Psi}_{a,b}(\omega)|^2 d\omega \tag{2.10}$$

and:

$$||\Psi_{a,b}||^2 = 1 \tag{2.11}$$

In the equations above, the $\hat{\psi}_{a,b}$ represent the Fourier transform of the wavelet. The resolution will change in time and frequency dependant on the mother wavelet, the scales and the translation, but the resolution cells area will remain the same[11]. This area of the resolution cell is again bound by the Gabor limit as seen in equation 2.5, and this concept is demonstrated in figure 2.8 below. It is important to note the the figure shows a morlet mother wavelet of order 5 in time, and its corresponding frequency response. This wavelet shape and frequency response is not a general case, but only representative for the case of the morlet wavelet.

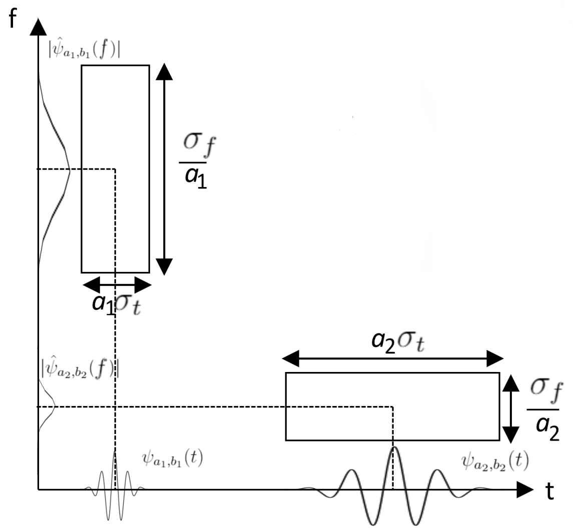


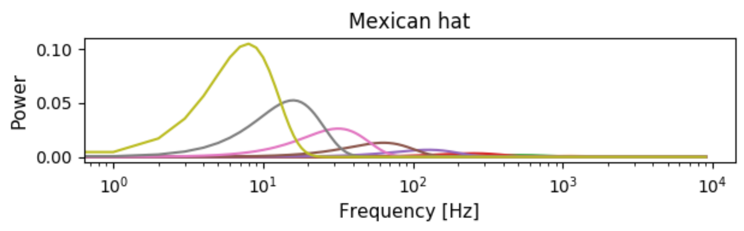
Figure 2.8: *Figure showing how the resolution cell area remain the same for different scales but with same mother wavelet (here morlet of order 5). There are two wavelets present with different scales, a_1 and a_2.*

Each mother wavelet will have a fixed resolution cell area that will trade-off between time and frequency as the scales change. The resolution cell area may however vary between the different mother wavelets.

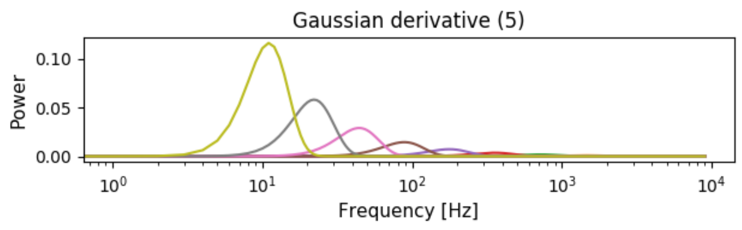
1.8 Bandwidth and center frequency

A principle behind wavelets is as mentioned; multi resolution analysis (MRA). MRA takes advantage of the fundamental trade-off between time and frequency in such a way that there is possible to achieve good time resolution for high frequency ranges and good frequency resolution for low frequency ranges. Figure 2.9 describes how this effect causes the bandwidth of different scaled wavelets to increase in relation to its center frequency. The different mother wavelets mexican hat, gaussian derivative and morlet have unique frequency responses that can be seen in figure 2.9 (a, b, c, d). The figures are made with 9 different scales. Increasing the amount of scales will lead to more overlap between the frequency responses of the different scaled wavelets. It is important that there is some overlap so to not miss out on any frequency components, but too much overlap will lead to smearing in the scaleograms. The frequency responses for the wavelets presented in figure 2.9 are all more or less symmetrical and formed like band pass filters, which is not the case for all wavelet families. The frequency along the x-axis is distributed in a logarithmic manner. This makes it look like the frequency responses for high scales contain most energy, but this is not the case - all curves have been normalized to cover an area of 1. It can be observed that the morlet of order 14 has the lowest bandwidth, while the mexican hat stands out by its somewhat asymmetrical frequency responses and higher bandwidth.

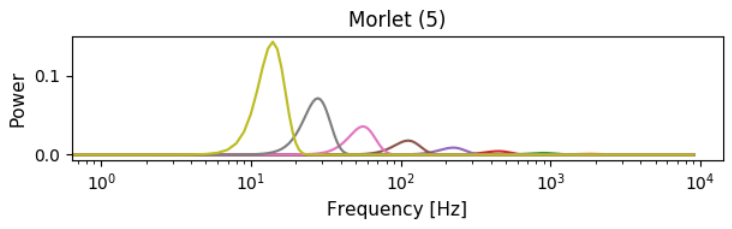
Figure 2.9e collects the center frequencies and bandwidths of the frequency responses for each wavelet type. It is recognized, especially for high frequencies, that the morlet of order 14 has the overall best frequency resolution followed by morlet of order 5, gaussian derivative of order 5 and lastly mexican hat. One would therefore expect scaleograms created with morlet to dominate for frequency resolution, while the mexican hat would contribute with a better time resolution.



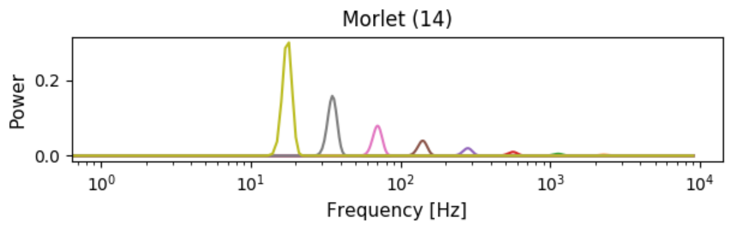
(a)



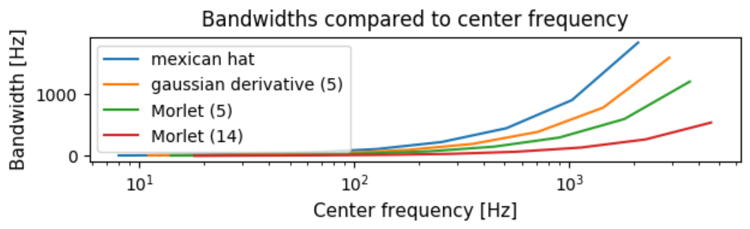
(b)



(c)



(d)



(e)

Figure 2.9: *The relation of bandwidth and center frequency in the frequency range 10 Hz to 9 kHz for three different wavelets. (a), (b), (c) and (d) contain frequency responses from different scaled wavelets, while (e) compares the bandwidths of (a), (b), (c) and (d)*

2 Convolutional neural networks

2.1 Historical overview

The idea of artificial neural networks, or ANNs, was born in the 1940's with the publication of "*A logical calculus of the ideas immanent in nervous activity*" by Warren McCulloch and Walter Pitts[12]. These early networks was vaguely inspired by biological brains and how they are built up by a network of simple units (neurons) that can learn and evolve from previous experience by changing their connections with each other. A simple network architecture can be seen in figure 2.10.

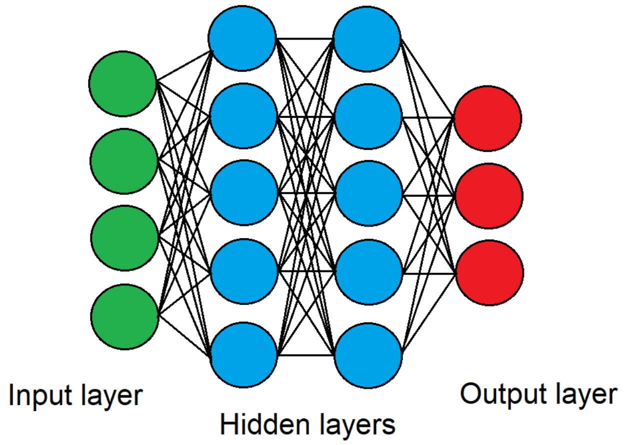


Figure 2.10: *Simple illustration of an ANN with four neurons as input layer, two hidden layers with five neurons and an output layer of three neurons. The layers are all connected*

The earliest ideas of convolutional neural networks, or CNNs, in the form they are used today came in the 1980s with the prepositions made by Kunihiko Fukushima. He proposed new types of layers in neural networks, one of them being the convolutional layer and the second being the downsampling layers that later became the well known pooling layer[13]. These prepositions were based on the discoveries of David Hubel and Torsten Wiesel who through the 1950s and 60s showed that the visual cortex of cats and monkeys contain special neurons that only correspond to a smaller part of the full field of view[14]. Several of these neurons were shown to create the full field of view and that they had a certain overlap between each other. Even though the principles were proposed, developed and implemented through the 1980s the popularity of CNNs did not accelerate before the 2000s. This was when it was shown that use of GPUs reduced the processing time of neural networks by a multitude compared to traditional CPU use. CNNs have historically been associated with image analysis, but in the later years their usefulness have been utilized in several other fields.

Artificial neural networks are regarded as a subcategory of the broader field of study that is Machine learning, and convolutional neural networks is again a subcategory of neural networks.

An exact modeling of a complete biological brain, such as a human brain, is still regarded as science fiction and lies years in the future. However different neural networks are today an important building block in the process of having a machine "learn" to perform narrow specific tasks without being directly pre-programmed to do so. The use of CNNs and machine learning in general have exploded in the last 10 years, and it is today applied in a broad spectrum of fields.

2.2 Basic principles of CNNs

As the name indicates, the mathematical operation of convolution is employed as a part of the process. This was one of the main ideas first presented by Kunihiko Fukushima. The data processing is done in a grid-like topology and can be applied to 1D data such as time-series as well as 2D images and even higher dimensions of data[15].

A typical CNN is built up of convolutional layers and pooling layers in combination with the typical structure of a regular feed-forward network[16]. The purpose of the two first layers is to compress the input data in such a way that only the most distinct features are fed into the forward network. This is also in principal how the visual cortex of mammals work as shown by Hubel and Wiesel[14]. Feeding a forward neural network directly with images or other data, containing a huge amount of details, is just not practical due the the potentially huge number of input neurons. By applying convolutional and pooling layer, a CNN can "perceive" the most distinct variations of the input data. This is done to make the inputs to the forward network as distinct, scaled down and usable as possible without having a number of input neurons that is "impossible" to implement.

2.2.1 Convolutional layers

The convolutional layers employs the convolutional operation. The operation takes two functions and produces a new function that describes how they correlate. In mathematical terms this is done by taking the integral of the product between the two, where one function is reversed and shifted. This can be written as:

$$s(\tau) = \int x(\tau)w(t-\tau)d\tau \tag{2.12}$$

where we again usually donates the operation as:

$$s(\tau) = (x * w)(\tau) \tag{2.13}$$

Here $x(\tau)$ donates a function dependent on τ and $w(\tau)$ donates the other function also dependant on τ, but with a time-offset as well represented by t. In the general form one can regard t as going from $-\infty$ to $+\infty$. When the two functions are non-zero, the integral of the product will be computed creating a sliding effect between the two functions. The convolution operation can also be written in its discrete form to fit discrete data:

$$s(\tau) = \sum_{a=-\infty}^{\infty} x(\tau)w(t-\tau) = (x * w)(t) \tag{2.14}$$

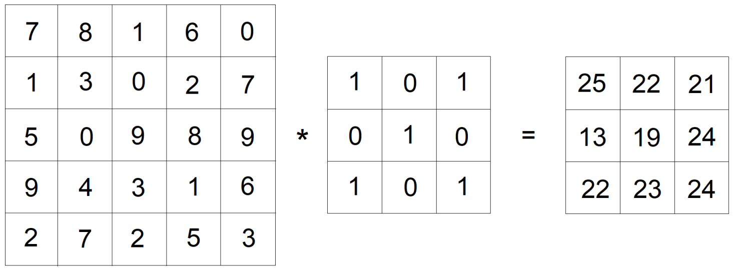
The convolutional layer(s) are usually present as the input layers, but may also be deeper in the network. The first function in equation 2.12 correspond to the input data and second correspond to what is called the kernel. The values of the kernel are called the weights, or sometimes filter values, and are what is adjusted during the training of the network. The output function can be regarded as a feature map of the inputs. The size of the kernel is to be decided from the problem on hand. As previously mentioned the CNNs historical use has been in the field of image recognition. This ability can have a wide area of use, among them audio and sound analysis, for instance in the case of time-frequency analysis with scaleograms and spectograms. In the general case the inputs will be multidimensional arrays of data, and the kernel will be arrays of a similar dimension, but usually smaller in size. The values of each element in the kernel will be changed and updated as a part of the networks learning process.

A concrete example of inputs can be regarded as follows: The input data may be a 2D-grid with different numerical values, where high values will indicate darker pixels and low values will indicate lighter pixels. The kernel that is to be convoluted with this 2D grid is initialized with random values and the size of the grid is freely chosen, but usually some magnitudes smaller than the input grid and with the same dimensions. The values in the kernel and the input are in the world of machine learning referred to as tensors. Each value in the input and kernel must be stored separately, and we can by this assume that these functions are zero everywhere but in the finite set of points that these values are present. The practical use of this is that we can implement the infinite summation as a summation over a finite number of array elements[16]. We can build on the convolution function in equation 2.14 to involve more than one dimensions at a time;

$$s(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) = (I*K)(i,j) \tag{2.15}$$

where I represent an input array, or image, and K represent the kernel. In machine learning algorithms the related cross-correlation is often applied. This is the same as the convolution but without reversing one of the inputs. Both these two operations are just referred to as convolution in machine learning applications and the functions of the machine learning library used mostly takes care of this for the user. The difference is only present in the learning algorithms and the issue can be disregarded by users.

In the case of discrete convolution each instance of the convolution steps can be viewed as a matrix multiplication. In figure 2.11 it can be seen an example of how the output of a simple convolution could look.



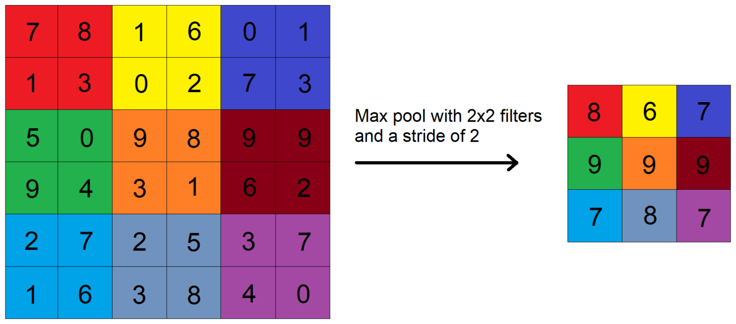
$$\begin{bmatrix} 7 & 8 & 1 & 6 & 0 \\ 1 & 3 & 0 & 2 & 7 \\ 5 & 0 & 9 & 8 & 9 \\ 9 & 4 & 3 & 1 & 6 \\ 2 & 7 & 2 & 5 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 25 & 22 & 21 \\ 13 & 19 & 24 \\ 22 & 23 & 24 \end{bmatrix}$$

Figure 2.11: *The figure shows an input of 5x5 values convoluted with a kernel of 3x3. The kernel is moved over the input by shifting one element at a time.*

How the input is convoluted with the kernel can be fine tuned from the application. The size of the kernel and the initial values of it is also decisive of the performance. The parameter stride can also be regarded, and decides how many steps the kernel takes per convolutional step. The kernel moves from left to right, and from the top to the bottom of the input grid.

2.2.2 Pooling layers

Another important layer of CNNs that distinguish them from simple forward networks are the so called pooling layers. A usual structure for CNNs is to have the outputs from the convolutional layer(s) to be passed through a non-linear activation function[17]. The outputs from these activations are then passed through a pooling layer. In figure 2.12 a max pooling can be seen.



$$\begin{bmatrix} 7 & 8 & 1 & 6 & 0 & 1 \\ 1 & 3 & 0 & 2 & 7 & 3 \\ 5 & 0 & 9 & 8 & 9 & 9 \\ 9 & 4 & 3 & 1 & 6 & 2 \\ 2 & 7 & 2 & 5 & 3 & 7 \\ 1 & 6 & 3 & 8 & 4 & 0 \end{bmatrix} \xrightarrow{\text{Max pool with 2x2 filters and a stride of 2}} \begin{bmatrix} 8 & 6 & 7 \\ 9 & 9 & 9 \\ 7 & 8 & 7 \end{bmatrix}$$

Figure 2.12: *Illustration showing how a larger array can be minimized by utilizing max pooling to get only the most distinct features*

These layers are meant to reduce the dimensions of the data by combining outputs from several neurons in the previous layer to a single neuron. The size of the pooling can vary from smaller clusters of neurons to larger sizes working on all the neurons. It is separated between max and average pooling, where max pooling returns the max value of the neuron cluster and average returns the average of the neurons. When considering pooling it is also useful to consider stride. Stride is the same as for the convolutional layers, and it decides how far the pooling filters move between each calculation.

2.2.3 Forward layers

This is usually the end layer(s) of a traditional CNN. Often implemented as a simple network as seen in figure 2.10. After one or more convolutional and pooling layers the data is fed into a conventional fully connected forward network with a variable amount of hidden layers. After the previous layers the data will be more compressed. This makes it possible to have computational effective number of neurons in the input layer of the forward layers. The output layer of the forward network will correspond to the number of different classes that the initial problem aims to separate between, if the problem on hand is a classification problem.

2.3 Metrics for data classification

When working with machine learning algorithms it is separated between training the network, validating it and testing it. These 3 datasets should contain data from the same classes, but different data in each set. There is no textbook answer to the required amount of data in each set, and this is completely up to the task on hand. It depends on the complexity of the task, the number of classes to be predicted, the complexity of the network and other factors. A rule of thumb is however to separate the data between 70% training, 15% validation and 15% testing. Training is then done by sending the training data through the network, usually in batches, and then update the weights of each neuron through a loss function and a optimizer function in a back propagating manner. The training data is sent through a variable amount of times, and this is called the number of epochs. When all this is done the goal is to measure the performance of the algorithm.

There are many ways to determine the performance of a CNN or machine learning algorithms in general. A multiple of metrics to do so exist. Which metric to use depends on the problem on hand and what the use of the machine learning algorithm is meant to be. For a classification problem where data is to be predicted as being a member of one, two or several different classes there exist a multitude of different ways to rate the performance of the network as well. There is no definite answer to how the performance should be measured. The metrics that is chosen to evaluate the machine learning model is however extremely important. The metrics influences how the performance of machine learning algorithms compares to other algorithms, and a choice of metrics that does not fit the problem can be very misleading of the actual performance.

2.3.1 Confusion matrix

A central term when finding metrics is the confusion matrix. In simple two-class classification problems confusion matrices are relatively simple. They however become more complex when considering multiclass and multilabel problems. A simple form of a confusion matrix can be seen in figure 2.13.

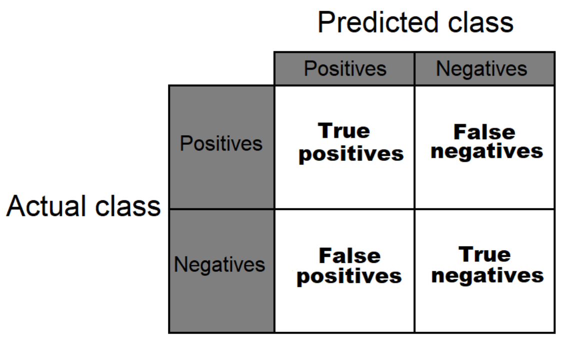


Figure 2.13: *Simple confusion matrix showing how outputs of a classification algorithm can be organized to find useful statistics.*

The analogy from the figure can be extended to multiclass and multilable problems as well by regarding the outputs for each class by themselves. The terminology indicates the following: When an output of the machine learning algorithm is predicted to be a member of a class, and this is actually true, there is a true positive(TP). When an output is predicted to be a member of a class, but is not actually in the class there is a false positive(FP). If an output is not predicted to be in a class but is actually a member of the class there is a false negative(FN) and when an output is predicted as not being in a class and is actually not there, it is a true negative(TN). The goal of all models is to minimize false positives and false negatives, but which one of most importance is case dependant.

2.3.2 Accuracy

A metric often used is the accuracy. And it can easily be found from the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.16}$$

Accuracy is a good indicator when having balanced classes, that is when there is approximately an equal amount of data being a member of each class. However when there is an overweight in one or more classes, the accuracy can be high even though the algorithm completely ignores correct classification in the classes with fewer members.

2.3.3 Precision and Recall

Two important metrics to consider when looking at performance is precision and recall. These are related to the confusion matrix as well, and this can be seen in figure 2.14

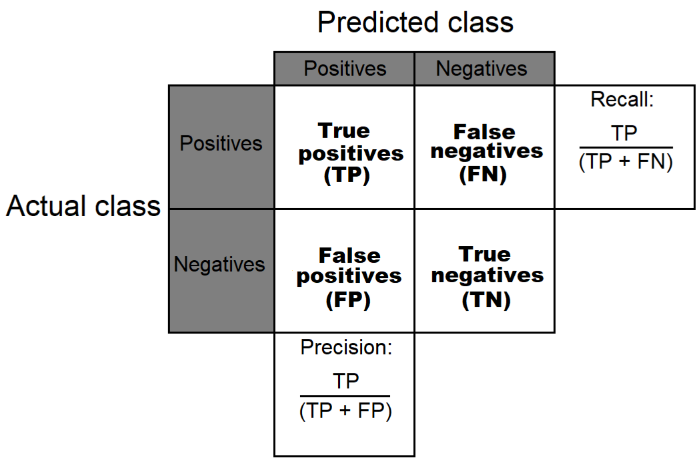


Figure 2.14: *Confusion matrix including precision and recall metrics.*

Precision is a measure of how many of the members classified as being in a class that is actually a member of that class. This implies that having a high precision indicates that a high number of positively classified entries are correct. Recall, on the other hand, indicates the proportion of true members of a class actually being classified as belonging to that correct class. So having a high recall indicates that a high number of members belonging to a certain class is classified correctly. The goal of any algorithm is to have as high precision and recall as possible. This shows that the algorithm returns results that are accurate (high precision), as well as returning most of all correct results (high recall). An algorithm with high recall and low precision returns many results, but most of its predicted outputs are incorrect when compared to the true labels. A system with high precision but low recall is the opposite: There are few results, but most of the predicted labels are correct when compared to the true label.

Another way to use precision and recall is to look at the area under the curve the two values make together (AUPRC) with different classification thresholds. This means that the threshold for what probabilistic level an output of the algorithm will be considered a member of the class is changed, so to give different values for precision and recall. A visualisation of this can be seen in figure 2.15. A high area under the curve indicates a high recall and a high precision, and then again a better algorithm.

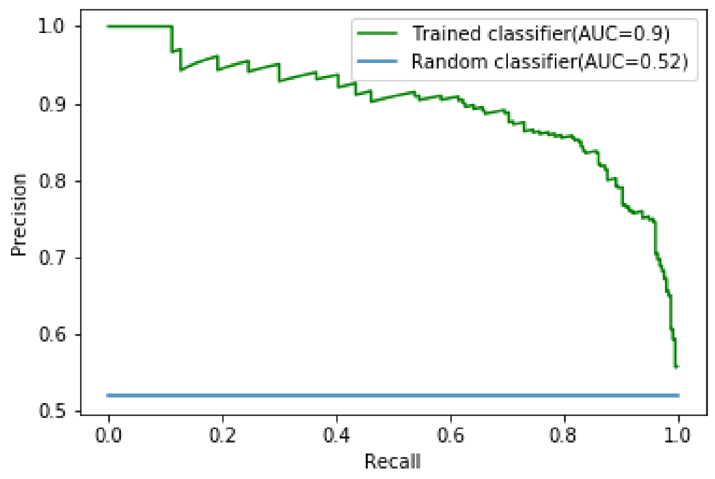


Figure 2.15: *Figure showing curves formed by precision and recall values at different probabilistic classification thresholds for a trained and a random binary classifier. The areas under the curves (AUC) are also given. It is important to note that a badly trained classifier can perform worse than the random one as well.*

2.3.4 F1-score

In the case where there is imbalance between the precision and recall scores it is convenient to use the F1-score which is given by equation 2.17

$$F1score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.17}$$

F1-score donates the harmonic mean between precision and recall, and gives a better measure of the incorrectly classified cases than the accuracy score. By using the F1-score it is possible to look at the relation between precision and recall. This score will give a more balanced result if the precision and recall values are drastically different from each other.

2.3.5 Macro vs. Micro averaging

These metrics can all be regarded in a macro or micro averaged way as well. A macro average computes the metrics independently for each class and then take the average. This will treat all classes equally and do not take into account imbalances in classes. This can lead to classes having a few number of entries contributing a lot to the final score, either negatively or positively. A micro average will sum up all the contributions of all classes to compute the average metric as a whole. Micro average is often used for imbalanced data due to all contributions being regarded the same independent of class. This can again lead to misleading results if some classes only have a few entries and are completely ignored. All metrics have their pros and cons, so comparing several of them will give a better picture of the performance.

Chapter 3

Experimental setup

In this chapter the setup and limitations for the experiment are presented. The perspective used to find this setup and the limitations is based on a constructive research approach[18]. To find the features and to produce results the theory is followed and this is the main source for the assumptions made. There exists a wast amount of different configurations for each transform. This comes in the form of choices regarding for instance different mother wavelets, amount of scales, window function and window size. A fair comparison between the two transform is hard to obtain due to their fundamental differences, but this chapter tries to narrow down configurations of each transform to only look at a few that have promising outlooks. The end-goal is to compare how a less traditional time-frequency analysis tool such as the wavelet transform performed, on this kind of data, compared to the more well known STFT. Due to this fact, the experiment focused more on testing different wavelet parameters than STFT parameters. Therefore, a bigger amount of wavelet features were extracted and tested. This chapter will first present an overview of the dataset used. Then the main ideas, concerns and assumptions regarding the task at hand will be discussed and forged into an implementation.

1 The Dataset

The dataset consists of audio provided through SINTEF Digital and recorded by Norsonic. It contains 2960 unique sound files, where each has a unique label. There are 11 classes in total with a varying amount of files belonging to each class. An overview is provided in table 3.1 below.

Class name (Class nr)	Description	Count
HND_AGR (1)	Handheld AngularGrinder	72
HND_HDR (2)	Handheld Hammerdrill	254
HVM_DIG (3)	Heavy Machinery Digger General	986
HVM_DIG_BLT (4)	Heavy Machinery Digger Beltmove	45
HVM_DIG_ENG (5)	Heavy Machinery Digger Engine	721
HVM_DIG_GRB (6)	Heavy Machinery Digger Grab-work	271
HVM_DIG_JCK (7)	Heavy Machinery Digger Jackhammer	53
HVM_DRR (8)	Heavy Machinery Drill Rig	58
ROD_CAR (9)	Road Car	337
ROD_MOP (10)	Road Moped	33
ROD_TRU (11)	Road Truck	130

Table 3.1: *Overview of classes present in the dataset, a description of them and the count of each.*

The classes are directly linked with sounds/noises found on construction sites, which again can be regarded as outdoor noise. Due to the imbalance of entries in each class it was decided to split the data into 70% for training, 15% for validation and 15% for testing so to make sure there would be an equal percentage of each class represented in all sets.

As previously mentioned the wavelet transform may have a desired trade-off between the frequency and time resolution for data such as this. By looking at figure 3.1 the average frequency spectrum of each class can be seen in blue colors. The green distributions implies the variance within each class.

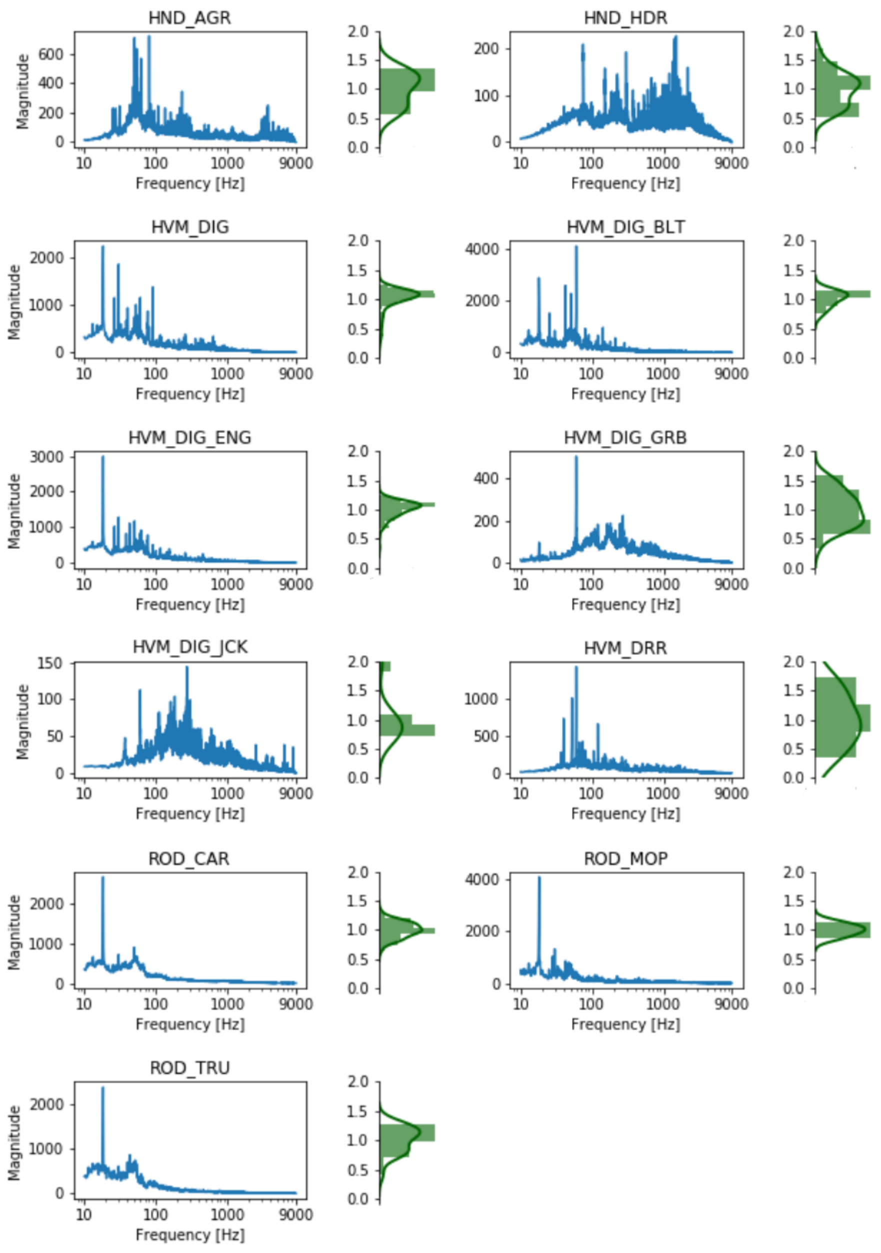


Figure 3.1: *Image showing the average frequency spectrum of each class and the deviation from this average in each class.*

As can be seen in the figure, many of the classes (e.g. HVM_DIG, ROD_TRU and HND_AGR) have a large amount of the frequency components localized in the lower frequency region. On the other hand, classes such as HND_HDR and HVM_DIG_JCK have frequency components more spread out and in the higher frequency regions. By observing the variance distribution of each class, it can be seen that some deviate more than others. For instance, one can observe that HVM_DIG_GRB deviates more than ROD_MOP. This indicated that samples found in these classes may be harder for a CNN to recognize.

2 Assumptions and limitations

2.1 Data length

The length of the sound files are mostly 5 seconds. Some files however differs in length, and are either longer or shorter than this. For the CNN to work as desired all data has to have equal length, so to create equally sized tensors. A solution to this problem was to set all lengths of the sound files to 5 seconds. This was done by either cutting away parts of the sound or by adding zero-valued samples to extend the file.

2.2 Wavelet transform

DWT are not suited for time-frequency analysis, as described in the theory. It was therefore natural to make the assumption that the CWT is the best wavelet transform for the task at hand.

There are many different wavelet families, and their resulting transform slightly varies. For this reason, it was decided to test different families and conclude which families that give the best result for the dataset. In table 3.2 some popular wavelet families are presented. These implementations all origin from Python's PyWavelets library which is used in this thesis. These families where chosen due to their popularity in many fields as described in the theory, and their availability in the PyWavelets library. The standard configuration of the morlet wavelet found in this library correspond to a morlet of order 5. Due to this the morlet of order 5 will simply be referred to as "morlet" through the rest of this thesis.

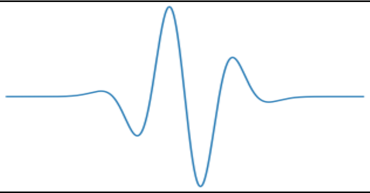
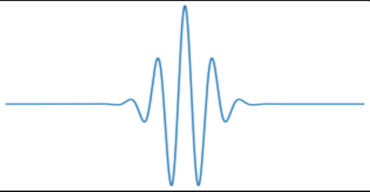
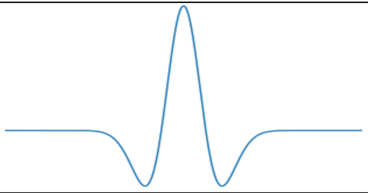
Wavelet family	Formula	Plot
Gaussian Derivative	$\Psi_5(t) = C_5(-8e^{-t^2}t(4t^4 - 20t^2 + 15))$	
Morlet	$\Psi_5(t) \approx e^{-\frac{t^2}{2}cos(5t)}$	
Mexican Hat	$\Psi_1(t) = \frac{2}{\sqrt{3}\sqrt[4]{\pi}}e^{-\frac{t^2}{2}}(1-t^2)$	

Table 3.2: *Overview of different wavelet families for the PyWavelets library*

2.3 STFT

Multiple different window functions and sizes can be used for the STFT. To limit the analysis, a selection of viable STFT configurations was chosen. The window function used for all STFT features is the Hanning window. It has been widely used and is one of the most popular window functions. Three different window sizes were implemented to see how this would impact the results due to the change in frequency and time resolution. Each window, for all the features found with STFT, had a 50% overlap so to capture any eventual information that would have been cut in between two windows.

2.4 Frequency range

For the sake of limiting the size of the scalogrames and spectograms, and thus minimizing the network run-time, a frequency range had to be fixed. A young, healthy human ear can usually hear frequencies from 20 Hz to 20 kHz[19]. As there may be some useful information even below the human threshold; the lower limit of the analysis was set to 10 Hz. The upper limit is based on the highest frequency components with information of interest. In figure 3.1 the average spectra of the sound files from each class are shown. There are generally small magnitudes of high frequency components, and therefore the upper limit was chosen to 9 kHz. This lead to the use of a sampling frequency of 18 kHz.

2.5 Distribution of frequency components

The amount of frequency components/scales and their distribution (logarithmic, linear etc.) impacts how the spectrograms and scaleograms appear. To achieve the same frequency distribution for all wavelet families a frequency bank of logarithmic distribution from 10 Hz to 9 kHz was created. The scales for each wavelet was then mapped by this bank of frequencies. When using a large amount of scales there will be a substantial overlap between the frequency responses of the different scaled wavelets. By using less amounts of scales there will, on the other hand, be a lack of overlap. As the theory indicates, some of the mother wavelets would likely be more susceptible to suffer from a high number of scales. As shown in figure 2.9 the mexican hat wavelet has a high frequency bandwidth which will result in a need for less amounts of scales.

2.6 Truncation of time

When calculating the CWT from a sound file, the resulting matrix of coefficient becomes $K \times N$, where K is the amount of scales and N is the amount of samples. When the sampling frequency f_s of the file is high, N will naturally also be large. Even though the the maximum time resolution can be excellent, the amount of resulting data might be overwhelmingly large. This leads to unnecessary long run times for the CNN as well as a impractical use of storage space.

This problem was solved by truncating the coefficients along the time axis. Truncation by a factor of L was done by averaging the absolute value of the L coefficients and combining these. This resulted in a matrix of size $K \times (N/L)$.

2.7 The problem of false frequency components

When creating the scaleograms with sampling frequency 18 kHz a problem occurred. When performing CWT with an audio signal containing high frequencies, the resulting scaleograms ended up containing components in the low frequency range. These components were often amplified to the point that it suppressed other valuable information.

A lot of speculation went in to this problem, but no concrete answer was found. Therefore a work-around was suggested. By lowering the sampling frequency before performing a CWT it was noticed that these "false components" disappeared. A solution was then to perform two separate analyses; one for high frequencies and one for low frequencies. For the reason of simplicity and convenience, the definition of high frequencies was set to be in the range 900 Hz to 9 kHz. Low frequencies was defined as 10 Hz to 900 Hz. The main difference between analysing the high frequency band and the low was that the sampling frequency was changed from 18kHz to 1.8kHz, or equivalently a decimation with a factor of 10.

When the scaleograms for both high and low frequency were created, they were merged into one picture. Since the high frequency scaleogram had 10 times higher sampling frequency, it also had 10 times the amount of values along the time axis. To be able to merge them, the high frequency scaleogram had to be truncated with a factor of 10. This made it so the minimum truncation for these pictures was set to 10, and all further truncation was a factor of 10.

Finally, the two scaleograms were normalized. This was done by comparing the 900 Hz-components in both pictures and then normalize the rest of the picture based on the resulting normalization coefficient. Lastly, the whole picture was normalized to 1. The normalization process can be described as follows:

$$C = \frac{\sum \Psi^{upper}_{900Hz,b}}{\sum \Psi^{lower}_{900Hz,b}} \tag{3.1}$$

$$\Psi_{a,b} = [\frac{\Psi^{upper}_{>900Hz,b}}{C}, \Psi^{lower}_{<900Hz,b}] \tag{3.2}$$

$$\Psi'_{a,b} = \frac{\Psi_{a,b}}{max(\Psi_{a,b})} \tag{3.3}$$

where C is the normalization coefficient, $\Psi_{a,b}$ are the CWT coefficients and $\Psi'_{a,b}$ are the CWT coefficients normalized to 1.

An overview of the work-around for the problem of false frequency components can be seen in figure 3.2. Figure 3.3 compares an STFT implementation and two different CWT implementations, whereas one was created with two sampling frequencies. The two scaleograms were both created with a morlet mother wavelet, 100 truncation and 50 scales. It can be observed that the ordinary CWT (3.3b) contains a substantial amount of low frequency components that is not present in the STFT (3.3a). With the new implementation of the CWT using two sampling frequencies most of the low frequency components no longer have a false impact like before.

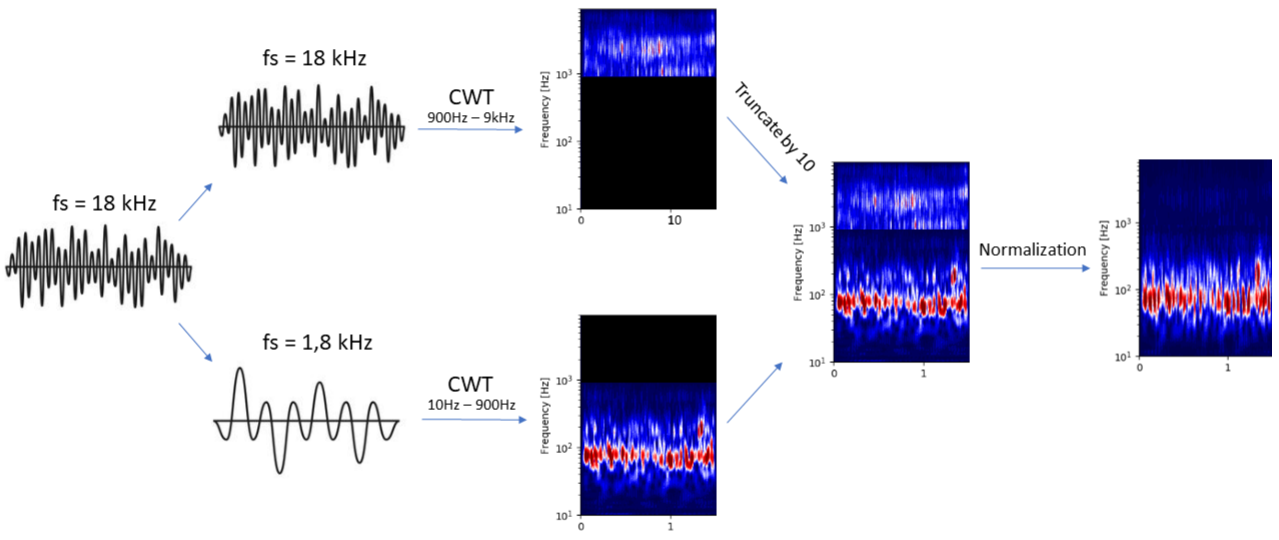
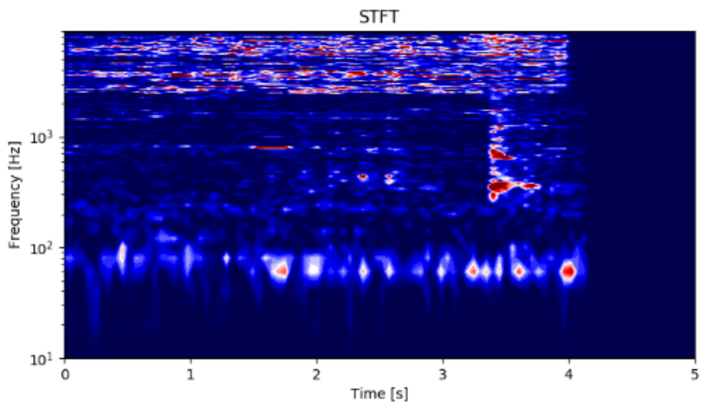
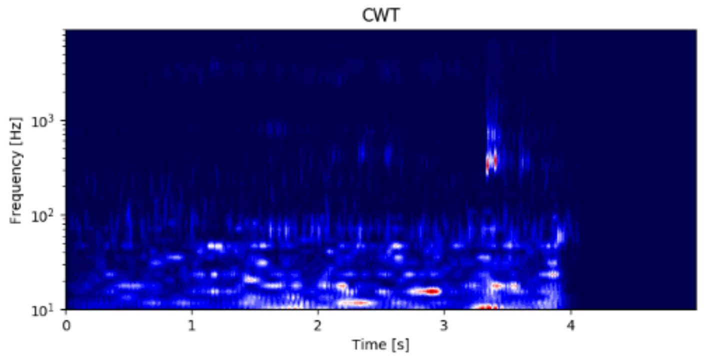


Figure 3.2: *The process of merging two scaleograms created with different sampling frequency*

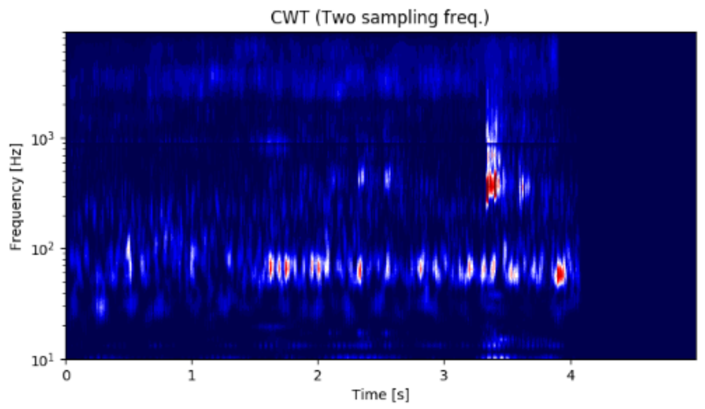
Performing CWT, using PyWavelets, on high frequency sound can be concluded to cause false frequency components. The solution, used throughout this thesis, was the analysis with two sampling frequencies as suggested in this section.



(a)



(b)



(c)

Figure 3.3: *Comparison between STFT (a), CWT with morlet (5) (b) and CWT with morlet (5) and two sampling frequencies (c).*

2.8 Visual evaluation of time truncation

To get a picture of how the time truncation affected the results from the wavelet transform it was decided to look at visual differences to identify if there is information lost - and to what degree.

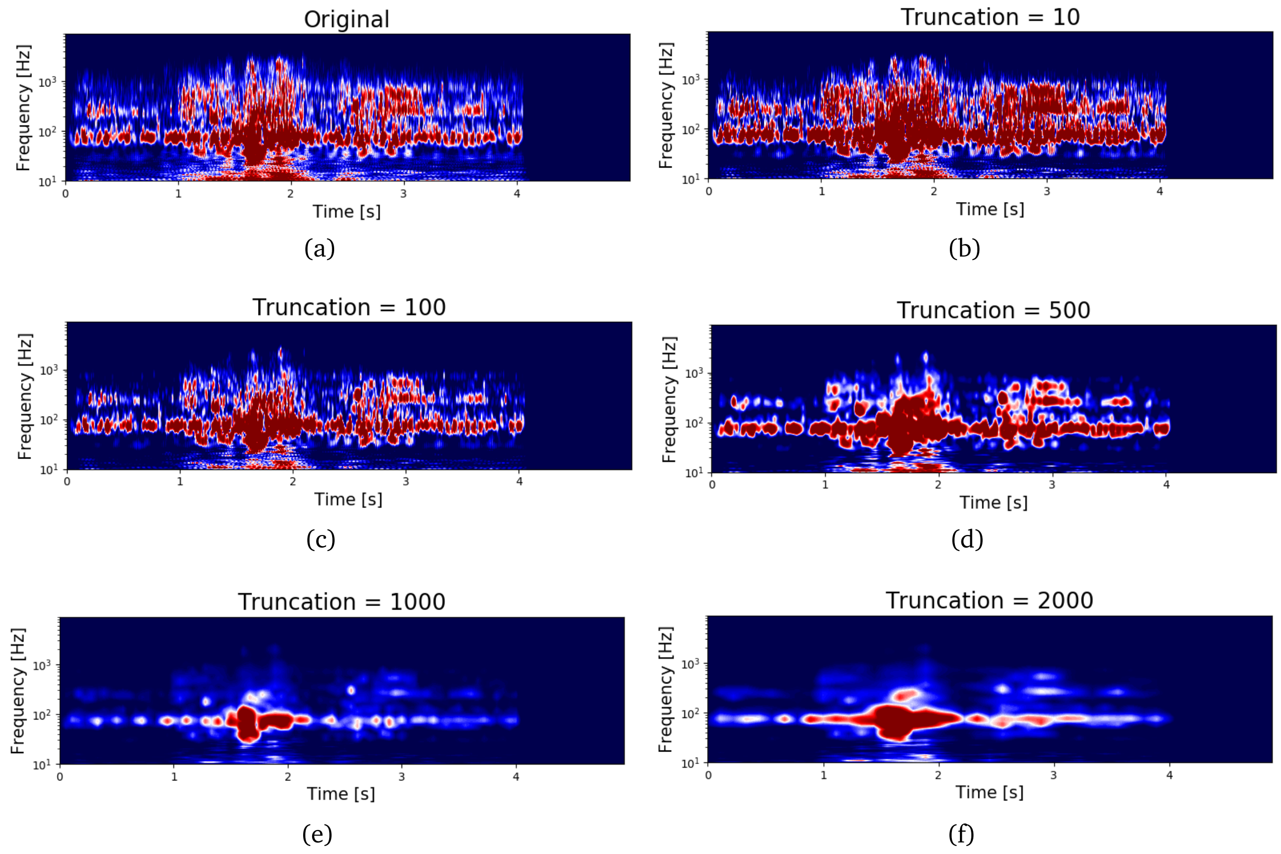


Figure 3.4: *Display of how the truncation factor affects the resulting scaleogram picture. These scaleograms were produced with morlet mother wavelet and 200 scales.*

Figure 4.1 displays how different truncation factors affected the resulting scaleograms when performing CWT. The scaleograms were produced with a morlet mother wavelet with 200 scales corresponding to 200 logarithmic frequency components between 10 Hz and 9 kHz. As expected; more and more details of the scaleograms disappeared as the truncation factor increased, but it was also clear that the main structure stayed intact. At higher truncation factors, like 1000 and 2000, it was clear that there can occur considerable loss of information. Although there will be higher loss of information for high truncation factors, it can also be observed that the contours of the sound will be more distinct. It is evident that truncation of the scaleograms will lead to loss of information. It was however deemed necessary so to reduce storage space.

2.9 Visual evaluation of distribution of scales

In figure 3.5 scaleograms with different amount of scales, from 10 to 500, are displayed. The scales were translated from a logarithmic distribution of frequency components from 10 Hz to 9 kHz.

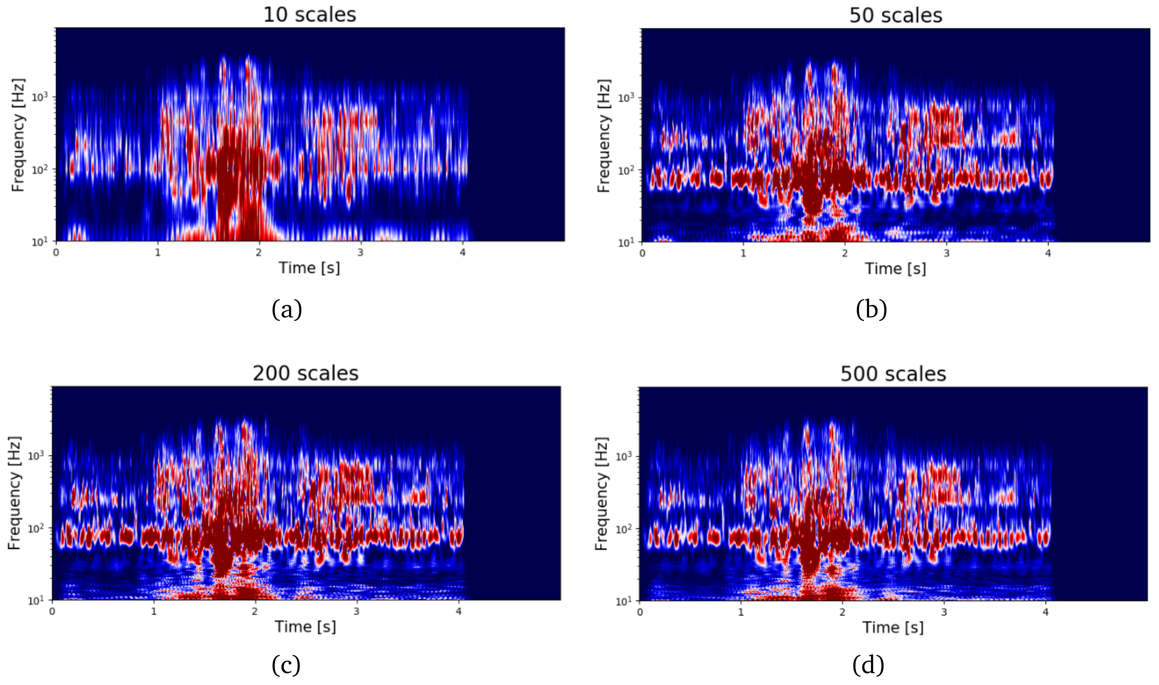


Figure 3.5: *Display of how the amount of scales affect the resulting scaleogram picture. These scaleograms were produced with a morlet mother wavelet and no truncation.*

By inspection of figure 3.5 it was observed a considerable improvement in resolution from 10 to 50 scales. The improvements in frequency resolution from 50 scales up to 500 scales are not as prominent by inspection, but some change can still be observed in the lower frequency bands. For wavelets with high bandwidth such as the mexican hat, the overlap between the different scaled wavelets occur at a lower amount of scales than for low bandwidth wavelets, e.g. morlet of order 14. This fact can be observed in figure 2.9. Just by inspecting figure 3.5 there can not be seen any considerable disadvantage of having a massive amount of scales. However as stated in the theory, and as can be seen in figure 2.9, the amount of scales that can be used is dependant on oscillations in the mother wavelet and the corresponding frequency response.

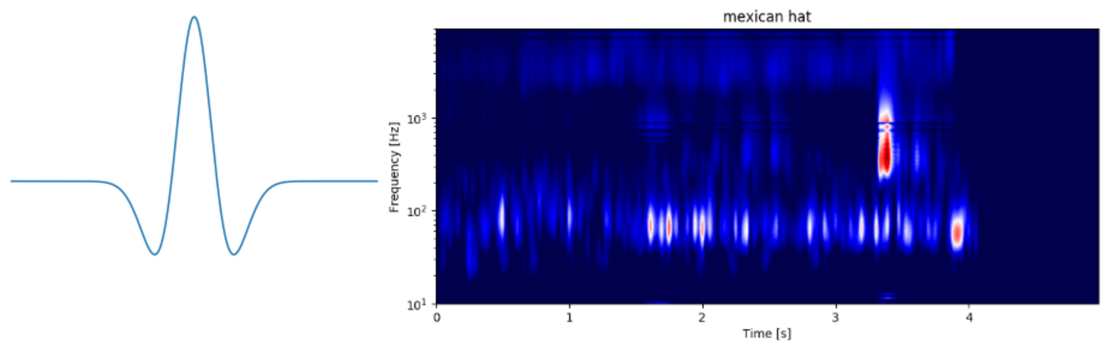
2.10 The final features and data size

A lot of different features were tested throughout the process, and in table 3.3 these are listed. For the STFT features, three different window lengths were used with two different amounts of frequency components (500 and 1000), except for the shortest window where only one amount of components were used. The window lengths chosen where 10 ms, 100 ms and 500 ms. Each extraction were done with Hanning windows with 50 % overlap. There were mainly chosen three different mother wavelets; the mexican hat, the morlet and gaussian(5). These can be seen in figure 3.6. Due to the visual inspection of different scaleograms with varying scale and truncation, it was decided to test for three different levels of scales (30, 50 and 200), and two different truncation factors (100 and 500). Both truncation factors were tested with 50 scales. A truncation factor of 100 was used with 30 scales and a truncation factor of 500 was used with 200 scales. These values were chosen to reduce extraction time and data size while still maintaining performance. After the initial results, it became apparent that more oscillations in the mother wavelet gave better results. Therefore three more features were extracted using the morlet wavelet by increasing the order to 11, 14 and 17 which can be seen in figure 3.7. Comparing figure 3.7 and 3.6 it becomes clear that increased oscillations in mother wavelet improves frequency resolution while reducing the time resolution. By visual inspection this seems to give an overall better and more detailed representation of the sound.

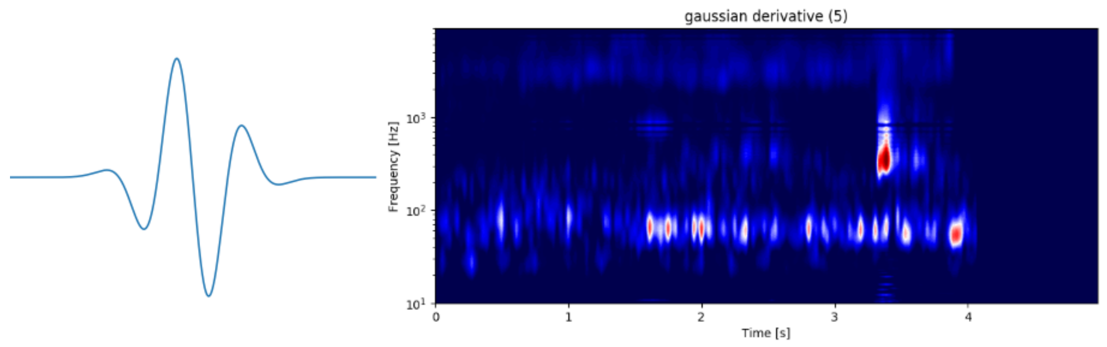
Truncation factor	100			500		
Scale	30	50	200	30	50	200
Mexican hat	X	X	X	X	X	X
Gaussian(5)	X	X	X	X	X	X
Morlet(5)	X	X	X	X	X	X
Morlet(11)						X
Morlet(14)						X
Morlet(17)						X

Frequency components	500			1000		
Window size	10ms	100ms	500ms	10ms	100ms	500ms
STFT		X	X	X	X	X

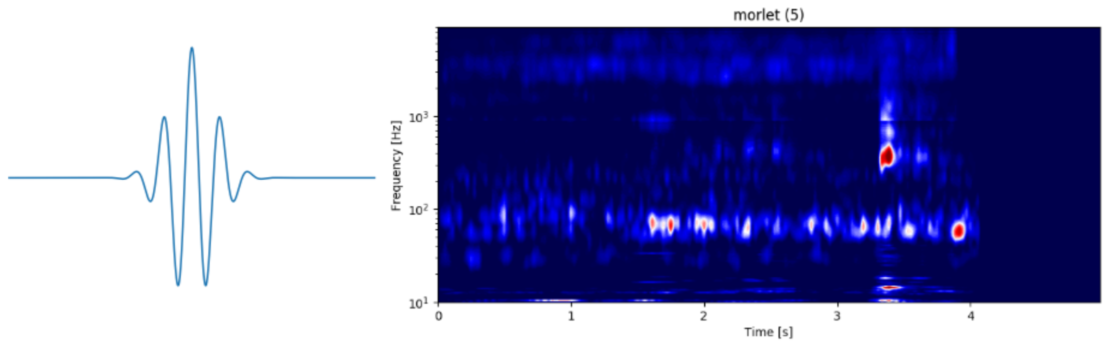
Table 3.3: *The features for CWT and STFT that were used in the making of the results.*



(a)

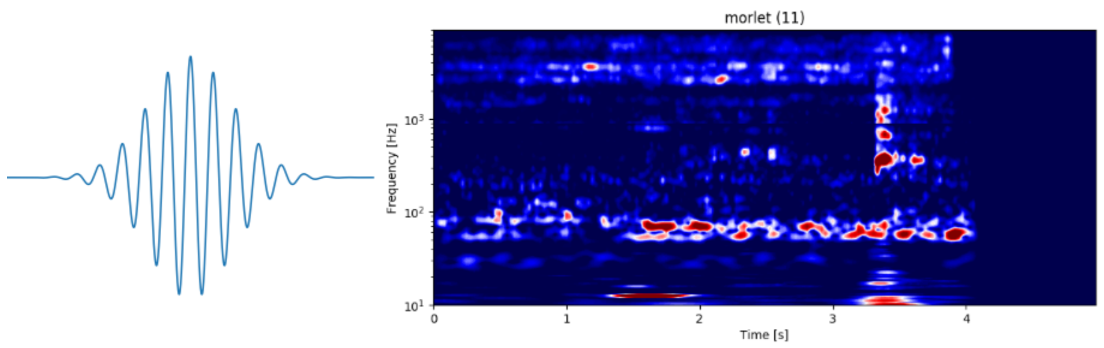


(b)

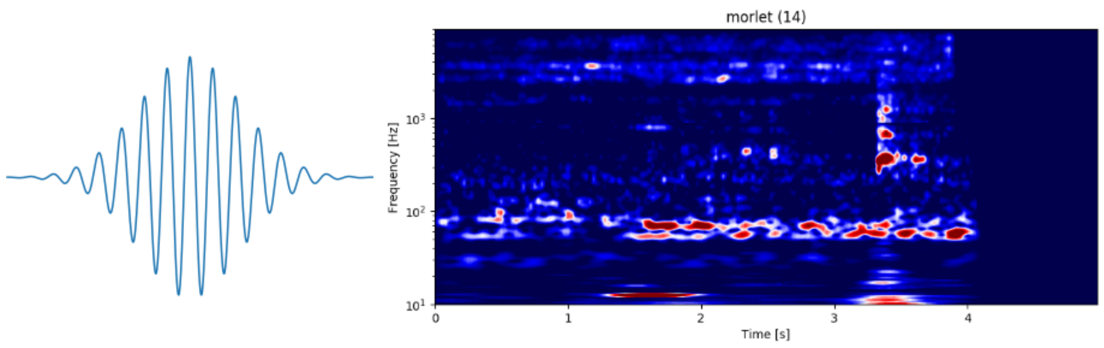


(c)

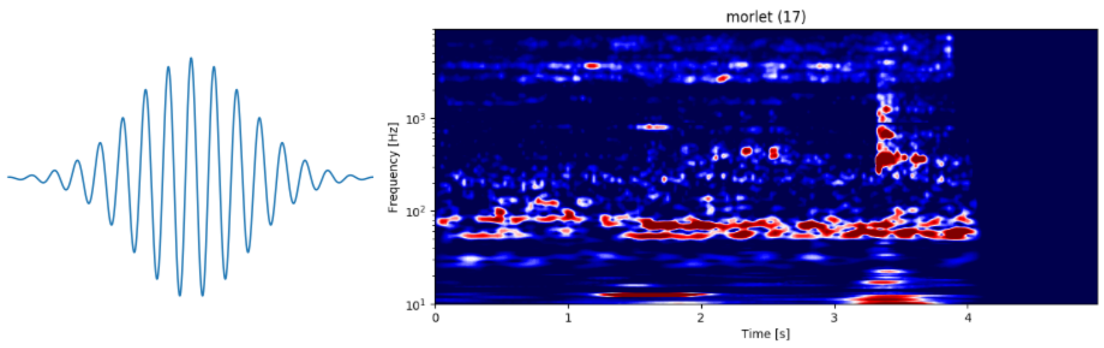
Figure 3.6: *An example of the scaleograms for mexican hat (a), gaussian derivative of order 5 (b) and morlet of order 5 (c). The mother wavelet is shown on the left side.*



(a)



(b)



(c)

Figure 3.7: *An example of the scaleograms for morlet of order 11 (a), 14 (b) and 17 (c). The mother wavelet is shown on the left side.*

3 CNN

3.1 Creating the CNN

Since the main goal was to see if features obtained with the wavelet transform could be superior to more traditional features such those found with STFTs, it was decided that focusing on optimizing the CNN would come second-hand. It was decided to use Python's PyTorch library to create the network. It was experimented with several architectures, and the final setup gave the best trade-off between run-time and complexity. The final CNN can be seen in figure 3.8, and consisted of two convolutional layers followed by a pooling layer. The data was then to be fed into a regular forward network with three layers. Finally the data would end up as being classified as one of the eleven classes in the last forward layer.

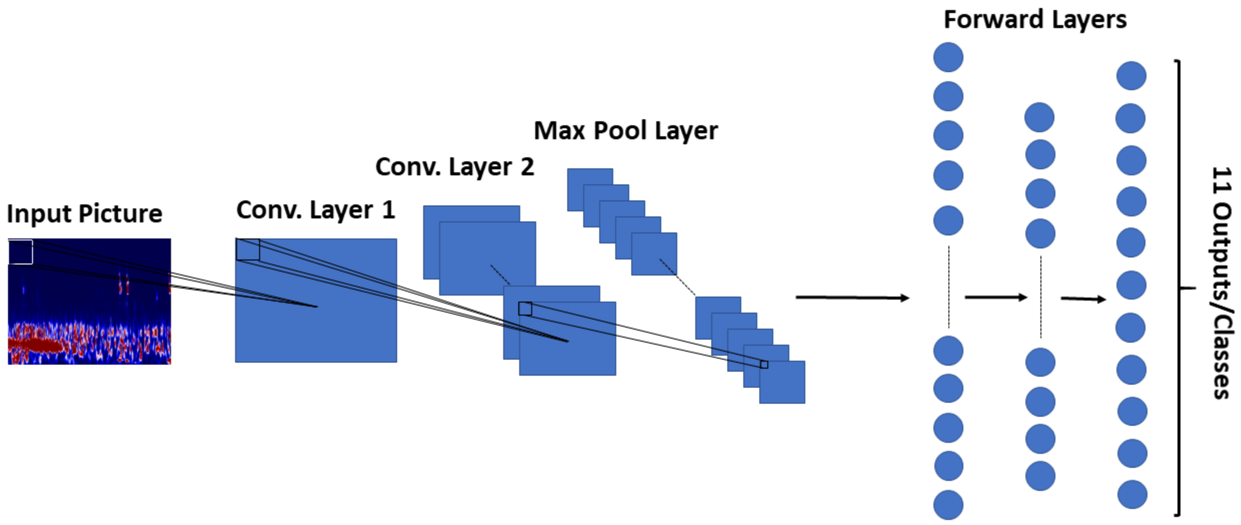


Figure 3.8: *Visualisation of the final network architecture. The number of neurons in is layer are not exact, except from in the output layer.*

The outputs from all the layers were passed through a ReLU activation function before the data in the last output layer was passed directly to a loss function. The ReLU function was used due to its popularity in neural networks in recent years. This structure fulfilled the criterion of keeping the network simple. Another advantage was that the ReLU function can lead to less dense solutions which comes from the fact that it does not always give a non-zero output, but sets all negative outputs to zero. This again leads to less neurons being utilized every turn and less dependence between features, as well as making it more computationally effective. The ReLU activation function can be seen in figure 3.9.

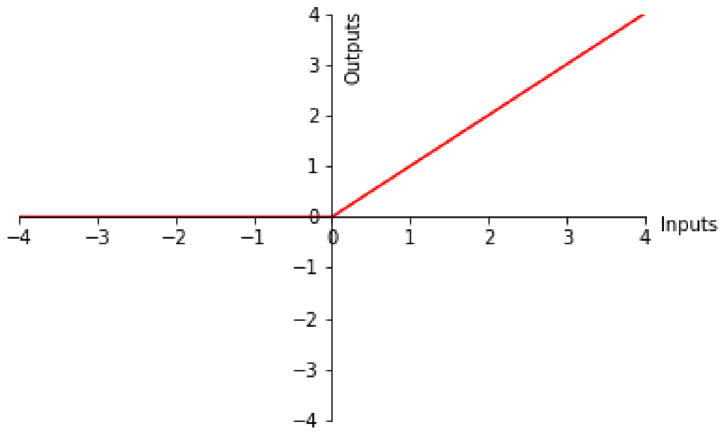


Figure 3.9: *ReLU activation function.*

After each epoch a loss and an optimization direction are calculated. To do this, the cross entropy loss function and the ADAM optimizer was used. The cross entropy loss was used due to its popularity and its proven performance for practical problems. The loss function is a combination of NLLLoss and log-softmax. This makes it possible to send the data directly to the loss calculation without using an activation function in the last layer. The log-softmax gives the log-values of a probabilistic output of a certain class occurring, making it suitable for an n-class problem such as the one at hand. The ADAM (Adaptive Moment Estimation Algorithm) optimizer was used due to its simplicity, computationally efficiency and well documented usage. It was experimented with using other optimizers, such as stochastic gradient decent, but the fine tuning needed for other optimizers were more time consuming than for the ADAM optimizer. So the final implementation ended up using the ADAM optimizer.

The network was made so to be compatible with both STFT and wavelets data, so to compare the performance as closely as possible. Some parameters such as the kernel sizes of the convolutions and the input layer to the forward network needed tuning depending on the size of the input data, but other parameters were kept the same. The batch size used, as well as the number of epochs and learning rate were also kept the same. Several configurations were experimented with, and the values that seemed to perform the best for different features were kept for the final results. A batch size of 19 combined with 50 epochs and a learning rate of 0.0001 was used to produce the final results. Bigger batches performed worse, more epochs tended to over-fit and using a larger or smaller learning rate did not fit with the other configurations.

3.2 Finding useful statistics

To compare the performance of the different features it was decided to focus on four different metrics. These metrics were micro F1-score, micro AUPRC, macro AUPRC as well as accuracy. The F1-score was chosen to be able to get a balanced score between the precision and recall.

AUPRC was chosen to look at the relation between precision and recall for all classification thresholds. Having both macro and micro averaged values once again came from the fact that the classes where imbalanced. The macro averaged results of the AUPRC was included to detect if certain classes were missed by the network. It also served to give a broader analysis and to discover any eventual differences between CWT features and STFT features. The accuracy was included as an extra metric to get as in-depth analysis as possible. Both accuracy and F1-score was found with a classification threshold of 0.5 (e.g 50 %), meaning that the outputs of the network had to be over 0.5 for the correct class to be counted as a correct output.

Chapter 4

Results

The final results were produced by a CNN with two convolutional layers, one pooling layer and three feed forward layers. As described earlier, the ReLU activation function was used between layers. A cross entropy loss function was used with an ADAM optimizer to calculate loss and optimize the weights. A learning rate of 0.0001 was used with a batch size of 19 over 50 epochs. The results were obtained with the same random seed for distribution of data among the different sets and for initializing the network. This was done to have an as equal comparison as possible. Some layers in the network had to be changed so to fit the different input sizes, but everything else were kept as similar as possible.

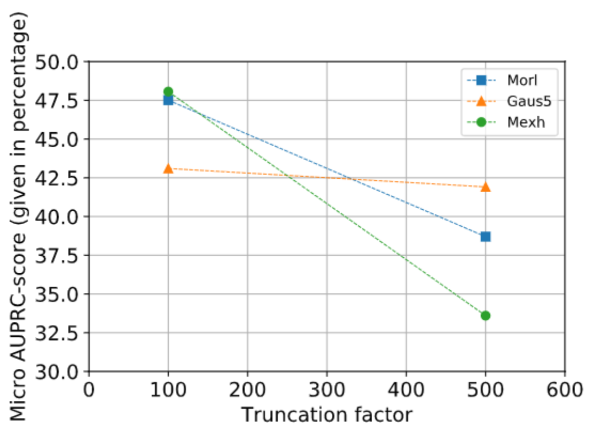
In this chapter the results are only presented so to be viewed as objectively as possible. The discussion around the results are presented in the next chapter.

Results were measured in micro and macro averaged AUPRC, micro averaged F1-score as well as accuracy, both class-wise and total accuracy. F1 score was found with a threshold of 0.5, same for accuracy. e.g network outputs must be greater than 0.5 for the correct class to be considered correct.

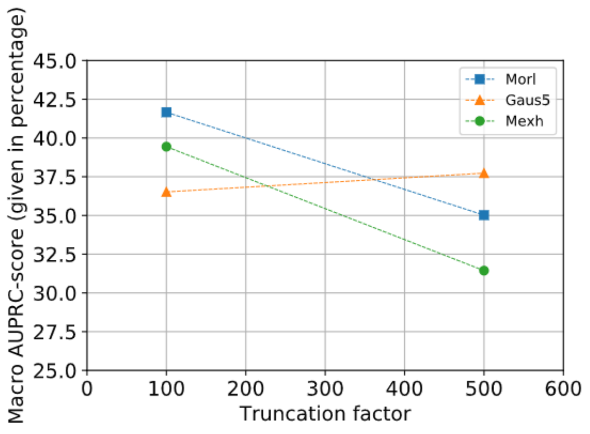
The results are presented for a total of 20 different extracted features, where 12 comes from different configurations of CWT and 5 comes from STFT. The last 3 results come from experimentation with the order of the morlet mother wavelet, and these are presented at the end of the chapter.

1 F1-score and AUPRC

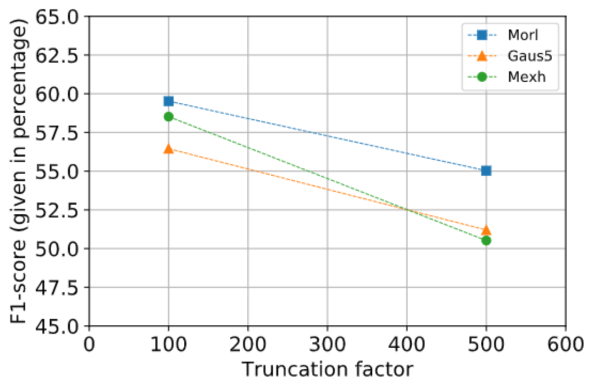
As stated earlier in the thesis it was necessary to truncate the time axis for the scaleograms due to the potentially huge data files, especially in combination with the larger scale values. The impact of the two truncation factors used can be seen in figure 4.1. It can be seen that a higher truncation factor resulted in decreased performance in every case expect for a slight increase in macro averaged AUPRC-score for the gaussian(5) wavelet.



(a)



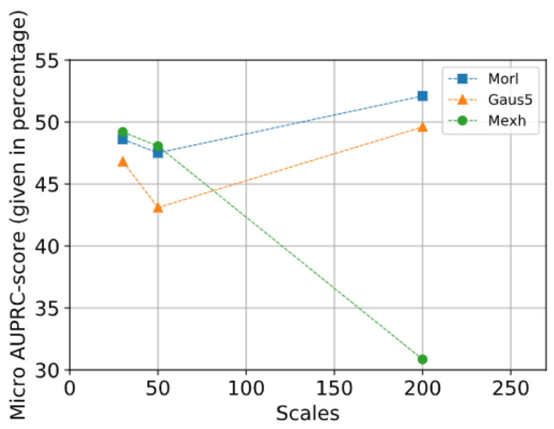
(b)



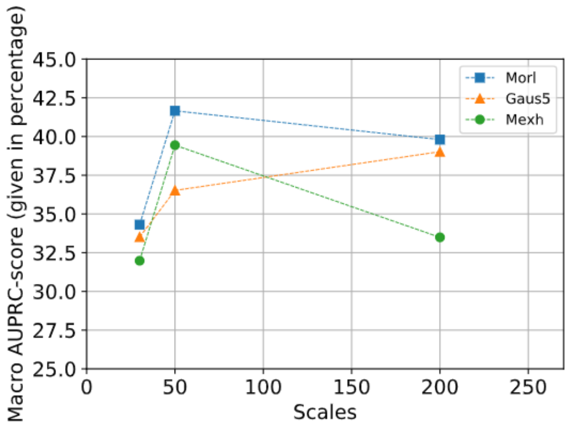
(c)

Figure 4.1: *Figure showing how three of the different scoring metrics where affected, for each mother wavelet, by the two truncation factors in use.*

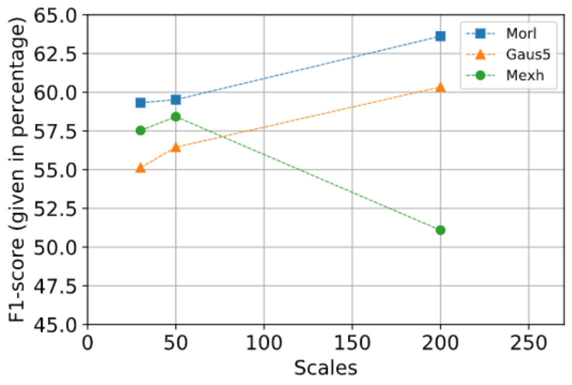
For the CWT, three different scale values were used for the different mother wavelets. The results can be seen in figure 4.2. The result from the different scales gives a more complex picture than that of the truncation factors. The trend is that the gaussian(5) gives the best results for 200 scales and this is also the case for the morlet, except for the macro AUPRC-score. The mexican hat's performance decrease at 200 scales and gives the best scores for 50 scales, except for the micro AUPRC-score.



(a)



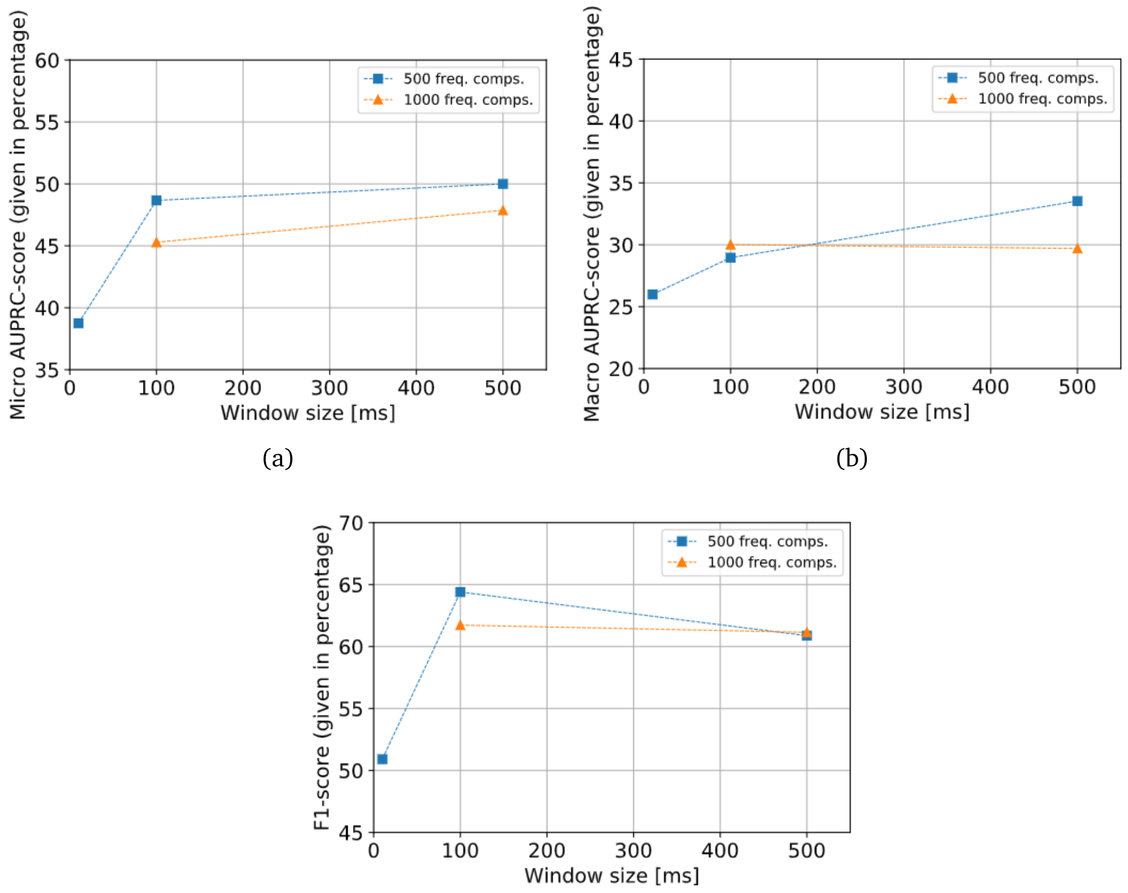
(b)



(c)

Figure 4.2: *Figure showing how three of the different scoring metrics where affected, for each mother wavelet, by the number of scales in use.*

For the STFT, three different window sizes and two different numbers of frequency components were used. The Hanning window function and a 50% overlap between each window were used in every implementation of the STFT. Figure 4.3 illustrates how the different scores were affected by the different window sizes and the number of frequency components. It can be clearly seen that the 10 ms window performed the worst for every metric, while both the 100 ms and 500 ms window performed better.



(a) (b)

(c)

Figure 4.3: *Figure showing how three of the different scoring metrics where affected, for each mother wavelet, by the number of scales in use.*

In table 4.1 the F1-score, micro and macro AUPRC can be seen for all the 15 features. The table shows the amount of frequency components used in the transform, or the amount of scales in the case of the wavelets. The time resolution is the same as the window size in the case of STFT and in the case of wavelets it is the highest theoretical obtainable resolution after the scalograms are truncated. The picture size indicate the size of the input arrays to the network. The first number indicates the number of frequency components/scales and the second is related to window size or truncation relative to the total length of the sound files (5 seconds). The color coding in table 4.2 are used in table 4.1 to imply the highest and lowest scores in each metric. Green represents the highest scores and red represents the lowest. It can be seen that the scores varies between wavelets and STFT, but that there is also a lot of variation dependant on the used configuration inside both transforms. The overall best scores in all categories was the morlet wavelet with 200 scales and 500 truncation. It does however not outperform any of the others in a way that makes it obviously better. Another trend that can be seen is that the wavelets in general outperform the STFT in macro averaged AUPRC, while the STFTs have a higher F1-score on average.

Method	Freq. comp.	Time res. (after truncation)	Picture size	micro AUPRC	macro AUPRC	micro F1-score
STFT	1000	500ms, 50% overlap	1000x18	0.4787	0.2970	0.6115
STFT	500	500ms, 50% overlap	500x18	0.5000	0.3353	0.6088
STFT	1000	100ms, 50% overlap	1000x98	0.4529	0.3001	0.6173
STFT	500	100ms, 50% overlap	500x98	0.4867	0.2896	0.6440
STFT	500	10ms, 50% overlap	500x998	0.3875	0.2599	0.5091
CWT gaus5	30	5.6ms	30x900	0.4682	0.3351	0.5513
CWT gaus5	50	5.6ms	50x900	0.4310	0.3652	0.5645
CWT gaus5	50	27.8ms	50x180	0.4191	0.3773	0.5120
CWT gaus5	200	27.8ms	200x180	0.4961	0.3902	0.6034
CWT morl	30	5.6ms	30x900	0.4861	0.3431	0.5932
CWT morl	50	5.6ms	50x900	0.4751	0.4166	0.5952
CWT morl	50	27.8ms	50x180	0.3870	0.3502	0.5503
CWT morl	200	27.8ms	200x180	0.5210	0.3980	0.6362
CWT mexh	30	5.6ms	30x900	0.4920	0.3198	0.5753
CWT mexh	50	5.6ms	50x900	0.4806	0.3944	0.5842
CWT mexh	50	27.8ms	50x180	0.3360	0.3144	0.5042
CWT mexh	200	27.8ms	200x180	0.3985	0.3349	0.5109

Table 4.1: *Table showing metrics for a CNN trained on different features obtained with different configurations of STFTs and CWTs. It should be mentioned that the time resolution for the CWTs is the highest attainable resolution based on the truncation factor.*

🟥	🟥	🟥	🟩	🟩	🟩

Table 4.2: *Color chart for the worst (red) to the best (green) results.*

2 Accuracy score for mexican hat

In table 4.3 the accuracy values for each class can be seen for the mexican hat wavelet. The classes are listed from 1 to 11 and correspond to the class values given in table 3.1. The accuracy is given for the different configurations. At the bottom, the average of each class accuracy (macro averaged) is given as well as the total accuracy of the entire test set (micro averaged).

Class accuracy for test set (Mexican hat)				
Class: (count)	30 scales 100 truncation	50 scales 100 truncation	50 scales 500 truncation	200 scales 500 truncation
Class 1 (10)	20 %	70 %	70%	80 %
Class 2 (40)	87.5 %	92.5 %	95 %	87.5 %
Class 3 (159)	71.70 %	40.89 %	27.67 %	34.59 %
Class 4 (7)	0 %	0 %	0 %	0 %
Class 5 (116)	22.41 %	57.76 %	56.03 %	55.17 %
Class 6 (44)	90.91 %	93.18 %	88.63 %	88.63 %
Class 7 (9)	88.88 %	77.77 %	88.88 %	88.88 %
Class 8 (10)	40 %	60 %	60 %	60 %
Class 9 (53)	49.06 %	50.94 %	43.40 %	35.85 %
Class 10 (6)	0 %	0 %	0 %	0 %
Class 11 (21)	4.76 %	33.33 %	52.38 %	23.81 %
Avg. Class Acc.	43.2 %	52.4 %	52.9 %	50.4 %
Accuracy:	53.89 %	55.94 %	51.11 %	50.66 %

Table 4.3: *Table showing accuracy for the different mexican hat wavelet features.*

3 Accuracy score for gaussian(5)

In table 4.4 the accuracy values for each class can be seen for the gaussian(5) wavelet. The classes are listed from 1 to 11 and correspond to the class values given in table 3.1. The accuracy is given for the different configurations. At the bottom, the average of each class accuracy (macro averaged) is given as well as the total accuracy of the entire test set (micro averaged).

Class accuracy for test set (Gaussian(5))				
Class: (count)	30 scales 100 truncation	50 scales 100 truncation	50 scales 500 truncation	200 scales 500 truncation
Class 1 (10)	60 %	70 %	80%	80 %
Class 2 (40)	90 %	97.5 %	92.5 %	95 %
Class 3 (159)	57.23 %	66.04 %	37.11 %	47.80 %
Class 4 (7)	0 %	0 %	0 %	0 %
Class 5 (116)	22.41 %	20.69 %	56.89 %	55.17 %
Class 6 (44)	93.18 %	56.81 %	84.09 %	88.64 %
Class 7 (9)	66.66 %	88.88 %	88.88 %	88.88 %
Class 8 (10)	30 %	90 %	60 %	40 %
Class 9 (53)	71.70 %	60.38 %	45.28 %	69.81 %
Class 10 (6)	0 %	0 %	0 %	0 %
Class 11 (21)	0 %	14.29 %	33.33 %	23.81 %
Avg. Class Acc.	44.65 %	51.32 %	52.55 %	53.56
Accuracy:	52 %	53.05 %	53.30 %	58.73 %

Table 4.4: *Table showing accuracy for the different gaussian(5) wavelet features.*

4 Accuracy score for morlet

In table 4.5 the accuracy values for each class can be seen for the morlet wavelet. The classes are listed from 1 to 11 and correspond to the class values given in table 3.1. The accuracy is given for the different configurations. At the bottom, the average of each class accuracy (macro averaged) is given as well as the total accuracy of the entire test set (micro averaged).

Class accuracy for test set (Morlet)				
Class: (count)	30 scales 100 truncation	50 scales 100 truncation	50 scales 500 truncation	200 scales 500 truncation
Class 1 (10)	50 %	80 %	80%	80 %
Class 2 (40)	85 %	92.5 %	92.5 %	92.5 %
Class 3 (159)	71.07 %	77.36 %	37.11 %	72.96 %
Class 4 (7)	0 %	0 %	0 %	0 %
Class 5 (116)	24.14 %	12.93 %	67.24 %	41.38 %
Class 6 (44)	97.73 %	84.09 %	75 %	90.91 %
Class 7 (9)	66.66 %	88.88 %	88.88 %	88.88 %
Class 8 (10)	40 %	90 %	40 %	40 %
Class 9 (53)	58.49 %	62.26 %	49.06 %	62.26 %
Class 10 (6)	0 %	33.33 %	0 %	0 %
Class 11 (21)	23.81 %	14.29 %	33.33 %	23.81 %
Avg. Class Acc.	47 %	57.78 %	51.19 %	55.18 %
Accuracy:	56.63 %	57.89 %	54.73 %	62.94 %

Table 4.5: *Table showing accuracy for the different morlet wavelet features.*

5 Accuracy score for STFT

In table 4.6 and 4.7 the accuracy values for each class can be seen for the STFT features. The classes are listed from 1 to 11 and correspond to the class values given in table 3.1. The accuracy is given for the number of frequency components and the different window sizes. At the bottom, the average of each class accuracy (macro averaged) is given as well as the total accuracy of the entire test set (micro averaged).

Class accuracy for test set (STFT)			
Class: (count)	500 freq. comps. 10 ms hanning window	500 freq. comps. 100 ms hanning window	500 freq. comps. 500 ms hanning window
Class 1 (10)	60 %	80%	20 %
Class 2 (40)	75 %	90 %	90 %
Class 3 (159)	66.04 %	76.73 %	62.26 %
Class 4 (7)	0 %	14.29 %	0 %
Class 5 (116)	26.72 %	55.17 %	61.21 %
Class 6 (44)	59.09 %	70.45 %	65.91%
Class 7 (9)	77.77 %	88.88 %	77.77 %
Class 8 (10)	20 %	10 %	10 %
Class 9 (53)	52.83 %	60.38 %	66.04 %
Class 10 (6)	16.66 %	0 %	16.66 %
Class 11 (21)	9.52 %	0 %	14.29 %
Avg. Class Acc.	42.15 %	49.63%	44.01 %
Accuracy:	50 %	63.79 %	59.79 %

Table 4.6: *Table showing accuracy for some different STFT features.*

Class accuracy for test set (STFT)		
Class: (count)	1000 freq. comps. 100 ms hanning window	1000 freq. comps. 500 ms hanning window
Class 1 (10)	50 %	30%
Class 2 (40)	92.5 %	70 %
Class 3 (159)	81.13 %	72.33 %
Class 4 (7)	0 %	0 %
Class 5 (116)	44.83 %	60.34 %
Class 6 (44)	68.18 %	59.09 %
Class 7 (9)	88.88 %	77.77 %
Class 8 (10)	10 %	10 %
Class 9 (53)	52.83 %	58.49 %
Class 10 (6)	0 %	16.66 %
Class 11 (21)	9.52 %	28.57 %
Avg. Class Acc.	45.26 %	43.93 %
Accuracy:	61.47 %	60.35 %

Table 4.7: *Table showing accuracy for the rest of the STFT features.*

6 Increased oscillations for morlet wavelet

This section is included to give a more in-depth view of what can be achieved with wavelets. The best performing mother wavelet was the morlet with 200 scales and a truncation factor of 500. It was therefore experimented more with this configuration. The increase in oscillation of the wavelet did not affect the extraction time and size of the data. Figure 4.4 display how the increase in oscillations affects the scores. The standard morlet is included as a reference. Except for the macro AUPRC, all the scores increase for increasing number of oscillations. It can be seen that the performance worsen from order 14 to 17.

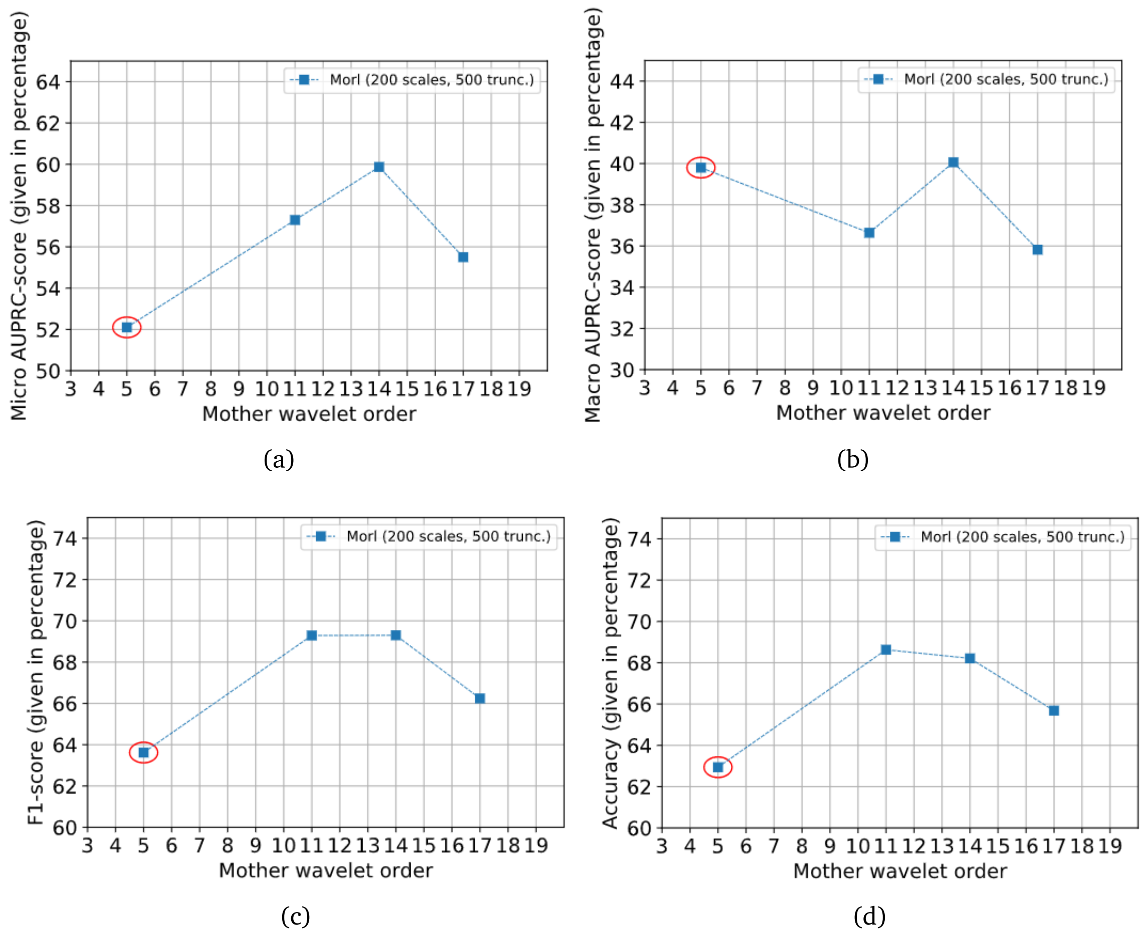


Figure 4.4: *Figure showing how four of the different scoring metrics where effected by the number of oscillations in the morlet. The original used mother wavelet is marked by a red circle.*

The exact metric scores for the increased oscillations can be seen in table 4.8 bellow. For the original morlet mother wavelet, see table 4.1.

Method	Scales	Time res. (after truncation)	Picture size	micro AUPRC	macro AUPRC	micro F1-score
CWT morl (11)	200	27.8ms	200x180	0.5730	0.3664	0.6929
CWT morl (14)	200	27.8ms	200x180	0.5987	0.4006	0.6930
CWT morl (17)	200	27.8ms	200x180	0.5550	0.3582	0.6624

Table 4.8: *Table showing the metric scores for increased oscillations in the morlet.*

The accuracy scores for each class and the total accuracy can be seen in table 4.9.

Class accuracy for test set Morlet increased oscillations			
Class: (count)	200 scales (11) 500 truncation	200 scales (14) 500 truncation	200 scales (17) 500 truncation
Class 1 (10)	40 %	50 %	20 %
Class 2 (40)	90 %	90 %	90 %
Class 3 (159)	69.81 %	64.15%	70.44%
Class 4 (7)	0 %	0 %	0 %
Class 5 (116)	69.83 %	75 %	56.03%
Class 6 (44)	93.18 %	90.91 %	90.91%
Class 7 (9)	88.88 %	77.77 %	77.77%
Class 8 (10)	20 %	40 %	50%
Class 9 (53)	64.15 %	64.15 %	67.92%
Class 10 (6)	16.66 %	16.66 %	33.33 %
Class 11 (21)	38.1 %	38.1%	33.33%
Avg. Class Acc.	53.69 %	55.16 %	53.61 %
Accuracy:	68.63 %	68.21 %	65.68%

Table 4.9: *Table showing the accuracy score for increased oscillations in the morlet.*

Chapter 5

Discussion

A discussion of the results, implementation and recommendations for further improvements are presented in this chapter.

1 Evaluation metrics

The main idea of including several classification metrics was to have as broad of an analysis as possible. Though the resulting metrics by them self did not end up as being very impressive, the comparison between them was the main focus and the most interesting part of it. As can be seen from table 4.1 using the right parameters for the feature extraction can heavily impact the performance. This does not only emphasis the difference between the STFT and the CWT, but also the difference between configuration parameters within each of the transforms as well.

Figure 4.1 displays the general decrease in performance when increasing the truncation factor for the scaleograms. This is due to the fact that truncation will lead to loss of information along the time axis. The truncation of time was however deemed necessary due to the increased extraction time and size of the final data set. Memory storage thus became an obstacle for performance.

Next, in figure 4.2, the impact of increasing the scales are presented. Looking at the F1-score (4.2c) a trend of increasing performance can be seen for morlet and gaussian(5), while the mexican hat performs best at 50 scales. This is likely due to the large bandwidth of the mexican hat which leads to overlap between the frequency response of the different scaled versions of the wavelets. Gaussian(5) and morlet has generally lower bandwidths than the mexican hat. This makes it so they can perform better at higher amounts of scales, since the frequency resolution can be exploited to its fullest. So where mexican hat should be used with low scales, the morlet and gaussian(5) can utilize more than 200 scales in the logarithmic frequency range 10 Hz-9 kHz.

The window size in STFT is important for deciding the relation between frequency and time resolution in spectograms. Figure 4.3 illustrates the results when using different window sizes. It is clearly seen that 10 ms performs the worst, while it is more difficult to separate if 100 ms or 500 ms is better. The reason why 10 ms windows performs worse is probably because it can not detect as low frequencies as the longer windows. This is a direct result related to the Gabor limit, and the maximum obtainable resolution i both domains.

Among all the results, it was the morlet of order 14 that performed the best with a micro F1 score of 0.6930, micro AUPRC of 0.5987 and macro AUPRC of 0.4006. As can be seen from figure 3.6 and 3.7 the high order morlets give the best frequency resolution, but as this resolution increases, it follows that the resolution in time decreases. This is as expected due to the smaller bandwidth for higher oscillation wavelets. From figure 3.1 it is shown that a lot of the classes contain lower frequency components (sub 100 Hz). A good frequency resolution in this area seems to be the key to distinguish some of these classes. Although there is somewhat lack of evidence, it could be argued that morlet(17) has too high frequency resolution to the point that time resolution is lacking. This is based on the fact that the result from morlet(17) decrease in performance compared to morlet(14) and therefore opposites the trend of higher oscillation wavelets performing better.

2 CWT vs. STFT

Some of the best results obtained with wavelets, e.g. the morlet with 200 scales and a truncation of 500, performed as good or even better for some metrics than the best ones obtained with STFT. An especially interesting result was the fact that the macro AUPRC score seemed to generally perform better for wavelets than STFT. This most likely came from the fact that the wavelet transform captured details in the low frequency area that the STFTs did not. As can be seen from the accuracy tables as well, the average class accuracy was higher for most of the wavelet features, especially for gaussian(5) and morlet. This is again an indicator that the wavelet features were more prominent or distinguishable over all the classes, even though the overall accuracy for the STFT features mostly outperformed the wavelets mexican hat, gaussian(5) and morlet. This can be seen from the high scores most of the STFT features obtained in class 3 and 5 of the test set. These two classes contains 57.9 % of all entries in the test set (and subsequently the same percentage of all entries in the total dataset). Even though the accuracy for these two classes were as good for the best performing wavelet features, the average class score was in most cases higher for wavelets. This indicates more distinguishable features between the classes.

A common trend observed was the fact that the two classes 4 and 10 were widely ignored by the network. These two classes are the two with the fewest members associated with them, respectively 7 and 6 in the test set. They are however not much smaller than classes 7 and 8, having respectively 8 and 10 members, which there were usually some degree of correct classification for. By looking at the different frequency spectrum associated with each class in figure 3.1 it can be seen that some classes share the same frequency components and that the CNN may have troubles distinguish these.

As the results indicate, the STFT feature with the 10 ms Hanning window performed the worst among the STFTs. This is to be expected due to lacking frequency resolution. As seen from the frequency spectra of the different classes, there are a lot of component in the lower frequency regions that seems to be crucial for being able to correctly classify some of the data.

The results for mexican hat, gaussian(5) and morlet did not indicate any huge advantages between using CWT against STFT for the data used in this setting. It is however worth noting that the originally extracted features of the CWTs, without the double frequency analysis, did in fact perform notably worse than for the STFTs. Most scores for the CWTs saw an increase in performance between 10 % and 20 % after the work-around with two sampling frequencies was applied.

3 Increasing wavelet oscillations

The amount of oscillations in the mother wavelet correlate to the order, n. The mexican hat is an example of a wavelet with a fixed amount of oscillations, and thus no order, while the morlet and gaussian derivative have unlimited amounts of orders. The morlet wavelet used in the first part of results correspond to an order of $n = 5$. This is the default configuration of the mother wavelet found when using the PyWavelets library, and it was therefore decided to use this in the first part of the results. As described in the theory, a higher order of the wavelet leads to more oscillations. When changing the amount of oscillations, the resolution cell is altered. With increasing oscillations the frequency resolution gets better and the time resolution gets worse.

The mother wavelets used for the main results of this thesis are the mexican hat, gaussian derivative of order 5 and the morlet of order 5. These are all wavelets of relatively low order and provide excellent time resolution while lacking some frequency resolution, especially for high frequencies. Due to the necessity of truncation in time when creating the scaleograms, the maximum obtainable resolution in time will always suffer. Since these fine details disappeared anyway, the focus on changing the resolution cell in favor of having better frequency resolution seemed to be a better solution.

It can also be argued that the mexican hat, gaussian(5) and morlet(5) have too little frequency resolution for high frequency occurrences. This can also clearly be seen from figure 2.9 in the theory. As an example the mexican hat has ∼4 kHz for a center frequency of 4 kHz, which covers a span from 2 kHz to 6 kHz. Gaussian(5) and the morlet implementations have substantially lower bandwidth for the same center frequency, with morlet(5) having ∼1.5 kHz.

The idea of using more oscillations in the mother wavelet can be backed up by the necessary truncation and lacking frequency resolution in high bins. This can also be seen very clearly by the increase of performance when using a higher number of oscillations for the morlet. The extra tests with morlet of order 11, 14 and 17, gave an increase in performance compared to the original with order 5. By observing figure 4.4, that shows the performance of the increased oscillations, it can be seen that the results start to worsen with order 17. It is therefore fair to assume that the ideal order for the morlet, for the audio data used, lies around 11 to 14 oscillations. These results are included as an extra section, rather than as a part of the main results. It is unfair to conclude that the wavelets outperform the STFT purely due to the fact that far more work were put into optimizing the wavelet features. There can however be seen a clear potential for correct use of CWT with this data.

4 Distribution of scales

One early assumption for this thesis was to distribute frequencies in a logarithmic manner from 10 Hz to 9 kHz and then map this distribution to corresponding scales. This method does not ensure a perfectly spaced distribution, but rather makes an approximate. It can be seen that when increasing the amount of scales the major visual changes in the resulting scaleograms are in the low frequency bins. This may indicate that the scales are distributed in a way that the bandwidth have more overlap for high frequencies than for low.

A better way of calculating the distribution of scales could be to investigate each scaled wavelet separately and find its bandwidth. The -3dB point will then serve as the start for the next scaled wavelet. This would make sure that the overlap between scales is consistent and one would not have to experiment with the amount of scales. A disadvantage would however be that the flexibility of varying scales for the sake of limiting data size.

5 Thoughts on the low frequency issue in scaleograms

The reason for using two different sampling frequencies when creating the scaleograms was, as mentioned, to work around the fact that unexpected low-frequency components occur. Visually, this way of making the scaleograms has huge improvements on high frequency sound files.

A speculation on why the phenomena of these unexpected low frequency components occurred is that the frequency response for high-scale (low frequency) wavelets contain non-zero magnitude for high frequencies. This would have caused high-scale wavelets to trigger at high frequencies.

To achieve correct and fair results it can be argued that one should perform the STFT analysis with two sampling frequencies as well. Although this might be good practice, it would not have mattered as STFT did not suffer from the same weaknesses as the CWT in PyWavelets did.

6 Dataset and extraction time

A real-world dataset created for the use in machine learning was used during this thesis. As with most applications meant for use in the real world there will not be perfect results. The entries in each class are unbalanced and some pre-processing were needed as described in the chapter "Experimental setup". As a result of the highly imbalanced classes and the deviation within each class, it can be argued that such data is not suited for a CNN so simple as the one designed. Even if this was the case, the comparison between CWT and STFT has been shown - with effect.

The features extracted for the different wavelets and the different STFTs ended up taking a variable amount of time, as expected. The extraction time was heavily dependant on the final data size and specially the amount of scales in the case of wavelets. An important notion when considering the performance of the different features is the amount of time spent on extraction and the final size of the data. A minor improvement in performance at the cost of memory storage may not always be feasible when working with limited resources. Time and memory space were limiting factors during the making of this thesis. In table 5.1 the final data size of each feature set can be seen. The extraction time was for the most part proportional to the data size, but a tendency of increased extraction time was seen with scales of more than 100 for wavelets. The size of the data was the same for each mother wavelet as expected.

Method	Main features	Array size	Data size
STFT	1000 freq. comps. 500 ms, 50 % overlap	1000x18	419 MB
STFT	500 freq. comps. 500 ms, 50 % overlap	500x18	210 MB
STFT	1000 freq. comps. 100 ms, 50 % overlap	1000x98	2.24 GB
STFT	500 freq. comps. 100 ms, 50 % overlap	500x98	1.11 GB
STFT	500 freq. comps. 10 ms, 50 % overlap	500x998	11.2 GB
CWT	30 scales, 100 trunc.	30x900	2.34 GB
CWT	50 scales, 100 trunc.	50x900	3.9 GB
CWT	50 scales, 500 trunc.	50x180	802 MB
CWT	200 scales, 500 trunc.	200x180	3.15 GB

Table 5.1: *Table showing the total data size of all features extracted from the 2960 sound files.*

7 Further work

As stated multiple times throughout this thesis, the focus has not been on optimizing the neural network, but rather comparing CWT with STFT. It is therefore natural to assume that the final scores and metrics obtained are not what optimally could be achieved. It would be interesting for further work to experiment more with the CNN configurations and different network architectures. There may be configurations that fit the data better than what was eventually used.

Even though wavelet transforms have been utilized in several fields over the last 50 years, the popularity seems to have been somewhat lacking outside the academia and in specialized cases. The resurgence of interest in wavelets during the later years have lead to new and interesting discoveries and prepositions for hybrid transforms. One such suggestion, that was encountered at the end of writing this thesis, is the superlet[20] proposed by Dr. Raul C. Mureşan and his team in 2019. The idea of the superlet comes from concepts in optical super-resolution. The process is based on having several sets of wavelets with increasing bandwidth that are to be combined geometrically into a so called superlets. This will allegedly maintain the good time resolution of wavelets for higher frequencies and simultaneously gain better frequency resolution in the upper bands than what is possible with a standard wavelet transform. Their research have shown that superlets outperform STFTs, CWTs, and other super-resolution methods on synthetic data and brain signals recorded in humans and rodents. The work done indicates that even though each single wavelet estimate follows the Gabor limit, the combination of multiple wavelets can be done in post-processing in a way so to increase the time-frequency resolution. Doing further work with such wavelet combinations would be highly interesting. If the superlets had been encountered earlier in the work process, it would most likely have been the focus of the thesis. It is however, due to limited time, only included as an idea for continued work.

For later analysis it would also be interesting to try another wavelet library to see if others perform better, or try to implement the transform manually.

Chapter 6

Conclusion

Throughout this thesis it has become clear that the continuous wavelet transform, with different mother wavelets, does not directly outperform the Short-Time Fourier Transform. However, gaussian(5) and morlet if not outperforms, perform equally as good as the STFT implementation for certain metrics. The extra testings with increased oscillations in the morlet wavelet performs even better than the STFT and the other wavelet implementations. For a more convincing comparison between the transforms, more STFT features should have been experimented with to strengthen the study. The CWTs seem to generally score better on macro averaged values (e.g AUPRC and accuracy). This seems to indicate that the CNN detects different patterns from the scaleograms and spectrograms.

The results implies that the amount of scales should depend on the mother wavelet used in the transform. The mexican hat performs best at low amount of scales (around 50 scales), while morlet and gaussian(5) performs better at higher scales (around 200). This indicates that the frequency response of the scaled wavelets should overlap in a favorable manner. Therefore it can be argued that the distribution of scales could be found more thoroughly by calculating the bandwidth of the wavelet at each scale and make sure that the bandwidths overlap in a certain way. This would have given a more efficient distribution of scales and also reduced the amount of variables for testing. It can be argued that the wavelets should have been implemented manually and not used from the Python library PyWavelets. This could have prevented having to work around the problem of false low frequency components.

The fact that the truncating of time was deemed necessary made the need for very good time resolution obsolete. This made it obvious that wavelet families with better time resolution were not too effective. This can be seen from the performance of the mexican hat wavelet. An increase of oscillations in the mother wavelet, as tested for the morlet, gave visibly better frequency resolution. This seems to be a key factor in achieving the best performance for this specific case.

The research has shown the usefulness of wavelets as compared to what is considered more traditional frequency-time analysis tools such as the STFT. There is still more to discover as of the best ways to apply the wavelet transform, and there is more to be done for further work.

Bibliography

[1] R. K. Young. "*Wavelet theory and its applications*". Springer, 1993.

[2] J. Qin and P. Sun. "*Applications and Comparison of Continuous Wavelets Transforms on Analysis of A-wave Impulse Noise*". *Archives of Acoustics 40, No. 4*, pages 503–512, 2015. doi: 10.1515/aoa-2015-0050.

[3] R. Polikar. "*The Story of Wavelets*".

[4] I. Daubechies. "*Orthonormal Bases of Compactly Supported Wavelets*". 1988.

[5] P.S. Addison. "*Introduction to redundancy rules: the continuous wavelet transform comes of age*". `https://royalsocietypublishing.org/doi/10.1098/rsta.2017.0258`. Last accessed:19/04-2020.

[6] C. Torrence and G. P. Compo. "*A Practical Guide to Wavelet Analysis*". `https://atoc.colorado.edu/research/wavelets/bams_79_01_0061.pdf`. Last accessed:02/06-2020.

[7] F. Chic. "*Definition of the Neurochemical Patterns of Human Head Injury in1H MRS Using Wavelet Analysis*". `https://cds.ismrm.org/ismrm-2001/PDF3/0822.pdf`. Last accessed: 14/05-2020.

[8] The PyWavelets Developers. "*Continuous Wavelet Transform (CWT)*". `https://pywavelets.readthedocs.io/en/latest/ref/cwt.html`. Last accessed:21/03-2020.

[9] C. Hurley and J. Mclean. "*Wavelet Analysis and Methods*". Scientific e-Resources, 2018.

[10] S. Mallet. "*A Wavelet Tour of Signal Processing, third edition*". 2009.

[11] A.H. Najmi and J. Sadowsky. "*The Continuous Wavelet Transform and Variable Resolution Time–Frequency Analysis*". *JOHNS HOPKINS APL TECHNICAL DIGEST*, page 134–140, 1997.

[12] W.S. McCulloch and W. Pitts. "*A logical calculus of the ideas immanent in nervous activity*". *Bulletin of Mathematical Biophysics*, pages 115–133, 1943. doi: 10.1007/BF02478259.

[13] K. Fukushima. "*Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position*". *Biological Cybernetics*, pages 193–202, 1980. doi: 10.1007/BF00344251.

[14] T.N. Wiesel D.H. Hubel. "*Receptive fields and functional architecture of monkey striate cortex*". *The Journal of Physiology*, page 215–243, 1968. doi: 10.1113/jphysiol.1968.sp008455.

[15] A. Khosla F. Yu L. Zhang X. Tang Z. Wu, S. Song and J. Xiao. "*3D ShapeNets: A Deep Representation for Volumetric Shapes*". CVPR, 2015.

[16] Y. Bengio I. Goodfellow and A. Courville. "*Deep Learning*". 2016.

[17] S. Sharma. "*Activation Functions in Neural Networks*". `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`. Last accessed: 14/04-2020.

[18] J. Vanhanen C. Lassenius, T. Soininen. "*Constructive Research*". `http://www.pm.lth.se/fileadmin/_migrated/content_uploads/3._constructive_research.pdf`. Last accessed: 24/05-2020.

[19] R. Pujol. "*Human auditory range*". `http://www.cochlea.org/en/hear/human-auditory-range`. Last accessed:12/06-2020.

[20] V. Moca, A. Nagy-Dabacan, H. Barzan, and R. Muresan. "*Superlets: time-frequency super-resolution using wavelet sets*". 03 2019. doi: 10.1101/583732.

Appendix A

The data used during this thesis is the property of Norsonic and SINTEF Digital. It is therefore confidential and can not be given out to anyone wanting to reproduce these results. The code used is however made by Håkon S. Kvalnes and Magnus S. Lysø, and can be shared freely. The code is written using different libraries in Python and can be obtained by contacting:

- haakon959@gmail.com
- magnusslyso@gmail.com