Master

Background Noise Classification and Denoising of Audio Signals utilizing Empirical Wavelet Transform and Deep Learning

ANTON GARBAR

SUPERVISOR
Linga Reddy Cenkeramaddi

University of Agder, 2021
Faculty of Engineering and Science
Department of Engineering and Sciences

Acknowledgments

I would like to thank Associate Professor Linga Reddy Cenkeramaddi for the guidance and support throughout the thesis. In addition, I would like to thank Rajesh Reddy Yakkati, for hints and help with the development of my first CNN. Lastly, I would also like to thank Sreenivas Reddy Yeduri for proofreading and technical help with finishing the compilation of the report.

Abstract

The importance of sound and speech in human life cannot be overstated. Ambient sounds inform us about our surroundings and warn us of potential dangers, such as a car approaching from behind. One of the most common modes of communication is speech. However, if it is contaminated with background noise, it may result in data loss. Recent advances in artificial intelligence enable machines to recognize and classify sound patterns, as well as remove complex background noise from contaminated speech.

This thesis investigates a method for improving existing background noise classification and denoising solutions. This is accomplished through signal decomposition with the Empirical Wavelet Transform and subsequent processing with the Convolutional Neural Network. Improvements of up to 18% have been observed.

Contents

Chapter 1

Introduction

1.1 Background

The advancements in artificial intelligence have completely transformed signal processing approaches. The use of machine learning in speech signal processing is rapidly expanding. The audio signals contain both human speech and background noise. The background sounds help us identify and gather information about the speaker's surroundings. Such knowledge can be extremely useful at times and has numerous applications in real-world situations.

Signals from secondary sources are always added to the signals from human speech. In general, speech signals can be used to obtain speaker information. Whereas secondary source information can be used to obtain surrounding information. Secondary signal processing aids the forensic department in tracking the speaker's activity. Thus, environmental sound classification (ESC) has a wide range of applications, including smart homes, animal sound classification, and environmental understanding. Existing classification techniques, such as support vector machine (SVM) and Gaussian Mixer Model (GMM), are inefficient for ESC because they are incapable of extracting more accurate and comprehensive features [1].

1.2 State-of-the-art

A heterogeneous system of Deep Mixture of Experts (DMoEs) has been proposed for classifying the acoustic scenes using convolutional neural networks (CNNs) [2]. Here, each DMoE is a mixture of different convolutional layers weighted by a gating network [2]. A novel CS-LBlock-Pat. model has been proposed for an efficient floor tracking of the speaker in a multi-storey building [3]. The proposed model was tested on an environmental sound classification (ESC) dataset collected from a ten-floor multi-story hospital. Finally, SVM was applied to the dataset to achieve an accuracy of 95.38% [3]. For audio event classification, the authors propose using nonlinear time normalization-based event representation prior to mid-term statistics extraction [4]. Following that, in order to reduce errors in the presence of noise, these short-term features are represented as a constant uniform distance sampling over a defined space [4]. In [5], the authors have designed 150 different CNN-based models for the ESC and test their accuracy on the Urbansound8k ESC dataset. It has been shown in [5] that the CNN model has achieved an accuracy of 82.5%, which is higher than its classical counterpart. A light weight CNN method for automatic heart sound classification has been proposed in [6]. This method needs a simple preprocessing on the heart sound data and then the time-frequency features are extracted based on the heart sound data. Finally, the heart sounds are classified based on the fusion features [6]. In [7], the authors have proposed a model that is based on an attention-enhanced DenseNet-BiLSTM network, and segment-based linear filter bank (LFB) features for detecting spoofing attacks in automatic speaker verification. Initially, the silent segments have been obtained from each speech signal using a short-term zero-crossing rate and energy. Then, the LFB features are obtained from the segments. Finally, attention-enhanced DenseNet-BiLSTM architecture has been built to mitigate the problem of overfitting [7]. The classical methods for integrating Mel-frequency cepstral coefficients (MFCCs) and the audio signal information in the temporal domain may lead to the loss of essential information [8]. Thus, in [8], the authors have adopted a tool named local binary pattern (LBP) to characterize the latent information on the temporal dynamics. Then, this LBP is used to encode the evolution process by considering the frame-level MFCC features as 2D images. Finally, the obtained features are fed to the d3C classifier for ESC [8].

In [9], the authors have coupled the spatio-temporal attention pooling layer with the convolutional recurrent neural network for acoustic scene classification to learn about patterns that are discriminative while suppressing the irrelevant patterns. Afterward, the final feature vector has been formed from the outer product of spatial and temporal attention vectors [9]. A standard hybrid model-based framework has been proposed in [10] to learn the representations for environmental audio scenes (EASs) and sound events (SEs). Then, the instance-specific adapted Gaussian mixture model and hidden Markov models have been explored for the EASs and SEs, respectively, to compute discriminatory representations. Finally, a discriminative model-based classifier has been used to identify EASs and SEs [10]. In [11], the authors have investigated the use of wavelet transform-based Mel-scaled features for acoustic scene classification. It has been concluded that the proposed features have shown better discriminative properties than other spectral features even with the same classification framework [11]. In [12], the authors have investigated the use of unsupervised and supervised Non-negative matrix factorization (NMF) for feature extraction. Then, the features are trained with deep neural networks for an efficient ESC [12]. A sensor-actuator system for surface identification and classification has been proposed in [13]. The proposed system contacts the surface to be identified by generating a mechanical impact and then it analyzes the intrinsic characteristics of the surface based on the resulted sound from the surface. Finally, a deep learning based machine learning pipeline has been utilized to identify the everyday surface sounds [13]. In smart cities, the surveillance of hazardous sounds such as gunshots, explosions, and etc., need to be developed for better monitoring. In [14], the authors have explored several classification methods over the SESA dataset for sound event recognition task and showed that SDG classifier results in higher accuracy compared to others. The existing classification algorithms that are based on CNNs are inefficient [15]. Thus, a capsule network has been introduced in [15] to detect the fake audios. After that, the dynamic routing algorithm has been explored to pay more attention to the capsule network for a better identification of audio spoofs [15]. A framework has been proposed in [16] for determining the user location from the recorded signals of the users' device. In [16], the authors have presented two algorithms namely SoundSignature and SoundSimilarity for user location identification. Here, SoundSignature has been utilized to extract the acoustic fingerprints from the recorded audio. Then, a audio similarity measure has been employed with SoundSignature to detect the user based on the acoustic signals from nearby users or microphones. An accurate and reliable identification of the first (S1) and second (S2) sounds of the phonocardiogram in the presence of S3 and S4, high-pitched sounds, murmurs, physiological interference, and other environmental noises has been proposed in [17, 18].

In [19], the authors have proposed three types of audio-visual deep neural networks (AVNs) such as feature level AVN (AVN-F), embedding level AVN (AVN-E), and joint learning AVN (AVN-J) for the verification of a speaker. In order to further enhance the robustness of speaker verification, the respective data augmentation methods have been proposed [19]. A combinatory feature-based stack autoencoder has been proposed in [20] for fundamental heart sound classification and experiments are conducted on both public datasets and recorded heart sounds. The usage of Gated Recurrent CNN for the identification of genuine/spoofed audio classification has been investigated in [21]. In order to enhance the robustness of the speech in the noise environment, the authors have considered the usage of signal-to-noise masks as new input features to inform about the input spectral features that are affected by noise to the anti-spoofing system [21].

In [22], the authors have explored the utilization of spatial information obtained from diverse spatial receivers such as wireless acoustic sensor networks for acoustic weapon classification. Further, maximum likelihood estimation based on fused data has been utilized for improving the classification accuracy [22]. A deep rational attention network (DRANet) has been proposed in [23] to guarantee the performance of a deep learning-based intelligent diagnosis method under strong method. The stability analysis of deep convolutional neural networks, long short-term memory, and vanilla neural networks has been investigated in [24] with the training of raw wave front-ends for automatic speaker verification. Then, a joint convolutional LSTM neural network has been proposed that outperforms the other state-of-the-art mechanisms. Further, a CNN-based raw waveform (RW-CNN) end-to-end computational scheme has been proposed in [25] for speaker identification with noise and reverberation data augmentation. Moreover, the authors have compared the performance of the proposed scheme with the MFCCs features and showed that RW-CNN outperforms even in adverse conditions [25].

Optimum allocation sampling (OAS)-based empirical mode method (EMD) has been proposed in [26] for automatic ESC. The method reduces the long length sound signals to homogeneous signals which are then decomposed into intrinsic mode functions (IMFs). These IMFs are fed into classifiers that use a multi-class least squares support vector machine (MC-LS-SVM) and an extreme learning machine (ELM) to assess the performance of the proposed mechanism. It has been concluded that the MC-LS-SVM and ELM results in an accuracy of 87.25% and 77.61%, respectively [26]. The environmental sound classification (ESC) has been assessed with conventional neural network in [27]. A deep neural network with two convolutional layers such as max-pooling layer and fully connected layers was trained on low-level data [27]. *Throughout the manunscript, the terms background sound classification and environmental sound classification are used interchangeable.* In [28], a 1-dimentional (1-D) CNN based end-to-end approach has been proposed for ESC which learns the representation directly from the audio signals. It has been concluded that the proposed system works with any input sizes [28]. A CNN-based ESC has been proposed in [29], which expresses the features of the ESC in Red-Green-Blue (RGB) image format. Following that, CNN is trained to improve ESC accuracy. A dilated CNN based ESC method has been proposed in [30] in order to improve the accuracy of ESC. Further, the performance of the proposed system with variation of dilation rate and number of layers of dilated convolution has been evaluated. It has been observed that large values of dilation rate and dilated convolution degrades the accuracy of the proposed system [30]. A robust filtering algorithm has been proposed in [31] in order to effectively suppress the noise and interference of the audio forensics that have been collected during questioned session. The proposed mechanism in [31] first encodes the time domain expression of the combined signal as the instantaneous frequencies of an analytical sinusoidal frequency modulated signal. Then, a sinusoidal time frequency distribution (STFD) has been generated using kernel function. The peaks of STFD corresponds to the intermediate frequencies of the denoised signal [31]. Band energy difference (BED) descriptor, a feature set, has been proposed in [32], for source attribution, i.e., a device from which the audio has been recorded. It has been observed in [32] that a power discrimination occurs at each frequency based on the device that is being used for the recording. The descriptor works in two phases such as device identification followed by device verification [32].

To address data scarcity issue, a deep CNN-based classification method with data augmentation has been proposed in [33]. It has been concluded in [33] that the CNN model with data augmentation performs better than both CNN without data augmentation and “shallow” dictionary learning method with the considered data augmentation. Finally, it has been determined that the use of class conditional data augmentation performs better than other classes of augmentations [33]. In addition, a deep CNN framework has been proposed in [34] for ESC with various augmentation techniques. Moreover, the authors in [34] introduced a new data augmentaion method based on Linear prediction cepstral coefficients (LPCC). The increased size of the dataset requires powerful GPUs to process it. Thus, the size of the dataset plays a vital role in the system configuration. In [35], a light weight dilated CNN (LD-CNN) classification method has been proposed to offer the comparable performance in comparison to the dilated CNN classification method with reduced dataset size. The proposed work lies on two-fold. Initially, the parameters have been reduced by filtering a two dimensional convolutional ($L \times W$) filters to two separable one dimensional convolutional filters ($L \times 1$ and $W \times 1$). Then, the first fully connection layer has been replaced by a feature sum layer in order to reduce the number of parameters. It has been concluded in [35] that the proposed classification model reduced the size of the dataset by 50 times with a loss of 1% to 2% accuracy.

1.3 Research directions

Based on the background and state-of-the art, the following research directions are addressed in this thesis.

1. Background classification in the human speech using time-frequency analysis and deep CNN.
2. Denoising the speech signals using time-frequency analysis and deep CNN.

Chapter 2

Background theory

This chapter's goal is to provide the theoretical foundation for this thesis. A brief explanation and mathematical background are provided. This chapter is primarily concerned with signal processing. It is assumed (based on prior experience) that readers are more familiar with Machine Learning than signal processing.

2.1 FFT

The Fourier Transform is a useful tool in the signal processing toolbox. It is, to be more precise, a discrete variant for this. The Discrete Fourier Transform (DFT) is a mathematical process that determines the harmonic, or frequency, content of a discrete signal sequence [36, p.59]. On the other hand, FFT stands for Fast Fourier Transform, and it's an effective method to run DFT.

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nm}{N}} \tag{2.1}$$

Equation (2.1) is DFT, and it is easy to understand and follow. However, the whole operation is simply summation and multiplication, which results in complex output.

Where:

x(n) - is sampled input data, with index n = 0, 1, 2, 3, ... N-1.

X(m) - is DFT output in form of frequency component, with index m = 0, 1, 2, 3, ... N-1.

X(m) - is DFT output in form of frequency component, with index m = 0, 1, 2, 3, ... N-1.

DFT output X(m) is complex in the form of $a \pm jb$. To get the magnitude and phase angle of the frequency component, two additional operations are performed.

$$|Xmag| = \sqrt{a^2 + b^2} \tag{2.2}$$

$$Xphase = \tan^{-1} \frac{b}{a} \tag{2.3}$$

Frequency associated with frequency component of X(m) computed by equation (2.4).

$$f(m) = m \frac{Sample frequency}{N} \tag{2.4}$$

2.1.1 FFT example

The figure (2.1) shows a cosine function with a 5Hz frequency and 45 deg phase angle. This is a simple test function to show signal processing possibilities with help of FFT.

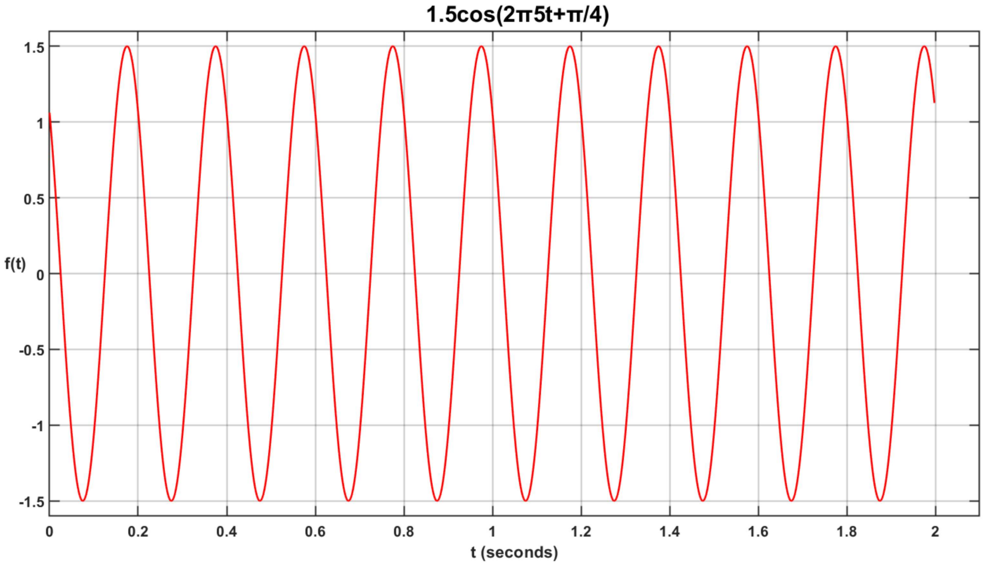


Figure 2.1: Graph of a simple test function, used for FFT display

Let's assume we know nothing about this test function, and it's given to us as an array of real numbers. Processing such array through equation (2.1) results in a new array, but this time in form of complex numbers. Running these complex numbers through equations (2.2) and (2.3) gives us frequency, amplitude and phase of a test signal.

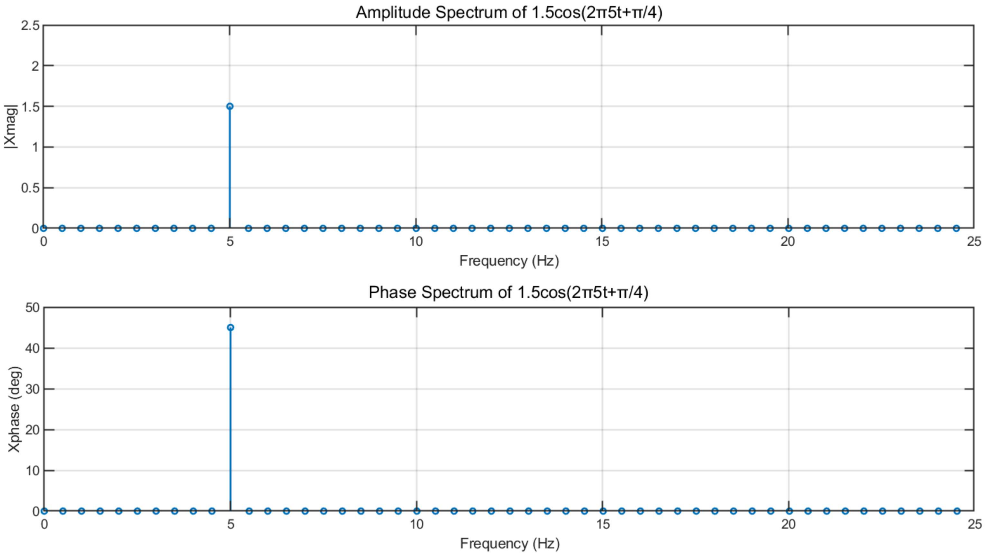


Figure 2.2: Amplitude, frequency and phase of a test signal

2.2 STFT

The Fourier transform of the signal cannot depict how the spectral content of the signal changes with time, which is critical in many non-stationary signals in practice [37]. That means Fourier

Transform does not display when the set of frequencies occurs. Instead, it provides the frequency information averaged over the entire signal time interval. Short-time Fourier transform (STFT) is a sequence of Fourier transforms of a windowed signal. STFT provides the time-localized frequency information for situations in which frequency components of a signal vary over time [38, p.176].

Windowed signal means that signal or part of the signal (frame) was multiplied by the window function. Often used window functions are Hamming and Hanning. However, by the nature of these functions, the signal will be attenuated at the edges. To prevent that attenuation, the overlap-add technique is used.

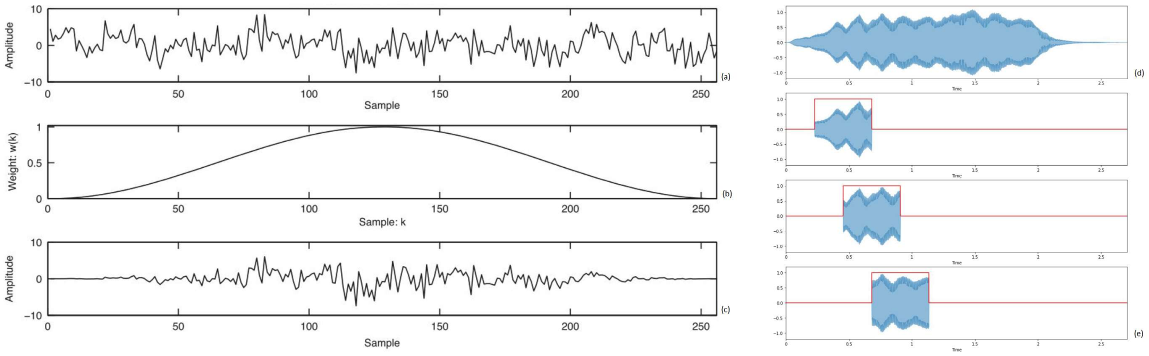


Figure 2.3: Example of windowed signal and overlap-add technique: (a) Signal before multiplying by the window function. (b) Hanning window function. (c) Product of signal and Hanning window function. (d) Non-windowed signal. (e) 50% overlap-add technique [39] [40].

Equation (2.5) depicts STFT, and it works by splitting the signal into time-frames by using the window function, then DFT is performed on each frame Figure 2.4. This results in a time-localized frequency spectrum. To visualize how frequencies change over time for the whole duration of the signal, spectrograms are used.

$$X_m(f) = \sum_{n=-\infty}^{\infty} x(n)g(n-mR)e^{-j2\pi fn} \tag{2.5}$$

Where [41]:

g(n) - window function of length M

$X_m(f)$ - DFT of windowed data centered about time mR

R - Hop size between successive DFTs. The hop size is the difference between the window length M and the overlap length L.

The time requirements decide the length of the window function. Often in speech-related analysis, we choose a 20ms time frame with at least 50% overlap. It allows for sampling of two periods of 100Hz fundamental frequency, which leads to better differentiation between speech sounds (vowels, diphthongs, and consonants). Nevertheless, there is a correlation between frame size and frequency resolution.

Gabor-Heisenberg uncertainty principle (2.6) states that by decreasing time resolution, frequency resolution increases and vice versa. Where Δf is "effective frequency width" and Δt is "the effective duration" of the analyzed signal [42].

$$\Delta f \Delta t \geq 0.5 \tag{2.6}$$

Increasing Δt (wider time-frames) degrades time resolution, but that leads to smaller Δf, which means narrower bandwidth. As a result, the certainty regarding the frequency content of the frame improves.

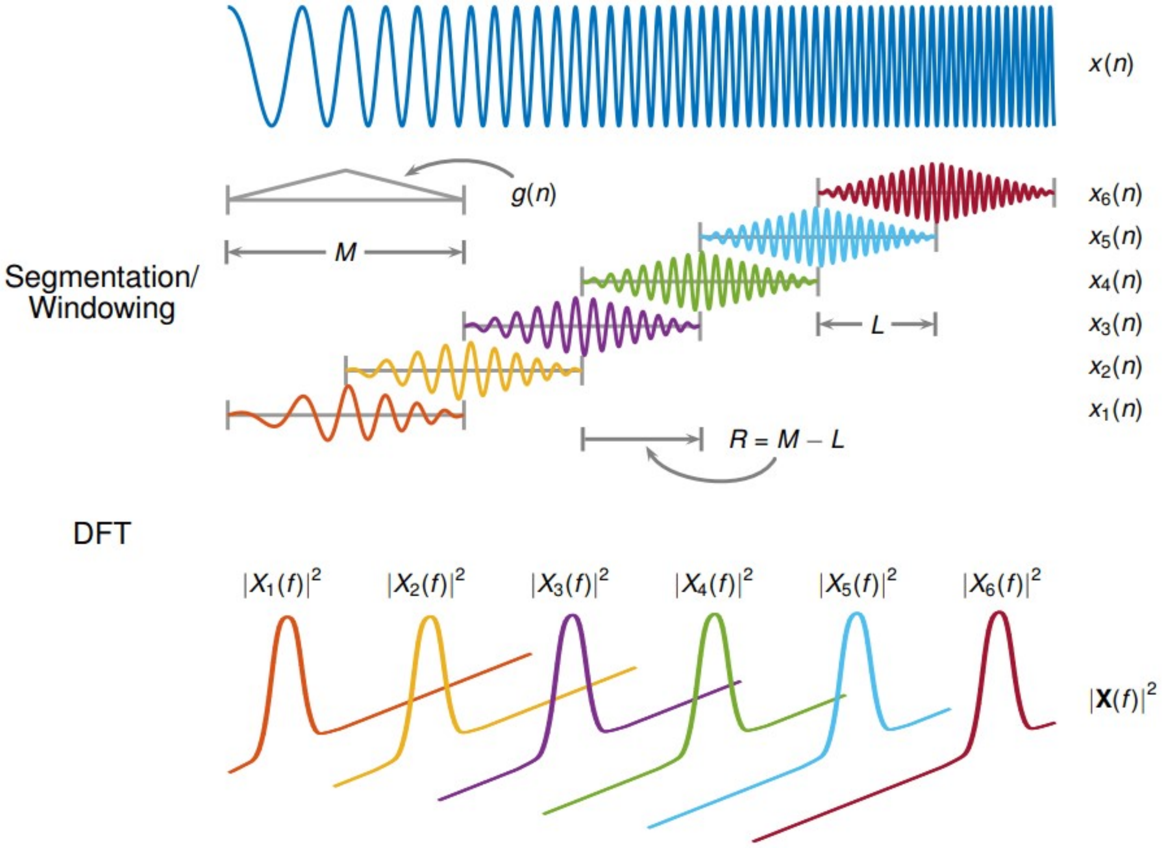


Figure 2.4: STFT overlap-add windowing [41]

2.2.1 FFT and STFT comparison

This subsection shows the practical limitations of FT in analyzing non-stationary signals and how to overcome such limitations with STFT. However, Gabor-Heisenberg uncertainty principle Eq. (2.6) plays a big role in deciding time resolution, and we will see how it affects signal analysis by STFT.

Experiments are performed with a linear sweep signal (2.7) (also known as chirp). This type of signal has linearly increased or decreased frequencies over a period of time.

$$y = c\sin\{[\frac{\pi}{b-a}[(\frac{(b-a)x}{d} + a)^2 - a^2]\} \tag{2.7}$$

The start and stop frequencies (Hz) are given by a and b, respectively. The length of the signal over which the frequency varies between a and b is given by d (seconds) [43, p.171].

The following is the formal definition of a non-stationary signal: A non-stationary stochastic signal is one whose statistical structure changes as time passes, for example, the mean, variance, correlation (covariance), and so on. Doppler-shifted sound, Bird sound, vibration response of machinery undergoing operational changes, speech, noise and vibration signals from accelerating traffic, chirp signals, the impulse response of a damped non-linear Duffing oscillator, trajectories of a Lorenz system, and velocity of an impacting oscillator are a few examples of such signals [44].

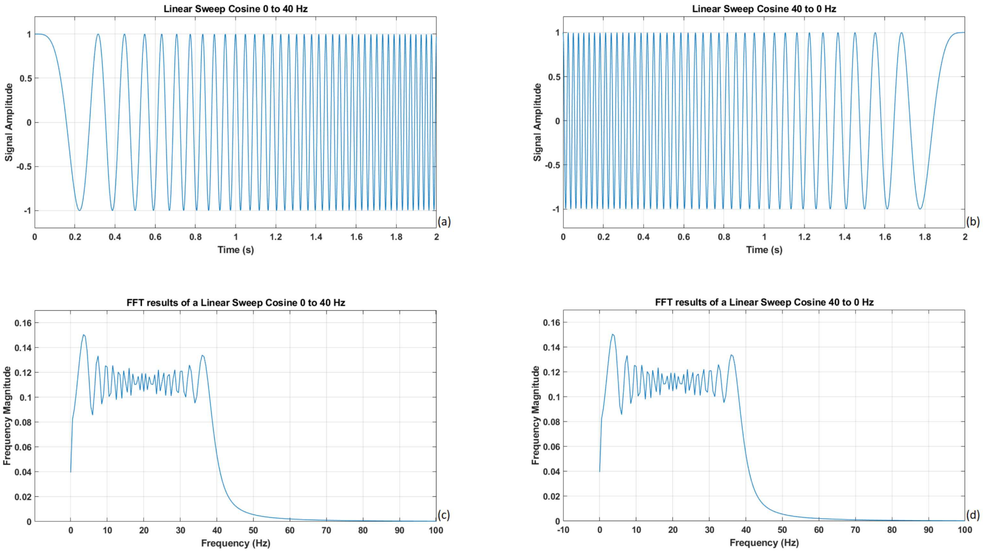


Figure 2.5: Fourier Transform of non-stationary signal: (a) Chirp frequency increase. (b) Chirp frequency decreases. (c) FFT results of Chirp increasing frequency. (d) FFT results of Chirp decreasing frequency.

The main limitation of performing FT on a non-stationary signal is that it does not show how the spectral content of the signal changes with time. A simple chirp signal is used to visualize this limitation Figure (2.5). Signals duration is 2 seconds with a bandwidth of 40 Hz. In Figure 2.5(a) signal starts at 0 Hz and at the time of 1 second, its frequency is 20 Hz, at the end of signal duration frequency is 40 Hz. The same signal is shown in Figure 2.5(b), only this time it is inverted. The results of performing FFT on these signals are identical. It visualizes that Fourier Transform may not be the best choice to analyze how signals are changing with time.

Now same chirp signal is analyzed by STFT and visualized in the form of a spectrogram Figure 2.6. The window function is Hanning. Overlap is 95%, high overlap produces smoother spectrogram, but often is a waste of hardware resources in real analysis applications. Spectrograms show that STFT can tell that frequencies are increasing or decreasing with time. The quality of the spectrogram and analysis depends on chosen time resolution. Spectrograms with 500ms time frames correctly show that frequency bandwidth is 40 Hz, whereas smaller time frames degrade frequency resolution. This is the trade-off between time and frequency.

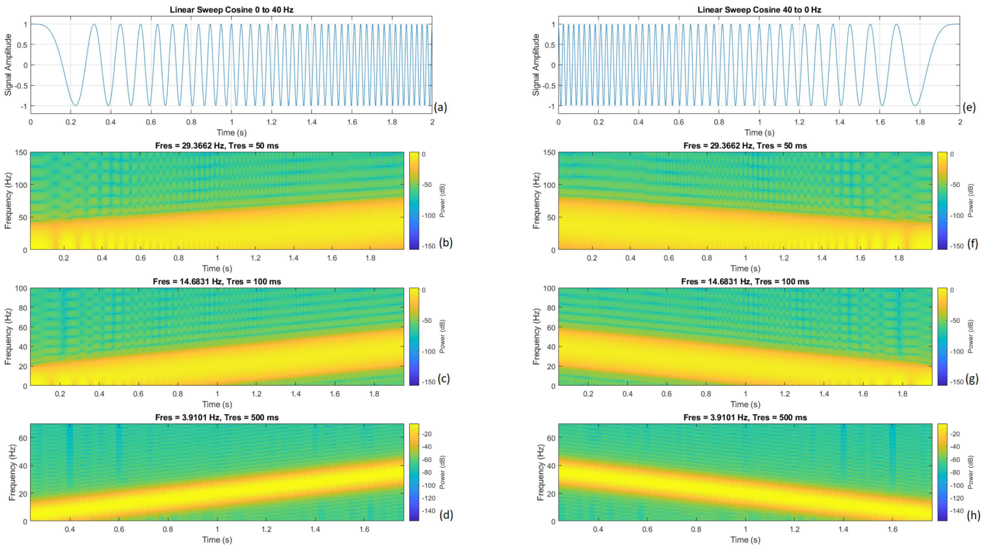


Figure 2.6: STFT spectrograms of non-stationary signal: (a) Chirp frequency increase. (b) Spectrogram of (a) time-res 50ms. (c) Spectrogram of (a) time-res 100ms. (d) Spectrogram of (a) time-res 500ms. (e) Chirp frequency increase. (f) Spectrogram of (e) time-res 50ms. (g) Spectrogram of (e) time-res 100ms. (h) Spectrogram of (e) time-res 500ms.

2.3 Wavelets and Wavelet Transform

Wavelets are mathematical functions that divide data into distinct frequency components and then examine each component with a resolution proportional to it's scale. They outperform classic Fourier methods in assessing physical circumstances with discontinuities and abrupt spikes in the signal. Fields such as astronomy, acoustics and speech, nuclear engineering, sub-band coding, image processing, neurophysiology, music, magnetic resonance imaging (MRI), speech discrimination, optics, fractals, turbulence, earthquake prediction, radar signal processing, computer vision, and pure mathematics applications such as solving partial differential equations are all using wavelets [45].

We start our explanation of wavelets and Wavelet Transform with a reminder of the Fourier Series. Idea is that, under certain conditions, we may approximate signal or function as a sum of other functions. For Fourier Series, signal $f(t)$ has to be periodic and have finite energy over one period T:

$$\int_0^T |f(t)|^2 \, dt < \infty \tag{2.8}$$

Then any such signal may be approximated by using superposition of base function $e^{jwt} = e^{j\frac{2\pi}{T}t} = cos(wt) + jsin(wt)$. Composition of sines and cosines with different frequencies and amplitudes approximates to $f(t)$ (higher number of sines and cosines improve approximation). It is known as an exponential form of Fourier Series:

$$f(t) = \sum_{-\infty}^{\infty} c_n e^{j\frac{2\pi}{T}nt} \quad n = ..., -1, 0, 1, ... \tag{2.9}$$

Now let remind ourself how Fourier Transform defined:

$$\hat{f}(w) = \int_{-\infty}^{\infty} f(t) e^{-jwt} \, dt \tag{2.10}$$

Both Fourier Transform and Series are using sine and cosine as their base functions. And since $sin(nx)$ and $cos(nx)$ complete n cycles (harmonics) every 2π radians, it is reasonable to generalise that function $f(t)$ lays on interval $[0, 2\pi]$. Then eq.(2.8) upper limit changes to 2π and function $f(t)$ repeats itself every 2π radians.

Almost all of the signals that are of practical interest are non-stationary. Meaning, they do not repeat every 2π radians. Such signals energy can be shown as:

$$\int_{-\infty}^{\infty} |f(t)|^2 \, dt < \infty \tag{2.11}$$

Non-stationary signals and periodic signals are essentially different. In particular, the local mean of every base function that forms non-stationary $f(t)$ should tend to zero at $\pm\infty$. However, a sinusoidal wave family cannot be such a basis (they are non-local and stretch out to infinity [45]). Therefore, for the basis functions of non-stationary signals, let us consider wavelets well-localized soliton-like 'small waves' [46].

2.3.1 Wavelets

Wavelets differ from sine and cosine functions. There are many types of wavelets and some have quite unique shapes, which are defined by their mathematical properties. There is no wavelet "one fits all". Daubechies and Symlets are good for signal and image denoising, while Biorthogonal is a good candidate for signal and image compression [47].

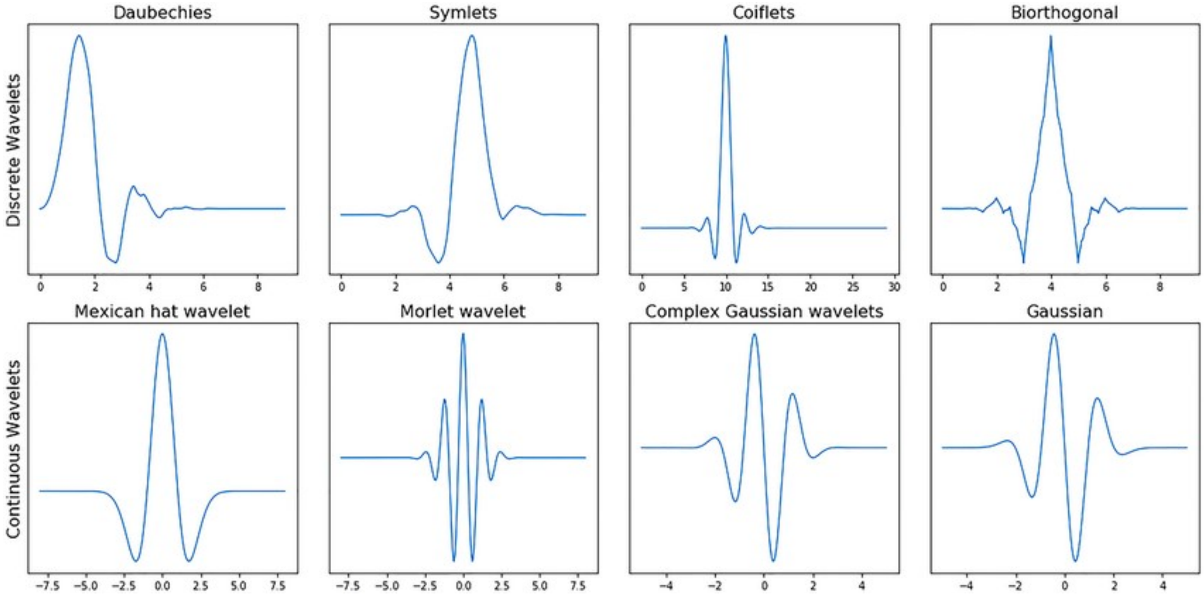


Figure 2.7: Figures of common mother wavelets [48]

To show how wavelets are expressed mathematically, we start with the Haar wavelet. Normalized form of the wavelets (more on that later):

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \tag{2.12}$$

Where

$\psi(t)$ - mother wavelet. Symbol reads as "psi" [49] [50].

$\psi_{j,k}(t)$ - daughter wavelet. When $j \geq 1$ it is considered as daughter wavelet and when $j = 0$ as mother wavelet.

$\phi(t)$ - scaling function (father wavelet). Haar's wavelet scaling function is a box function. Symbol reads as "phi".

$$boxfunction \begin{cases} 0\, for\, t\ <\ 0 \\ 1\, for\, 0\ <\ t\ <\ 1 \\ 0\, for\, t\ >\ 1 \end{cases}$$

$\psi(t) = \phi(2t) - \phi(2t-1)$ - Haar mother wavelet in terms of scaling function.

j - Dilation (scale) parameter. Stretches or compresses wavelet.

k - Translation (position) parameter. Moves wavelet across t - axis.

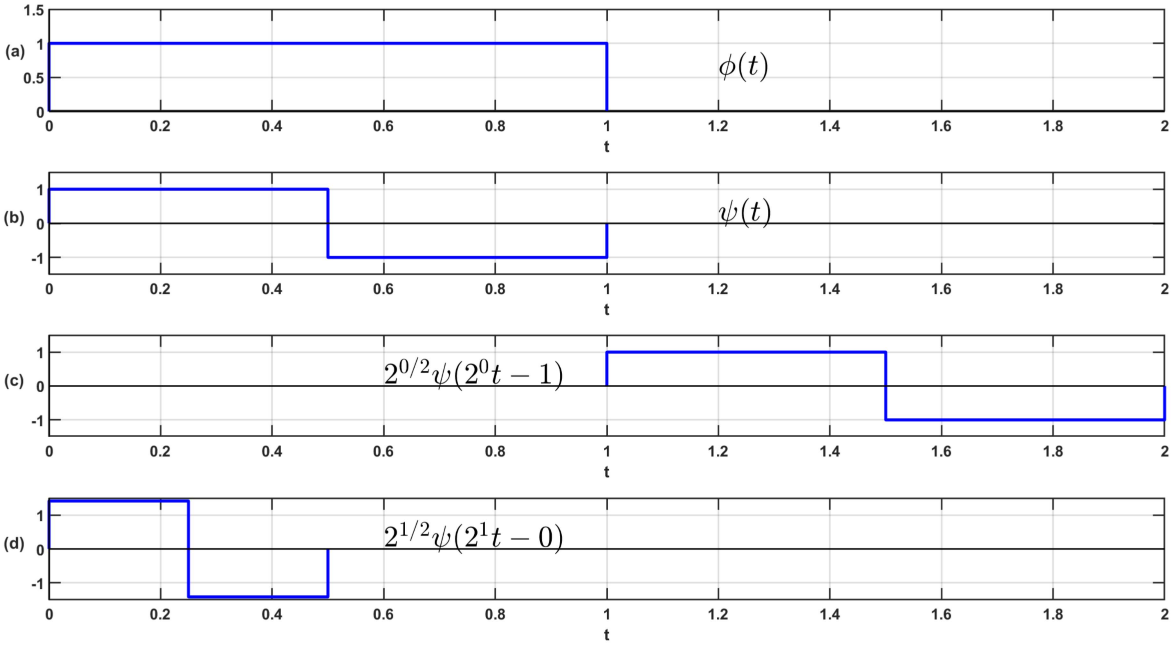


Figure 2.8: Haar wavelet: (a) Scaling function $\phi(t)$. (b) Mother wavelet $\psi(t)$. (c) Translation of $\psi(t)$ by $k = 1$ while $j = 0$. (d) Dilation of $\psi(t)$ by $j = 1$ while $k = 0$

The wavelets' normalized form (2.12), means

$$\int_{-\infty}^{\infty} |\psi_{j,k}|^2\, dt = 1$$

The $2^{j/2}$ is the normalization constant for any values of j and k. Based on the definition of $\psi_{j,k}$, it is evident that $j = 0$ corresponds to the mother wavelet, translated by k units Fig.2.8(c). The successive daughter wavelets are obtained for values of $j \geq 1$. Increasing values of j results in narrower and narrower daughters! Normalized, narrower daughter wavelet (one corresponding to the parent wavelet) to $j = 1$) is taller by the factor $\sqrt{2}$ when compared to mother's wavelet Fig.2.8(d). The rule is followed by subsequent generations. Hence, as $j \rightarrow \infty$, with increasing height, the wavelets become extremely thin. The scaling function captures the average of a portion of the signal in an interval indicated by its width; the scaling function window locations are determined by the values of k. The differences are captured by the wavelets. The mother wavelet collects the signal differences at a size similar to the scaling function, while the thinner daughter wavelets investigate the differences at progressively finer scales. This is why wavelet transform is also known as multi-resolution analysis. Because the wavelet transform uses a finite size window, it may capture the local nature of a function or signal considerably more efficiently than the Fourier transform [50].

2.3.2 Wavelet Transforms

Now we take a look at Discrete Wavelet Expansion and Transform.

Discrete Wavelet Expansion [51]:

$$f(t) = \sum_k c_{j_0}(k) 2^{\frac{j_0}{2}} \phi(2^{j_0} t - k) + \sum_k \sum_{j=j_0} d_j(k) 2^{\frac{j}{2}} \psi(2^j t - k) \tag{2.13}$$

Which can be rewritten in compact form(2.14). Where j_0 denotes the lowest frequency band/coarsest scale. Increasing index (j_{0+1}) means frequency band shifts up in frequency.

$$f(t) = \sum_k c_{j_0}(k) \phi_{j,k}(t) + \sum_k \sum_{j=j_0} d_j(k) \psi_{j,k}(t) \tag{2.14}$$

Discrete Wavelet Transform is act of calculating coefficients c_j and d_j:

$$c_j(k) = \langle f(t), \phi_{j,k}(t) \rangle = \int_{-\infty}^{\infty} f(t) \phi_{j,k}(t)\, dt \tag{2.15}$$

$$d_j(k) = \langle f(t), \psi_{j,k}(t) \rangle = \int_{-\infty}^{\infty} f(t) \psi_{j,k}(t)\, dt \tag{2.16}$$

Coefficients c_j are called scaling or approximation coefficients and d_j wavelet or detail coefficients. Equation of Wavelet Transform is almost identical to Fourier Transform (2.10), the difference is in basis functions. While both e^{-jwt} (FT basis) and wavelets act as band-pass filter, e^{-jwt} band-passes are evenly spaced (shifted) in spectrum. Wavelets $\psi_{j,k}$ band-passes increase bandwidth by 2 and shift in frequency by 2 every time index j increases by 1 [52]. The scaling function behaves as a low-pass filter and does not shift in the spectrum. That can be seen in (2.14) where scaling function calculated only for j_0.

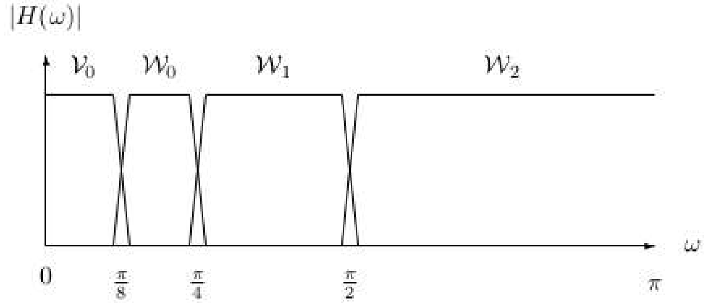


Figure 2.9: Spectrum of scaling and wavelet functions. V_0 and W_0 denotes scaling and mother wavelet function for j_0. $W_{1,2,3...}$ denotes daughter wavelet functions for $j_{1,2,3...}$ [51].

Fourier Transform coefficients are complex and come in the form $a \pm jb$, this is because of e^{-jwt} is a complex function. By running these coefficients through (2.2) we compute the frequency spectrum of a test signal. In the case of Wavelet Transform most of the wavelets are real functions. That means plotting a graph of wavelet coefficients will not display frequency spectrum but an approximation of filtered test signal in time.

Figure (2.10) show results of Discrete Wavelet Transform. The wavelet function that is used is Haar. Approximation in V_0 denotes c_{j_0} coefficients (scaling function). Plotting these coefficients results in a time-domain signal that has been processed through a low-pass filter. The next step is to compute d_{j_0} coefficients (mother wavelet), which represent the band-pass filtered part of the test signal. Approximation in V_1 is the sum of c_{j_0} and d_{j_0} coefficients. Approximation in V_2 is the sum of c_{j_0}, d_{j_0} and d_1 coefficients. And the list goes on, increasing the amount of wavelet coefficients improves approximation. The sum of coefficients can be thought as a result of low-pass filtering. Filter shift its cut-off frequency higher, as the number of transformed wavelets increase. This can

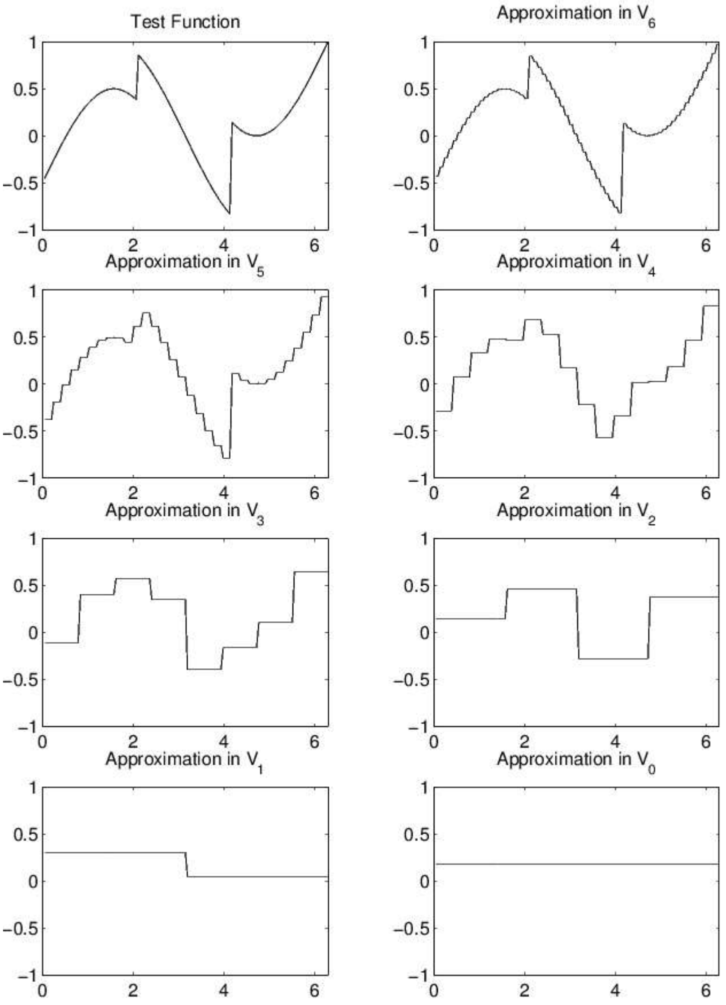


Figure 2.10: Haar approximation of test function with two discontinuities [51].

be seen in fig.(2.9), where sum of V_0, W_0 and W_1 creates low-pass filter with cut-off frequency at $\pi/2$.

A new tool is required to represent time-frequency (spectrum) from real valued Discrete Wavelet Transform coefficients. This tool is known as the Hilbert Transform. It is a method for producing complex-valued signals from real-valued signals [36, p.479]. Then by applying (2.2) and (2.3) on a complex signal we can extract frequency and phase of wavelet coefficients.

If the wavelet is complex, such as the complex Morlet wavelet it is possible to compute the frequency spectrum of the signal directly. The same logic will apply - wavelets produce a band-pass filter bank. Computing Wavelet Transform generates wavelet coefficients, this time in complex form. We can not use a complex Morlet wavelet with Discrete Wavelet Transform, for that we have to apply Continuous Wavelet Transform.

$$CWT(a,b) = \int_{-\infty}^{\infty} f(t)\frac{1}{a}\psi^*(\frac{t-b}{a})\,dt \tag{2.17}$$

Equation (2.17) depicts Continuous Wavelet Transform. Parameter a - dilation, b - translation and $*$ denotes complex conjugate [53]. Since we divide by a, it has to be greater than 0 ($a > 0$). By increasing a parameter, wavelet stretches in time, which leads to narrower bandwidth that shift towards zero Hz.

Since by definition Continuous Wavelet Transform (same as Fourier Transform as well) takes infinitesimal steps for all its parameters (t,b,a) it produces a lot of redundant information. Because of that Discrete Wavelet Transform are often preferred. To combat redundancy, the Dyadic scale is used for the dilation of a wavelet. That means the bandwidth of the next wavelet increased or decreased by 2 (1,2,4,8,16,32...). This can be seen in fig.(2.9) where bandwidth keeps on increasing.

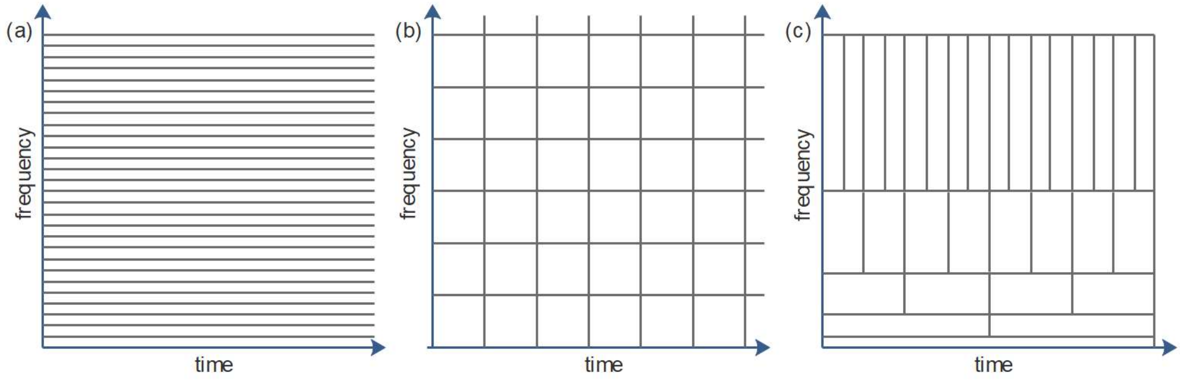


Figure 2.11: Time-frequency comparison: (a) Fourier Transform. (b) Short-Time Fourier Transform. (c) Continuous Wavelet Transform.

Let us compare the spectrum produced by Fourier Transform, Short Time Fourier Transform, and Discrete Wavelet Transform fig.(2.11). Fourier Transform does not have the concept of time, but frequency resolution is very good from low to high frequencies. Short-Time Fourier Transform balances between time and frequency resolution. where increasing one degrades another. Wavelet Transform is something in between, frequency content extracted dyadically. Low frequencies are extracted throughout the whole duration of the signal. Next frequency content shifted up by 2 and resolution decreased by 2. That scheme provides us with good time localization of frequency content for the practical signals. Often we are interested in high-frequency content when working with such signals.

2.4 Empirical Wavelet Transform

Empirical Wavelet Transform is a method for constructing a family of wavelets that are tailored to the processed data. From a Fourier perspective, this technique is analogous to creating a set of bandpass filters. One approach to achieving adaptability is to think that the filters' supports are dependent on where the information in the examined signal's spectrum is placed. [54].

Let us discuss why such transform is very useful. Take a look at the test signal composed of low-frequency sinusoids fig.(2.12). The bandwidth of the signal is 56 Hz and the signal is sampled at 1 kHz. By following Nyquist criteria our scale of frequencies is $[0, \pi]$ = [0Hz,500Hz]. By executing Discrete Wavelet Transform with same dyadic scale as in fig.(2.9) all frequencies of the test signal would lay in V_0 (scaling function as lowpass filter), which spans from 0 to 62.5 Hz. That means no decomposition of the test signal would happen at all. To achieve full decomposition down to all 5 sinusoids would require a high number of wavelets and a lot of them would be empty (assuming the dyadic scale is used).

A solution to this problem is to decompose this signal by Empirical Wavelet Transform. Dyadic scaling is fixed, while EWT allows us to adapt to a signal. It segments the spectrum of a signal by detecting the boundaries of each bandpass filter that wavelets represent. For the segmentation of the Fourier spectrum of the examined signal, many methods such as local maxima, lowest local minima between two selected local maxima (local maxmin), adaptive methodology, scale-space, and order statistic filters (OSF) have been utilized [55].

We visualise Empirical Wavelet Tranform by decomposing test signal $f(t) = \sin(2\pi 20t) + \sin(2\pi 30t) +$

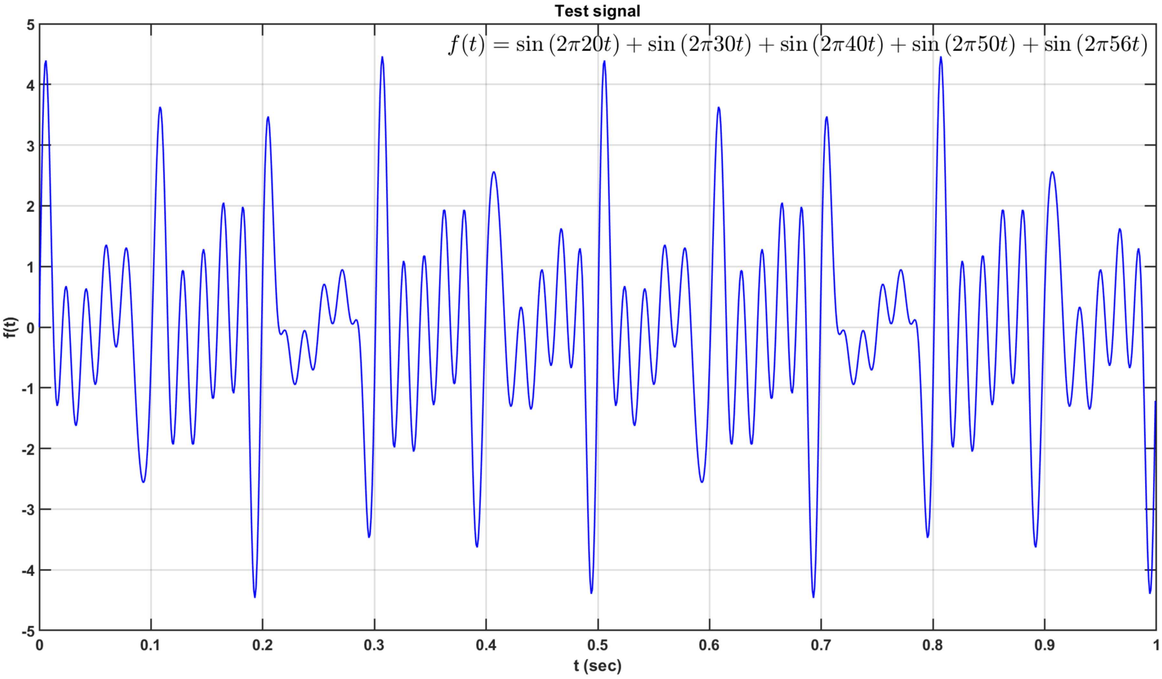


Figure 2.12: Test signal composed of 5 sinusoids: 20, 30, 40, 50 and 56 Hz

$\sin(2\pi 40t) + \sin(2\pi 50t) + \sin(2\pi 56t)$. Method for boundaries detection is local maxima. Algorithm automatically finds boundaries and creates wavelets which are used for signal decomposition. Figure (2.13) shows such segmentation of the spectrum. EWT algorithm is not perfect and creates 1 segment too many. Effects of this extra segment can be seen in fig.(2.14) where it represented by "ewt(1)". By looking at magnitude of "ewt(1)" (10^{-15}) we can safely discard this segment without affecting signal reconstruction from the rest the parts.

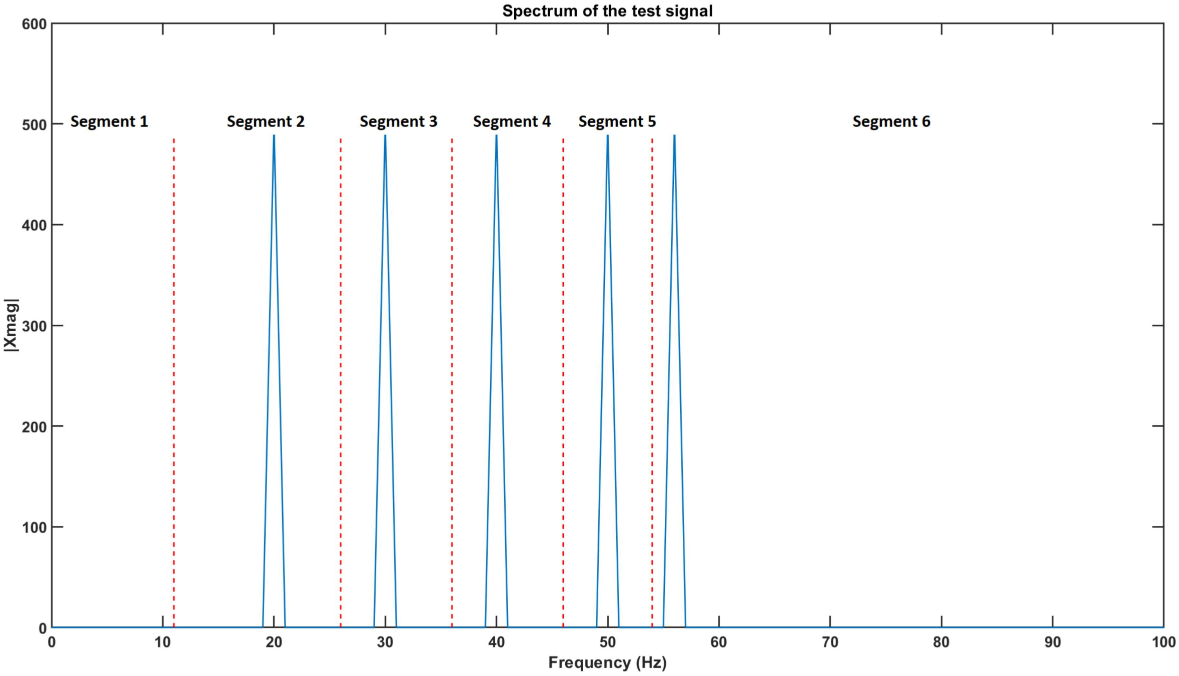


Figure 2.13: Segmentation of the test signals spectrum

The mathematical expression of empirical wavelets are as follows [54]:

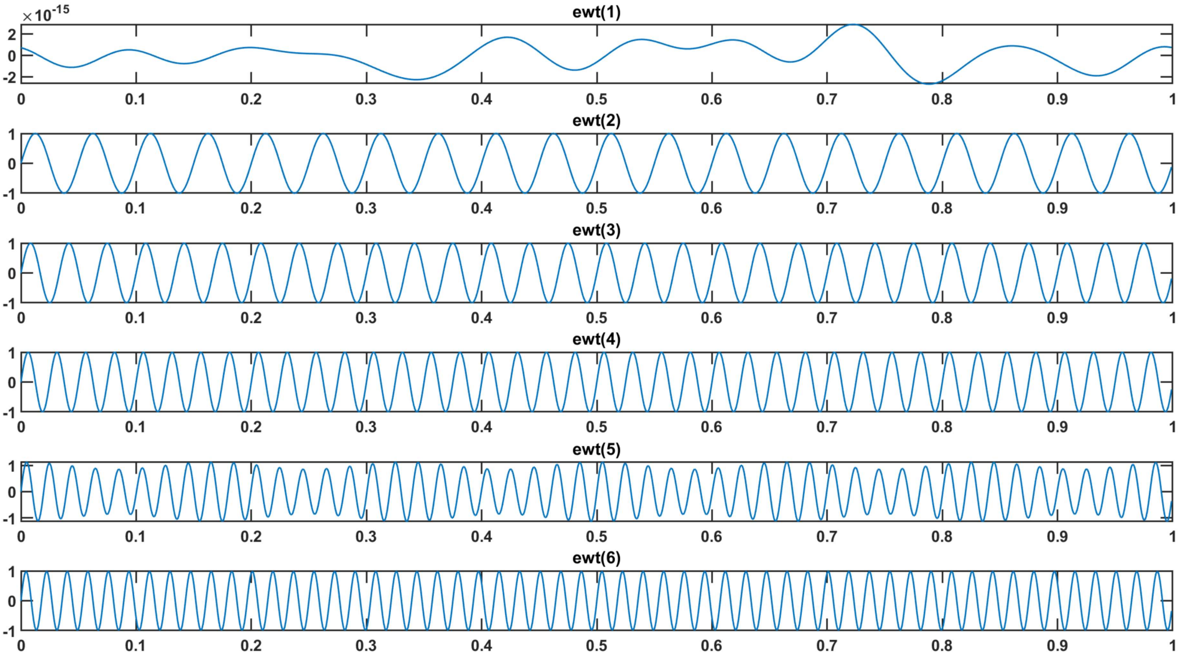


Figure 2.14: Decomposition of the test signal

$$\phi_n(\omega) = \begin{cases} 1 & if|\omega| \le \omega_n - \tau_n \\ \cos\left[\frac{\pi}{2}\beta\left(\frac{1}{2\tau_n}(|\omega| - \omega_n + \tau_n)\right)\right] & \\ & if\ \omega_n - \tau_n \le |\omega| \le \omega_n + \tau_n \\ 0 & otherwise \end{cases} \tag{2.18}$$

$$\psi_n(\omega) = \begin{cases} 1 & if\omega_n + \tau_n \le |\omega| \le \omega_{n+1} - \tau_{n+1} \\ \cos\left[\frac{\pi}{2}\beta\left(\frac{1}{2\tau_{n+1}}(|\omega| - \omega_{n+1} + \tau_{n+1})\right)\right] & \\ & if\omega_{n+1} - \tau_{n+1} \le |\omega| \le \omega_{n+1} + \tau_{n+1} \\ \sin\left[\frac{\pi}{2}\beta\left(\frac{1}{2\tau_n}(|\omega| - \omega_n + \tau_n)\right)\right] & \\ & if\omega_n - \tau_n \le |\omega| \le \omega_n + \tau_n \\ 0 & otherwise. \end{cases} \tag{2.19}$$

$$\beta(x) = x^4(35 - 84x + 70x^2 - 20x^3) \tag{2.20}$$

Where ω_n denotes the center of each segment and τ_n overlap between segments.

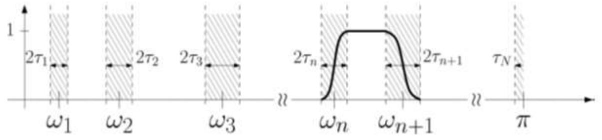


Figure 2.15: EWT visualization of segmentation with use of ω_n and τ_n [54]

2.4.1 Fixed segmentation of the spectrum

Often we would like to segment the spectrum according to our needs. This can be done by manually constructing filter banks. But the manual method is error-prone and becomes more difficult as the number of segments increases. With the help of EWT, it is possible to set the boundaries of each segment and let wavelets act as bandpass filters.

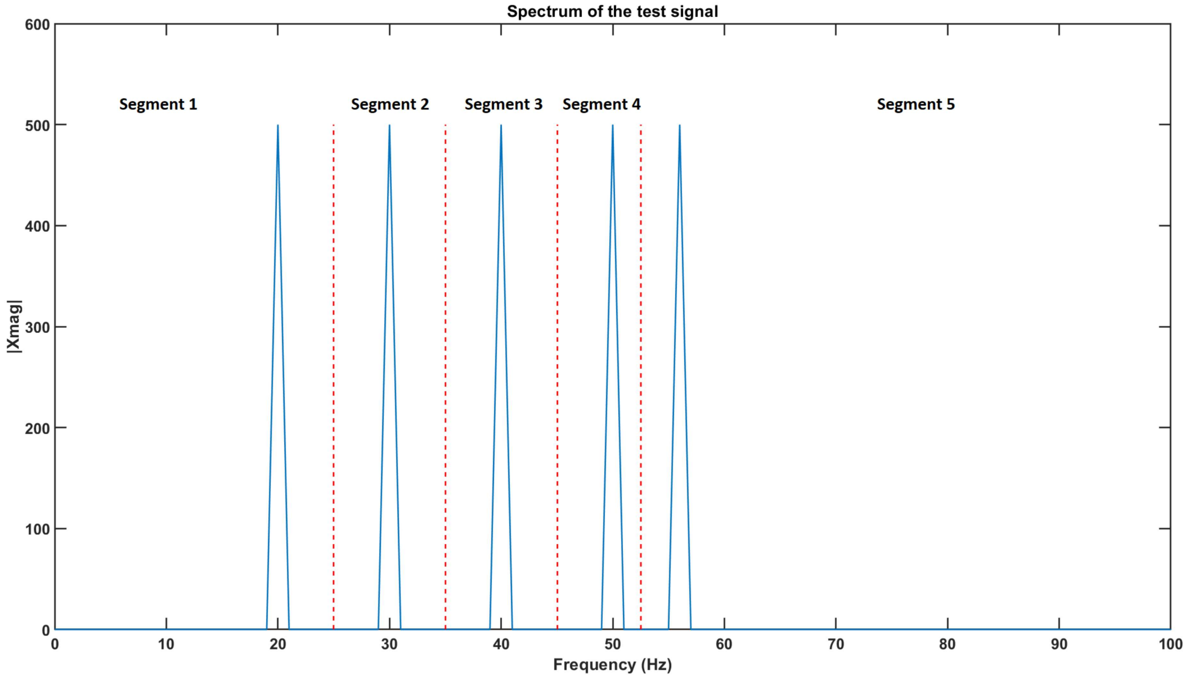


Figure 2.16: Manual segmentation of the test signals spectrum

Equation (2.21) depicts how to calculate boundaries for spectrum segmentation [55]. B_0 is the upper boundary of the first segment on a scale of $[0,\pi]$ (0 and π are included by default). F_0 is the upper frequency in Hz of the first segment and F_s is the sampling frequency of the signal in Hz.

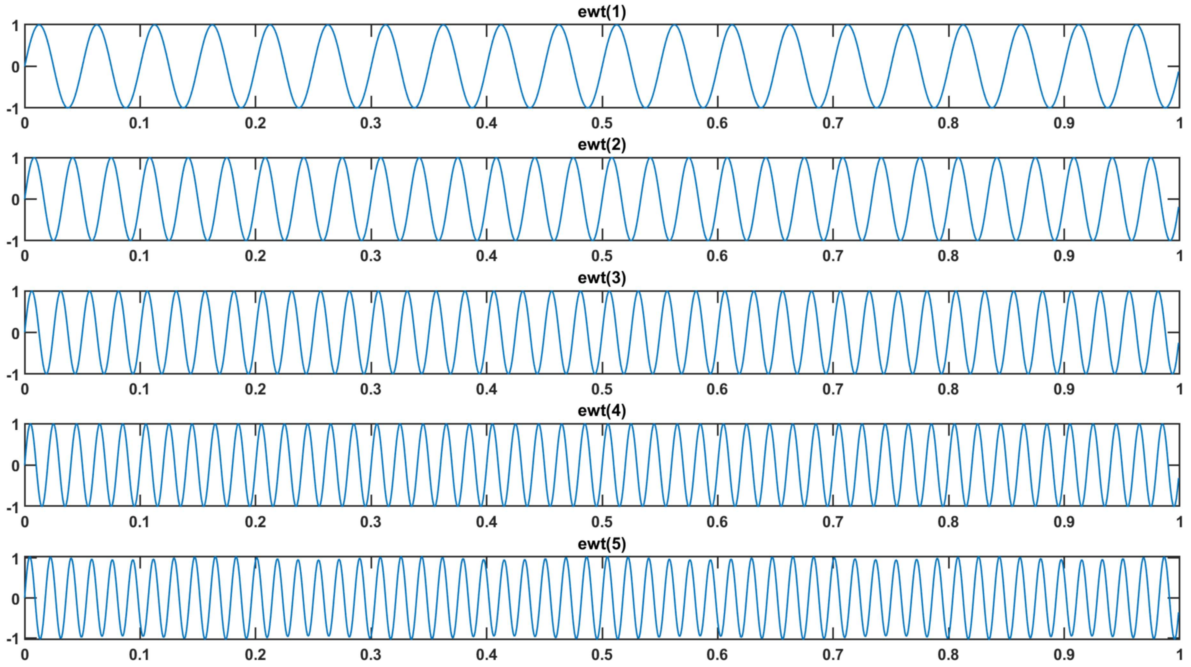


Figure 2.17: Decomposition based on manual segmentation of test signals spectrum

$$B_i = \frac{2\pi F_i}{Fs} \tag{2.21}$$

Let us visualise manual segmentation and remove extra segment from the same test signal. We pick frequencies for segmentation as follows: "25, 35, 45 and 53 Hz". These produce following segments: [0-25Hz], [25-35Hz], [35-45Hz], [45-53Hz], [53-500Hz]. Figure (2.16) shows result of manual segmentation with chosen boundaries, while fig.(2.21) show decomposition of the test signal.

2.5 Convolutional Neural Network

Convolutional Neural Network is a type of Deep Neural Network (has many layers), which is typically used in machine vision and image classification. CNN requires fewer hardware resources when processing big images than other Artificial Neural Networks [56]. Convolutional operation (multiply and add) is done by sliding the filter kernel over the image in small steps. This can be greatly accelerated by using video graphics cards. Due to its simplicity and low amount of parameters chance of network overfitting is reduced. All of these make CNN a perfect candidate for our task of noise classification and removal.

An image can be represented as a matrix, e.g. full HD resolution image is a 1080x1920 pixels matrix. In general, images are represented as HxWxD, where H is height W is the width and D is depth. D is used to represent color (at the input layer), so $D = 1$ tells us it is the black and white image or $D = 3$ could mean RGB (Red, Green, and Blue).

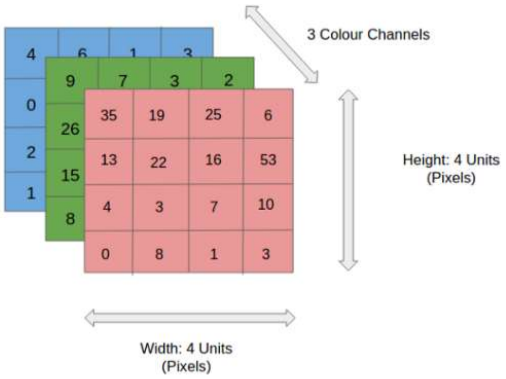


Figure 2.18: 4x4x3 RGB image [57]

Audio tracks that are used in this thesis is mono, and it is just an array of numbers (1xN matrix). As it has been shown EWT can be used for signals decomposition. Several segments of audio frames or tracks form an image-like matrix with sizes that are chosen by the user.

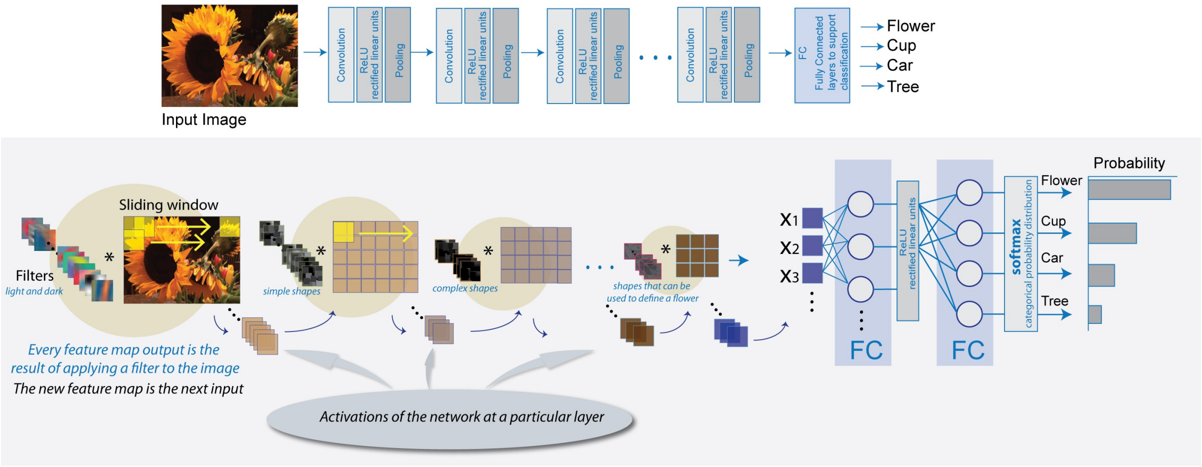


Figure 2.19: CNN architecture for image classification [58]

Figure (2.19) depicts one of the typical configurations of CNN for image classification. It has multiple repeating layers. Order of these layers matters and it goes like this: input layer, convolution layer, activation function layer, pooling layer, fully connected layer, softmax layer, and classification layer. In the case of convolution, activation, and pooling layers repetition output of a pooling layer acts as the input layer for the following convolution layer.

2.5.1 Convolution Layer

The convolution layer is the most important layer and its job is to extract features from the input. Features such as edges or colors. Later based on these features network is trained to identify in our case, noise patterns. Which are then used for the classification of the noise types. Feature extraction is done by shifting filters (kernel) over an input, starting from the top left corner and going to the right. The usual size of the kernel is 3x3xN or 5x5xN, where N is the depth of the input layer [59].

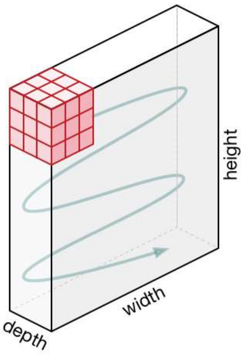


Figure 2.20: 3x3x3 kernel movement [57]

Figure (2.20) depicts movement of a kernel. Since the input layer has a depth $D = 3$, the kernel forms a cube. This cube is composed of 3 filters the size of 3x3 which are stacked on top of each other. We call them filters because their job is to filter out features. CNN training consists of computing filter weights that will give users better and better classification results.

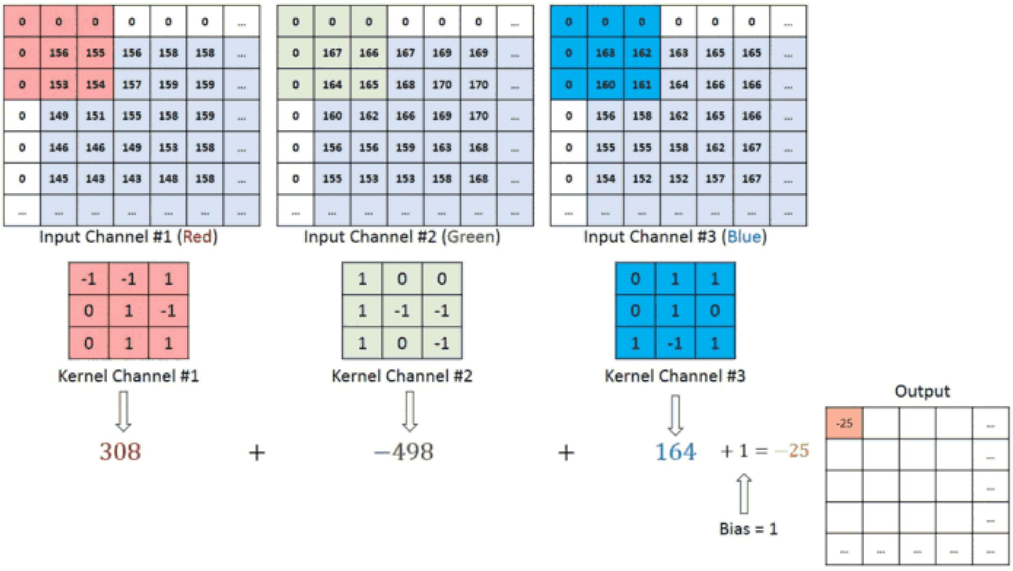


Figure 2.21: Kernel convolution visualization [57]

Figure (2.21) depicts kernel convolution. Input image with a size of HxWx3 is split into 3 channels with sizes of HxWx1. Each filter is having its own weights and gets its own channel to slide over. Convolution of a filter itself and a part of the input is simply multiplied and add method fig.(2.22). It is important to note that the output of the convolution layer does not have the same dimension as the input. The size of the output depends on the size of the filter and stride parameters. Stride tells filter how many pixels to shift in a horizontal and vertical position after each convolution. Usually, it is desirable to reduce the size of the output matrix, in cases where it is not, we can add zeros at the edges of an input image.

Figure (2.23) visualizes how extracted features from the convolution layer may look. The picture of a cat has a grayscale color scheme and its depth is 1. Fourfilters are applied to the same input image and produce four new images.

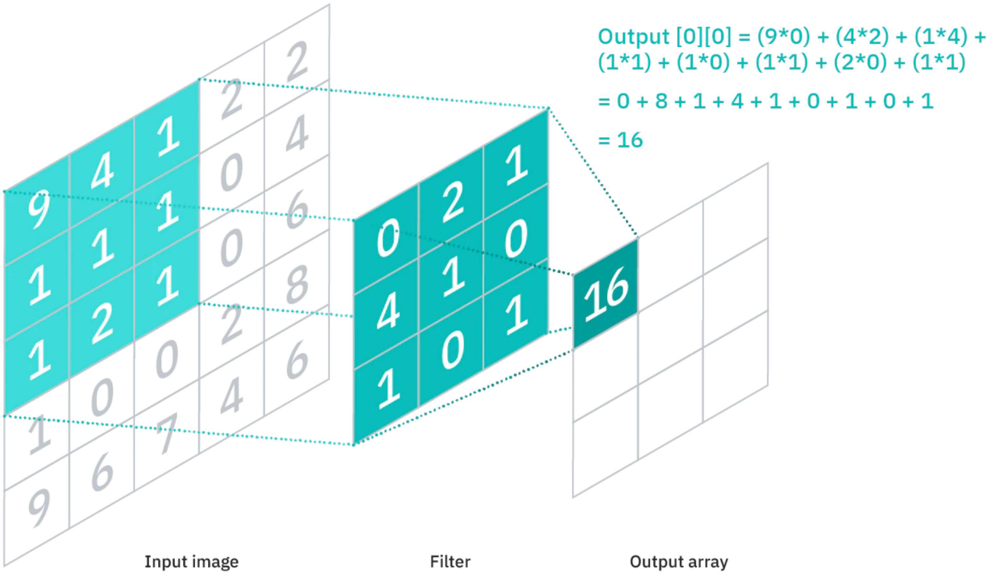


Figure 2.22: Filter convolution example [60]

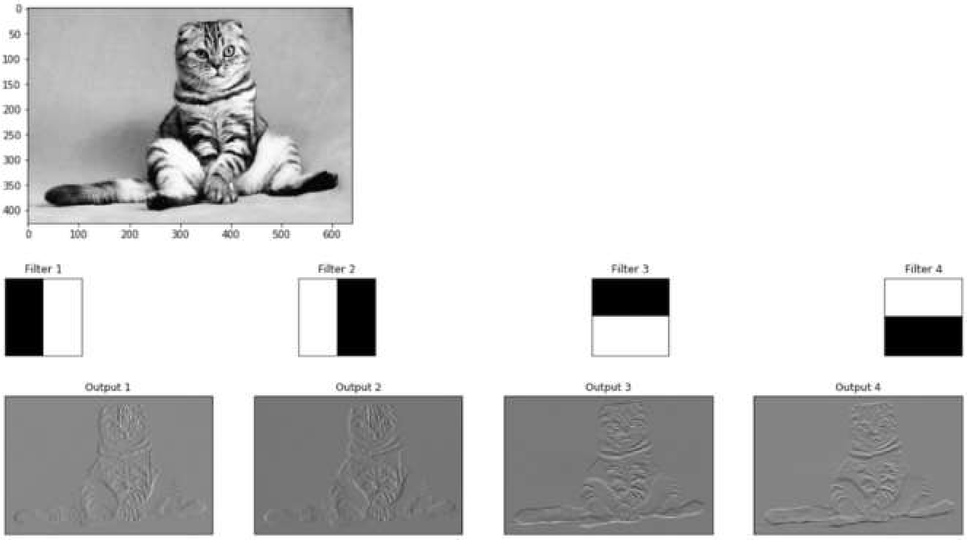


Figure 2.23: Convolution Layer extracted features visualization [59]

2.5.2 Activation Function Layer

The activation function layer is the layer where a non-linear function is applied to the input. This makes the network non-linear, which makes it harder to train but enables it to learn complex features.

Figure (2.24) depicts graphs of four common activation functions and their formulas. These functions are widely used not only in CNN but in many other Neural Networks. The main job of these functions is to make the network non-linear. The sigmoid function do that by squishing input between 0 and 1, while Tanh squishes between -1 and 1. These two functions were popular choices before, but they have a problem with saturation. This leads to a "vanishing gradient" problem in networks with many layers [56].

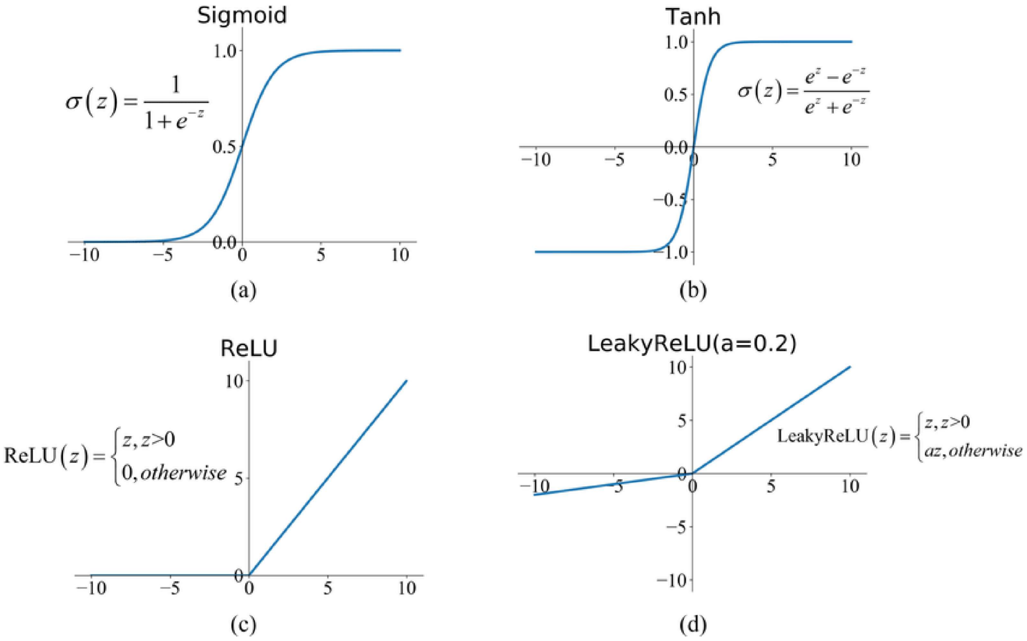
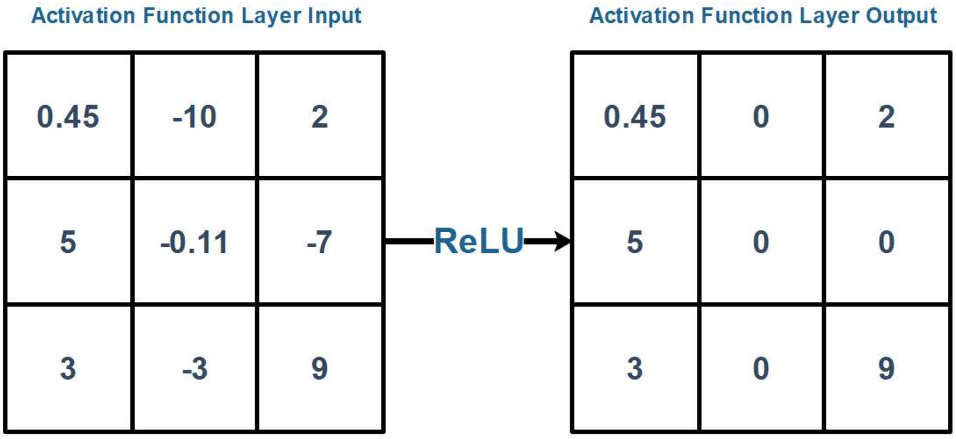


Figure 2.24: Four commonly used activation functions [61]

A popular activation function that is easy to compute is ReLU. This function zeroes out every pixel that has a negative value but leaves anything else unchanged. The activation function layer does not change the size of the input matrix. Output and input are equal in size.



Activation Function Layer Input

0.45	-10	2
5	-0.11	-7
3	-3	9

Activation Function Layer Output

0.45	0	2
5	0	0
3	0	9

Figure 2.25: ReLU activation function visualisation

2.5.3 MaxPool Layer

MaxPool layer's job is to deliberately reduce the size of the input matrix. This reduces computational load and chance of overfitting, the downside is loss of information. Same as with convolution layer filters and stride are used for downsizing. The usual size of maxpool filter is 2x2 or 3x3 [62].

Common variants of pooling filters are Max Pool and Average Pool. Max Pool simply returns the highest value while Average Pool computes the average of the part of the signal that filter slides over.

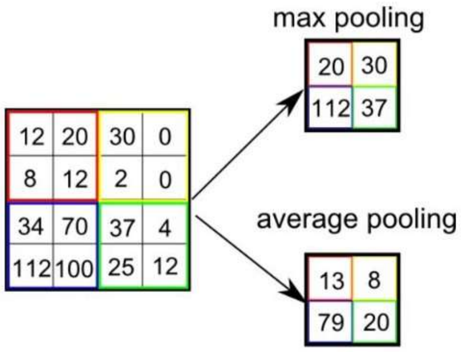


Figure 2.26: Max Pool and Average Pool visualization [57]

2.5.4 Fully Connected Layer

The last part of CNN is a regular Neural Network. The main part of this Neural Network is a Fully Connected Layer. FC layer job is to combine outputs from the previous layer to the next one. The next layer can be a Softmax function layer or just one more FC layer. Each Fully Connected layer have their own weights and biases, which are too used in network training. This leads to an increase in prediction accuracy [57].

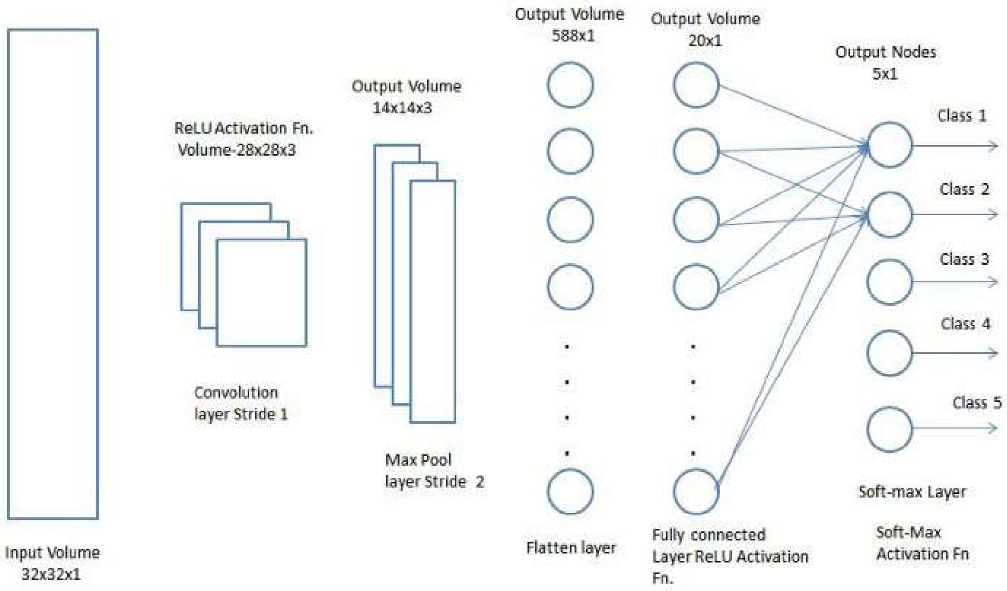


Figure 2.27: Complete classification CNN [57]

Chapter 3

Proposed Solutions

This chapter is split into two parts. It describes a classification of audio noise embedded in human speech and the removal of such noise from the signal. Both classification and noise removal is done with the help of Convolutional Neural Networks. Both cases utilize Empirical Wavelet Transform for signals decomposition.

3.1 Classification Part

The classification part of this thesis solves the problem of classifying 14 different kinds of noise embedded into audio signals. There are only 30 audio signals to train on, per type of noise. The network is trained for signal-to-noise ratios of 10, 0, and -5 decibels.

We can artificially increase the amount of train data by splitting signals into frames. The next step is to decompose each frame with Empirical Wavelet Transform. Boundaries for the segmentation of the Fourier spectrum are chosen by the user. But to make it simple, the frequency range for each segment is the same. The decomposed frame forms a 2D object, we can imagine it as a picture. The next step is to feed these picture-like frames into Convolutional Neural Network, which extracts features from each frame. Then these extracted features are sent to another neural network whose job is to classify each frame according to the type of noise.

Based on hardware and accuracy requirements, our classification network allows the user to choose and modify some of its parameters. Such as the size of frames and the number of segments. For example, a low number of segments decreases the accuracy of classification but requires less RAM.

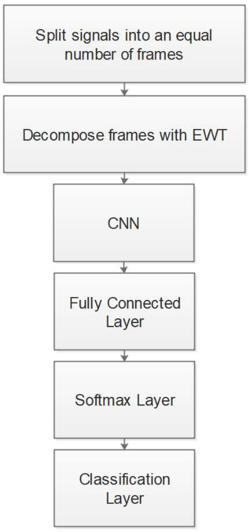

Split signals into an equal number of frames
Decompose frames with EWT
CNN
Fully Connected Layer
Softmax Layer
Classification Layer

Figure 3.1: Noise classification block diagram

3.1.1 Data set

This thesis uses an audio set that consists of 30 tracks (can be found at [63]). The language spoken is English. Tracks are divided evenly between male and female voices, with an average length of 2.5 seconds. Tracks of audio noise set are taken from Aurora experimental framework (can be found at [64]).

Noise set consists of the sound of: "airplane, airport, babble, car, drone, exhibition, helicopter, restaurant, sea waves, station, street, toilet flush, train and washing machine". Noise sounds can be categorized into 3 types. Stationary - noise is homogeneous and varies very little with time, also there is little change in its frequencies. Airplane, car, helicopter, train and washing machine belongs to the stationary type of noise. The next type of noise is non-stationary - noise that changes with time, medium to a sharp change in frequencies. Drone, sea waves, station, street, and toilet flush belong to the non-stationary type of noise. A final type of noise is babble - noise that is made of unintelligible human speech in the background. This type of noise is particularly difficult to classify since it contains a resemblance to human speech. Airport, babble, exhibition, and restaurant belong to babble type of noise.

By noisy audio, we imply the mixing of clean (non-noisy) audio tracks with particular noise and SNR (signal-to-noise ratio). Chosen SNRs are 10, 5, 0 and -5 decibels. The sampling rate of noisy audio tracks is 8 kHz.

3.1.2 Framing and Segmentation

Preparation for classification consists of framing and segmentation of the audio tracks. First, all 30 noisy tracks are split into an equal number of frames. To calculate the number of frames this split will produce, we need to find the shortest track. In our set, this is track number 4 which is 16928 samples long. Then we divide 16928 by the length of the frame (one of the variables that the user may change) and round the quotient down to the nearest integer. A computed integer is the number of frames that we split tracks into. It will depend on the chosen length of the frame. If a track is longer than 16928 samples, then we simply ignore the rest. Figure (3.2) visualizes this procedure.

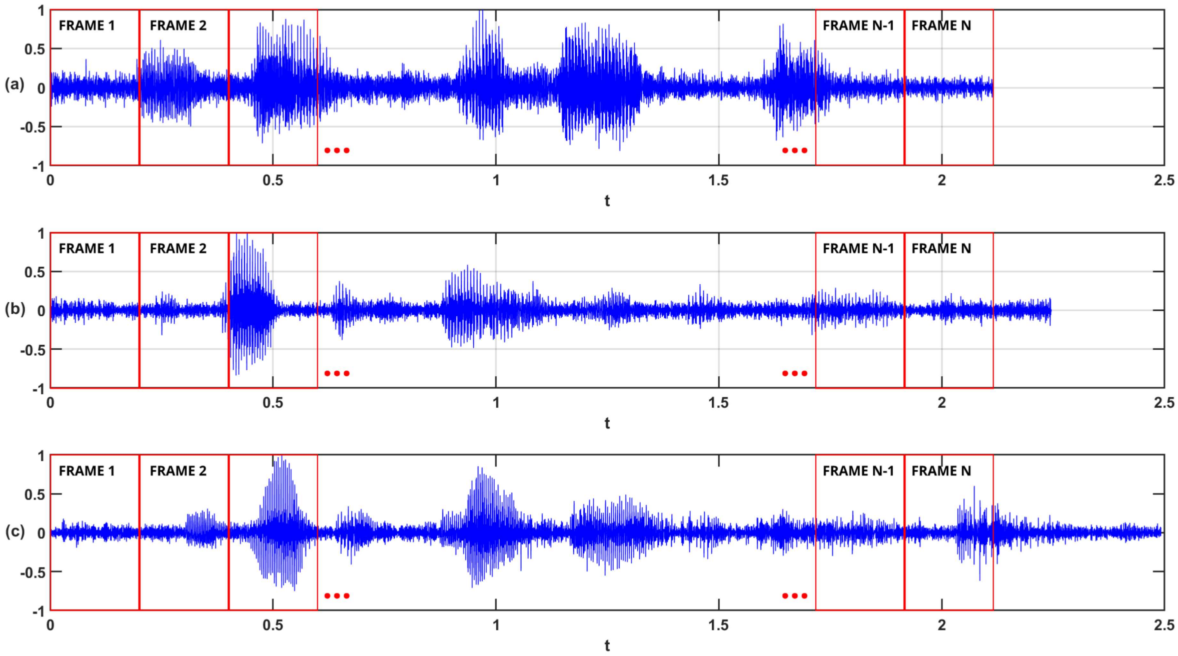


Figure 3.2: Splitting of noisy (5dB SNR) tracks into an equal number of frames: a) 16928 samples. b) 17967 samples. c) 19934 samples

The next step is to create boundaries for frames segmentation. Each frame will be decomposed by EWT into an equal number of segments. Amount of segments is chosen by the user. A higher number of segments requires a higher amount of RAM and VRAM but increases the accuracy of noise prediction. To make it simple, the boundaries of each segment are evenly spaced. Figure (3.3) visualises segmentation of one of the frames. Boundaries computed by eq.(2.21), where 0 and π are included by default. The bandwidth of each segment is then given by dividing Nyquist frequency by the total amount of segments.

$$\frac{4\ kHz}{50\ segments} = 80\ Hz$$

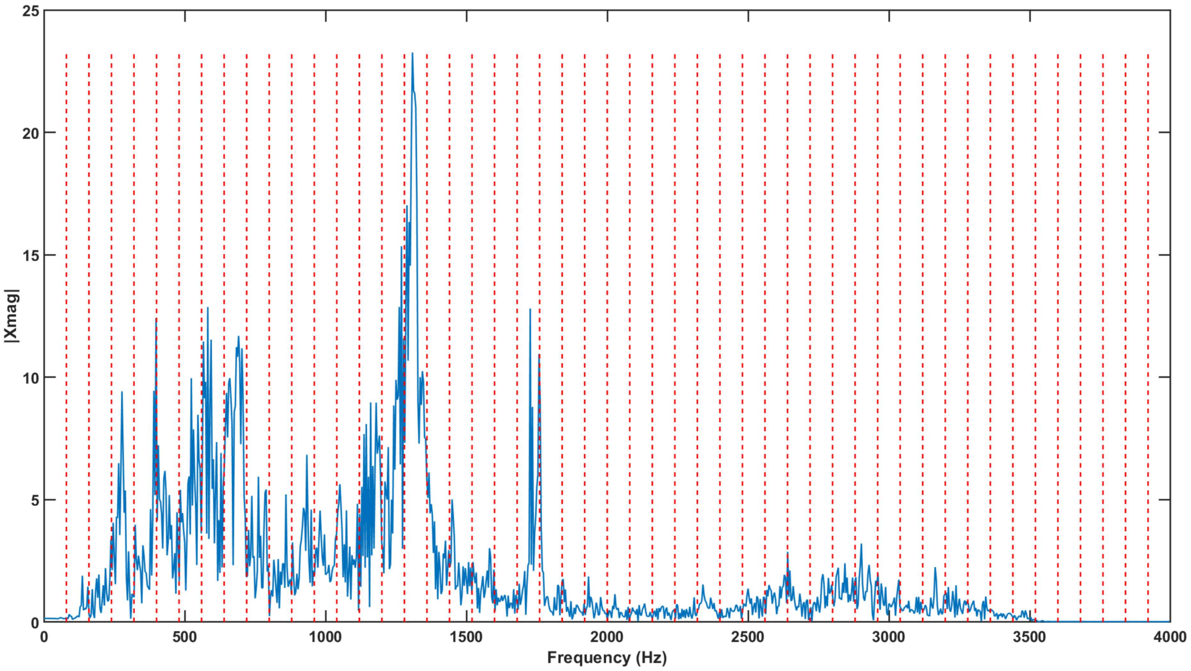


Figure 3.3: Segmentation of frames Fourier spectrum. 50 segments

Figure (3.4) visualises partial decomposition of one frame. We can see how segment 10, 20, 30, and 40 out of 50 looks like. Of course in the code, these segments are simply arrays of numbers.

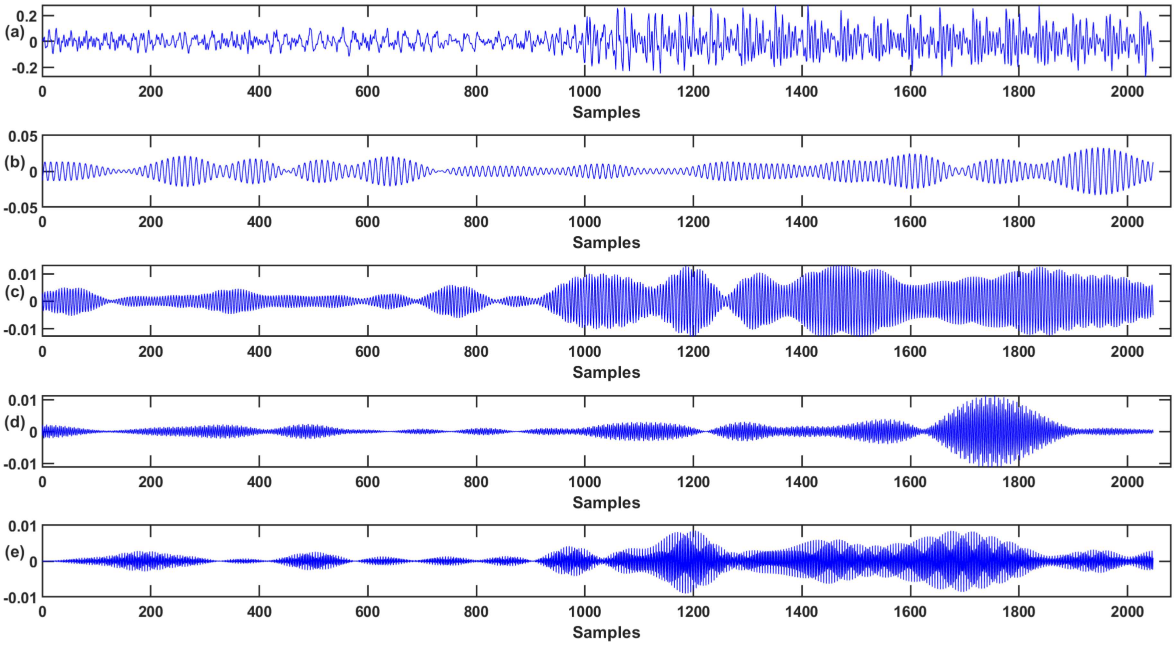


Figure 3.4: Decomposed frame: a) Frame 2048 samples. b) Segment 10. c) Segment 20. d) Segment 30. e) Segment 40.

Therefore the best way to describe decomposed frames is by matrices. Then, the 2048 samples frame, decomposed to 50 segments describes as a 50x2048 matrix.

The total amount of frames depends on the chosen length of the frame. If this length is 2048 samples then each noisy track produces 8 frames, which is 240 frames per class of noise. By having 14 classes of noise, the total number of frames is 3360.

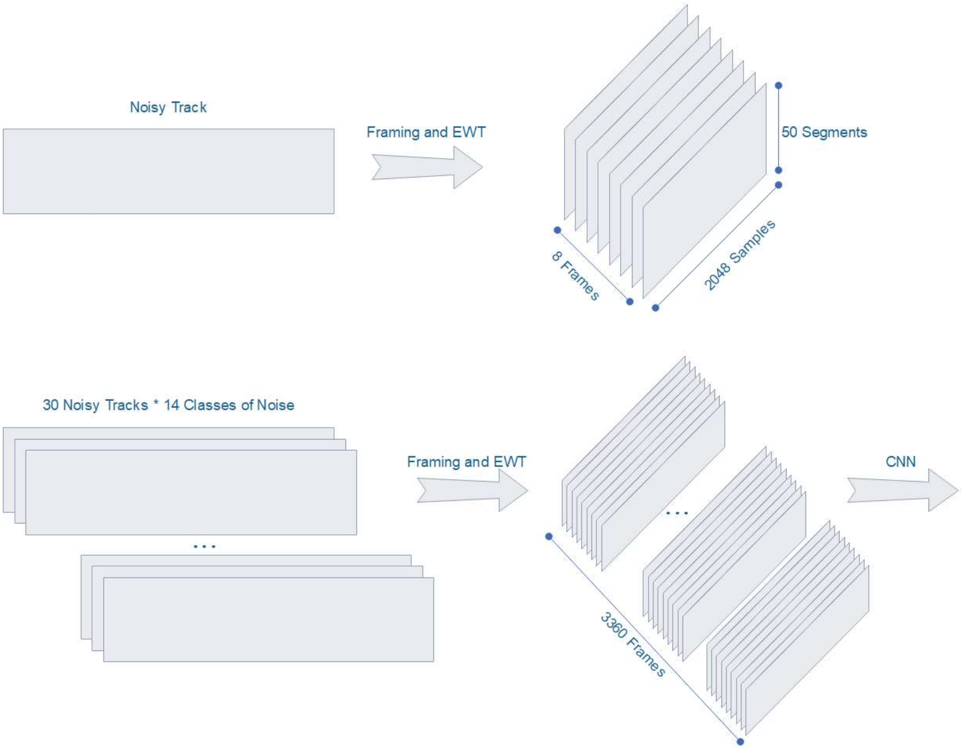


Figure 3.5: Visualization of signals preparation prior to classification

3.1.3 Noise Classification

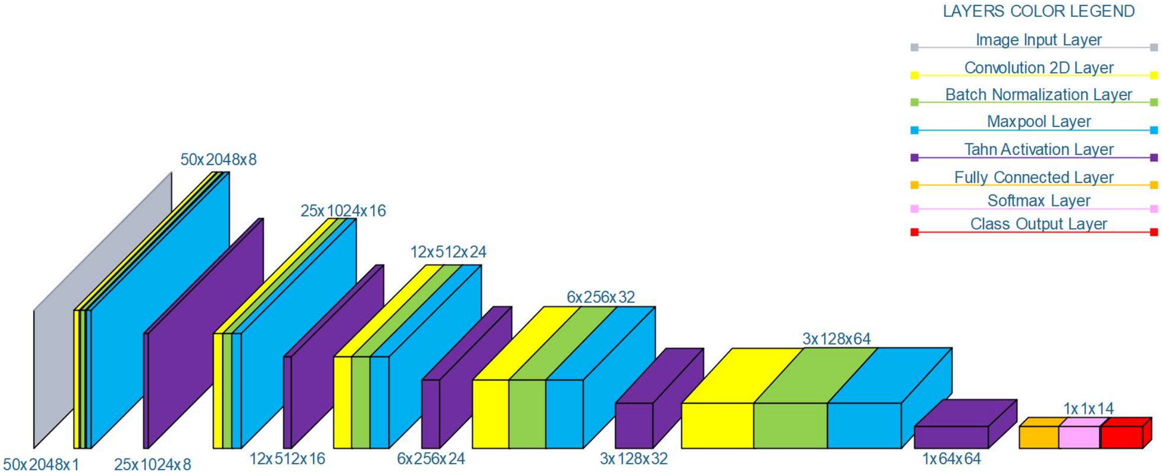
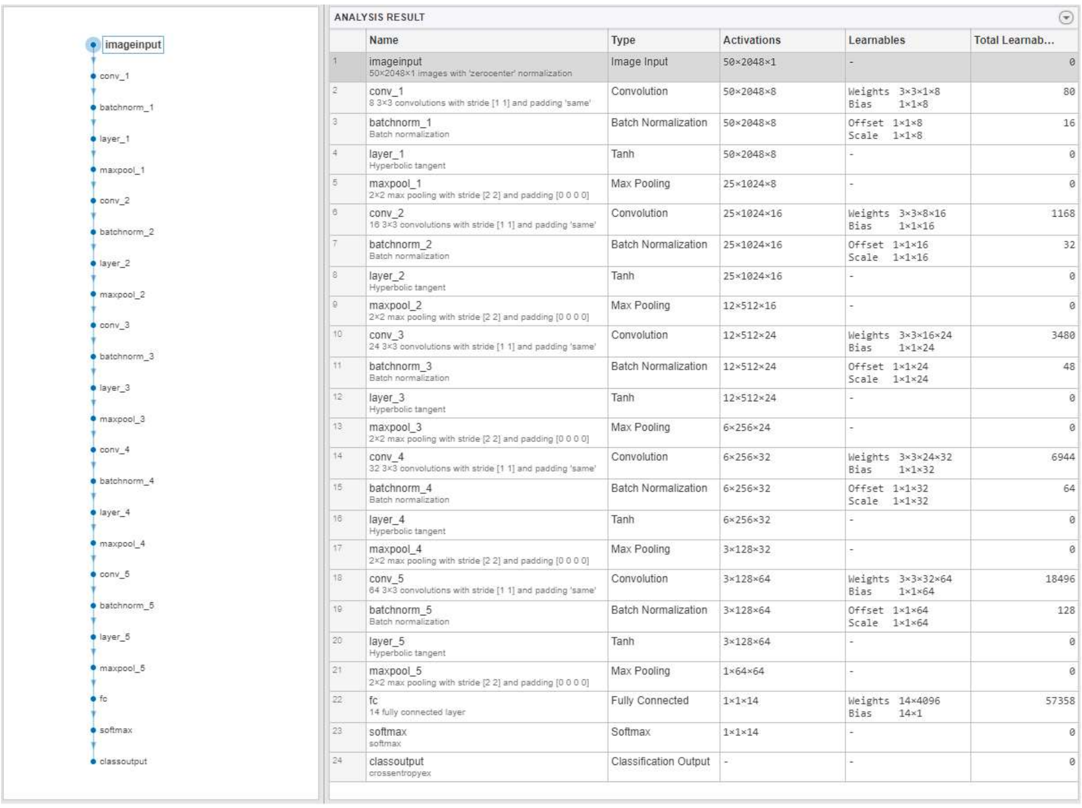


Figure 3.6: Classification CNN Layers Deep Stacking

The noise classification network consists of 24 layers in total. The first layer is input and the size of it will affect subsequent layers. Here we chose to train the network with a 50x2048 frame size. The next stack of 3 layers is convolution, batch normalization, and maxpool layers, followed by the activation layer. This process of layer stacking is then repeated 4 times. This is visualised in fig.(3.6). The classification itself is done at the last 3 layers, which consists of fully connected, softmax, and classoutput layers.

The size of the filter kernel is 3x3. For features extraction, we chose to start with 8 filters and double their amount at each subsequent convolution layer. By doing this we increase the amount of extracted features and increase the accuracy of prediction.



ANALYSIS RESULT

	Name	Type	Activations	Learnables	Total Learnab...
1	imageinput 50×2048×1 images with 'zerocenter' normalization	Image Input	50×2048×1	-	0
2	conv_1 8 3×3 convolutions with stride [1 1] and padding 'same'	Convolution	50×2048×8	Weights 3×3×1×8 Bias 1×1×8	80
3	batchnorm_1 Batch normalization	Batch Normalization	50×2048×8	Offset 1×1×8 Scale 1×1×8	16
4	layer_1 Hyperbolic tangent	Tanh	50×2048×8	-	0
5	maxpool_1 2×2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	25×1024×8	-	0
6	conv_2 16 3×3 convolutions with stride [1 1] and padding 'same'	Convolution	25×1024×16	Weights 3×3×8×16 Bias 1×1×16	1168
7	batchnorm_2 Batch normalization	Batch Normalization	25×1024×16	Offset 1×1×16 Scale 1×1×16	32
8	layer_2 Hyperbolic tangent	Tanh	25×1024×16	-	0
9	maxpool_2 2×2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	12×512×16	-	0
10	conv_3 24 3×3 convolutions with stride [1 1] and padding 'same'	Convolution	12×512×24	Weights 3×3×16×24 Bias 1×1×24	3480
11	batchnorm_3 Batch normalization	Batch Normalization	12×512×24	Offset 1×1×24 Scale 1×1×24	48
12	layer_3 Hyperbolic tangent	Tanh	12×512×24	-	0
13	maxpool_3 2×2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	6×256×24	-	0
14	conv_4 32 3×3 convolutions with stride [1 1] and padding 'same'	Convolution	6×256×32	Weights 3×3×24×32 Bias 1×1×32	6944
15	batchnorm_4 Batch normalization	Batch Normalization	6×256×32	Offset 1×1×32 Scale 1×1×32	64
16	layer_4 Hyperbolic tangent	Tanh	6×256×32	-	0
17	maxpool_4 2×2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	3×128×32	-	0
18	conv_5 64 3×3 convolutions with stride [1 1] and padding 'same'	Convolution	3×128×64	Weights 3×3×32×64 Bias 1×1×64	18496
19	batchnorm_5 Batch normalization	Batch Normalization	3×128×64	Offset 1×1×64 Scale 1×1×64	128
20	layer_5 Hyperbolic tangent	Tanh	3×128×64	-	0
21	maxpool_5 2×2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	1×64×64	-	0
22	fc 14 fully connected layer	Fully Connected	1×1×14	Weights 14×4096 Bias 14×1	57358
23	softmax softmax	Softmax	1×1×14	-	0
24	classoutput crossentropyex	Classification Output	-	-	0

Figure 3.7: Analysis report of the noise classification network

3.2 Denoising Part

The denoising part of this thesis attempts to remove 14 different kinds of noise that is embedded into audio signals. Signal-to-noise ratios of noisy signals are 0, -5, -10 decibels. Denoising itself is done with the help of CNN. There are 30 audio signals per type of noise.

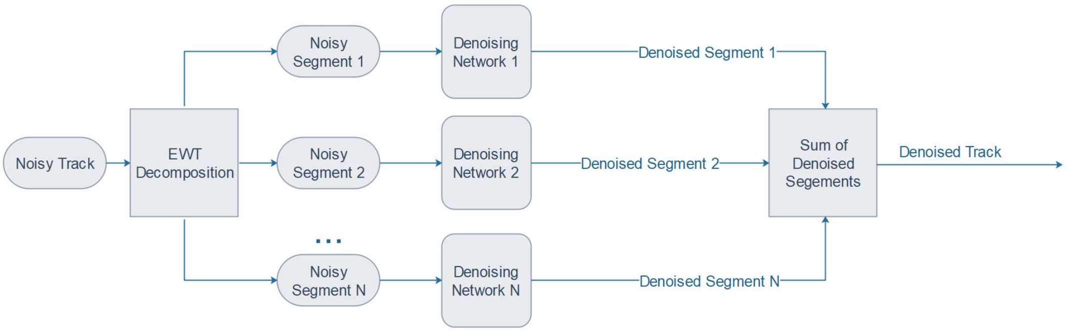


Figure 3.8: Denoising block diagram

The first step of removing noise from the audio track is to decompose it with Empirical Wavelet Transform. As in the classification part, boundaries for the segmentation are chosen by the user and the frequency range for each segment are the same. Amount of segments that are used are chosen by the user as well. Each segment is then processed by each own CNN. Output from these networks is denoised segments. Sum of these segments forms a denoised signal. The main idea is that each denoising CNN is working with the specific bandwidth of a noisy signal. Thus increasing the amount of noise was removed.

There are some similarities with classification networks. Both networks are using EWT for decomposition, and CNN for features extraction from each EWT segment. The first difference is that in the classification part, signals are split into an equal amount of frames and each of those frames is decomposed by EWT. In the denoising part, the full duration of the signal is utilized and decomposed by EWT. Signals length difference is no longer an issue. The second difference is that the classification network extracts features from time-domain data, while the denoise network extract from the frequency domain.

3.2.1 Denoising

The core of the denoising network is taken from the Matlab example, which can be found at [65].

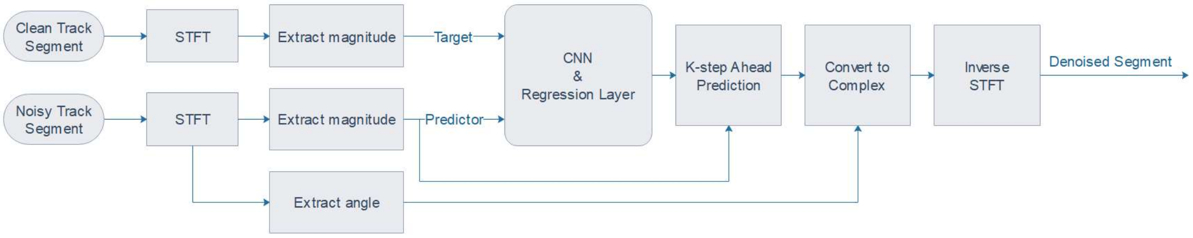


Figure 3.9: Denoising network block diagram [65]

The first step is to extract the frequency content of each segment. This is done by computing STFT with the following parameters: window length is 256 samples, the window function is hamming, the overlap is 75%. Next is to extract frequency magnitudes (eq.(2.2)) from clean and noisy STFT frames and extract phase angles from noisy STFT frames (eq.(2.3)). When extracting magnitudes and phase angles only the last half of the frame is computed. This is due to the dropping of negative frequencies and thus reducing the size of data [65].

Computed magnitudes of clean STFT frames are called targets. Each frame of clean STFT is the target and has a size of 129x1. Computed magnitudes of noisy STFT frames are called predictors. Predictors size is different from targets, 8 frames of noisy STFT produce 1 predictor with the size of 129x8 fig.(3.10).

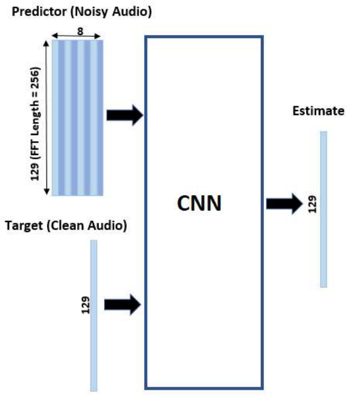
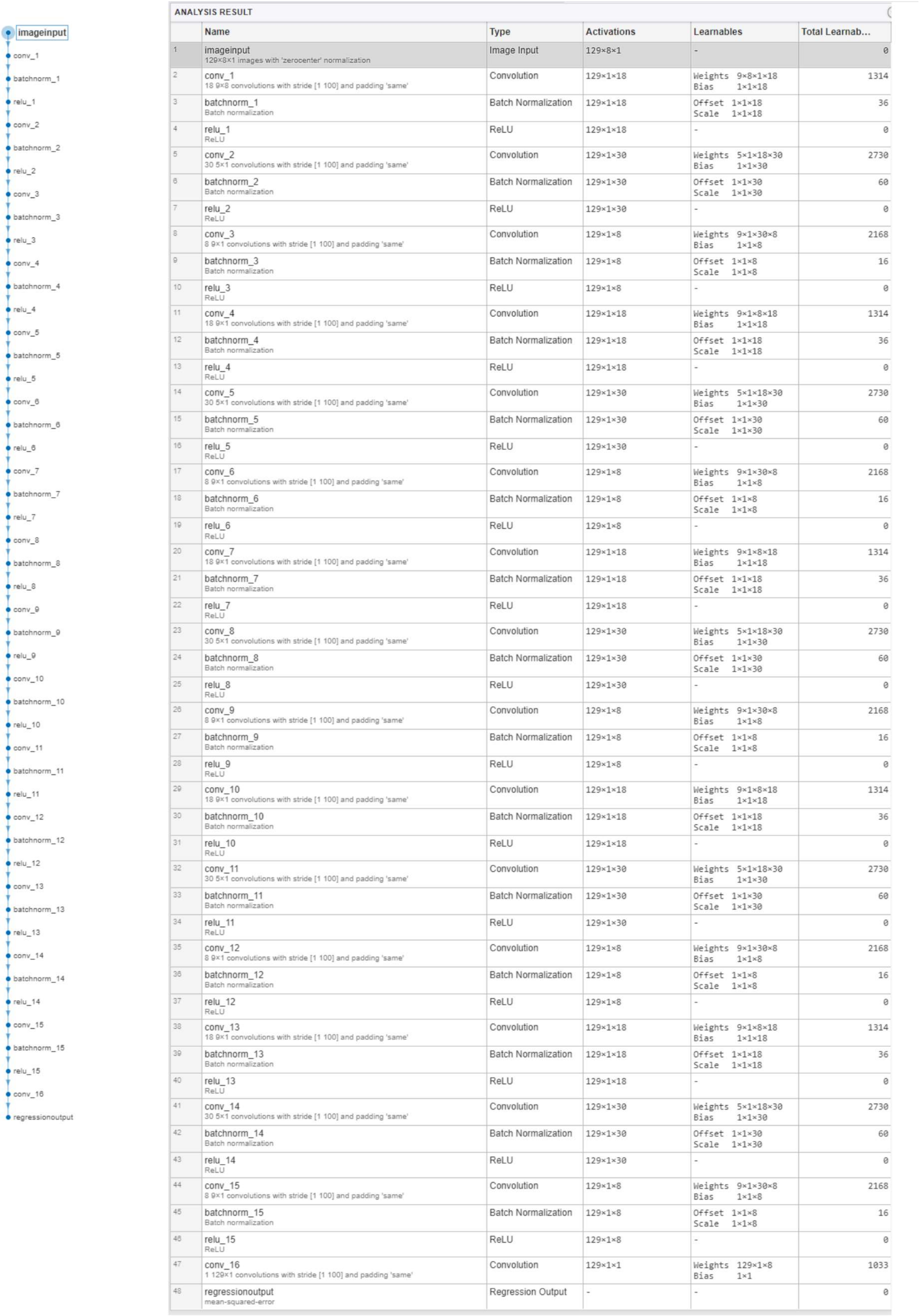


Figure 3.10: Predictor and target frames [65]

Targets and predictors are then processed by CNN with the Regression layer as the last one. The regression layer returns a half-mean-squared error of the predicted responses, which are used in the K-step ahead prediction algorithm. This algorithm takes two parameters, half-mean-squared errors, and predictors. Resulting in predicted frequency magnitudes of denoised STFT frame of a segment of a signal. Frequency magnitudes and extracted angles are combined together to form an array of complex numbers. This is done by exponential form of complex numbers $r\angle\theta = re^{j\theta}$. Where r is frequency magnitude and θ is extracted angle.

To go from the frequency domain back to the time-domain Inverse STFT is performed on converted complex numbers array. ISTFT parameters are identical to STFT ones. The result of ISTFT is a



ANALYSIS RESULT

	Name	Type	Activations	Learnables	Total Learnab...
1	imageinput 129×8×1 images with 'zerocenter' normalization	Image Input	129×8×1	-	0
2	conv_1 18 9×8 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×18	Weights 9×8×1×18 Bias 1×1×18	1314
3	batchnorm_1 Batch normalization	Batch Normalization	129×1×18	Offset 1×1×18 Scale 1×1×18	36
4	relu_1 ReLU	ReLU	129×1×18	-	0
5	conv_2 30 5×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×30	Weights 5×1×18×30 Bias 1×1×30	2730
6	batchnorm_2 Batch normalization	Batch Normalization	129×1×30	Offset 1×1×30 Scale 1×1×30	60
7	relu_2 ReLU	ReLU	129×1×30	-	0
8	conv_3 8 9×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×8	Weights 9×1×30×8 Bias 1×1×8	2168
9	batchnorm_3 Batch normalization	Batch Normalization	129×1×8	Offset 1×1×8 Scale 1×1×8	16
10	relu_3 ReLU	ReLU	129×1×8	-	0
11	conv_4 18 9×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×18	Weights 9×1×8×18 Bias 1×1×18	1314
12	batchnorm_4 Batch normalization	Batch Normalization	129×1×18	Offset 1×1×18 Scale 1×1×18	36
13	relu_4 ReLU	ReLU	129×1×18	-	0
14	conv_5 30 5×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×30	Weights 5×1×18×30 Bias 1×1×30	2730
15	batchnorm_5 Batch normalization	Batch Normalization	129×1×30	Offset 1×1×30 Scale 1×1×30	60
16	relu_5 ReLU	ReLU	129×1×30	-	0
17	conv_6 8 9×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×8	Weights 9×1×30×8 Bias 1×1×8	2168
18	batchnorm_6 Batch normalization	Batch Normalization	129×1×8	Offset 1×1×8 Scale 1×1×8	16
19	relu_6 ReLU	ReLU	129×1×8	-	0
20	conv_7 18 9×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×18	Weights 9×1×8×18 Bias 1×1×18	1314
21	batchnorm_7 Batch normalization	Batch Normalization	129×1×18	Offset 1×1×18 Scale 1×1×18	36
22	relu_7 ReLU	ReLU	129×1×18	-	0
23	conv_8 30 5×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×30	Weights 5×1×18×30 Bias 1×1×30	2730
24	batchnorm_8 Batch normalization	Batch Normalization	129×1×30	Offset 1×1×30 Scale 1×1×30	60
25	relu_8 ReLU	ReLU	129×1×30	-	0
26	conv_9 8 9×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×8	Weights 9×1×30×8 Bias 1×1×8	2168
27	batchnorm_9 Batch normalization	Batch Normalization	129×1×8	Offset 1×1×8 Scale 1×1×8	16
28	relu_9 ReLU	ReLU	129×1×8	-	0
29	conv_10 18 9×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×18	Weights 9×1×8×18 Bias 1×1×18	1314
30	batchnorm_10 Batch normalization	Batch Normalization	129×1×18	Offset 1×1×18 Scale 1×1×18	36
31	relu_10 ReLU	ReLU	129×1×18	-	0
32	conv_11 30 5×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×30	Weights 5×1×18×30 Bias 1×1×30	2730
33	batchnorm_11 Batch normalization	Batch Normalization	129×1×30	Offset 1×1×30 Scale 1×1×30	60
34	relu_11 ReLU	ReLU	129×1×30	-	0
35	conv_12 8 9×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×8	Weights 9×1×30×8 Bias 1×1×8	2168
36	batchnorm_12 Batch normalization	Batch Normalization	129×1×8	Offset 1×1×8 Scale 1×1×8	16
37	relu_12 ReLU	ReLU	129×1×8	-	0
38	conv_13 18 9×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×18	Weights 9×1×8×18 Bias 1×1×18	1314
39	batchnorm_13 Batch normalization	Batch Normalization	129×1×18	Offset 1×1×18 Scale 1×1×18	36
40	relu_13 ReLU	ReLU	129×1×18	-	0
41	conv_14 30 5×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×30	Weights 5×1×18×30 Bias 1×1×30	2730
42	batchnorm_14 Batch normalization	Batch Normalization	129×1×30	Offset 1×1×30 Scale 1×1×30	60
43	relu_14 ReLU	ReLU	129×1×30	-	0
44	conv_15 8 9×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×8	Weights 9×1×30×8 Bias 1×1×8	2168
45	batchnorm_15 Batch normalization	Batch Normalization	129×1×8	Offset 1×1×8 Scale 1×1×8	16
46	relu_15 ReLU	ReLU	129×1×8	-	0
47	conv_16 1 129×1 convolutions with stride [1 100] and padding 'same'	Convolution	129×1×1	Weights 129×1×8 Bias 1×1	1033
48	regressionoutput mean-squared-error	Regression Output	-	-	0

Figure 3.11: Analysis report of denoising network

chain of frames which is a denoised segment of a signal. To get the denoised signal itself, all of the segments are simply summed together.

3.2.2 LPC Cepstrum Distance

Linear Predictive Coding Cepstrum Distance methods are used as an objective measurement of denoising quality. This is done by comparing denoised and noise-free (clean) signals and returning a score between 0 and 10, where 0 is a perfect match and 10 is no match at all. The algorithm first calculated LPC coefficients for both clean and denoised signals, then converted these coefficients to Cepstrum coefficients. The score is calculated based on the distance between clean and denoised Cepstrum coefficients [66].

More about Linear Predictive Coding can be found at [67]. Cepstrum and its applications [68]. And LPC Cepstrum Distance as an objective measure [69].

Chapter 4

Results

This chapter is divided into two parts, classification of noise patterns and denoising. Several tests are performed to determine the quality of chosen methods.

4.1 Noise Classification

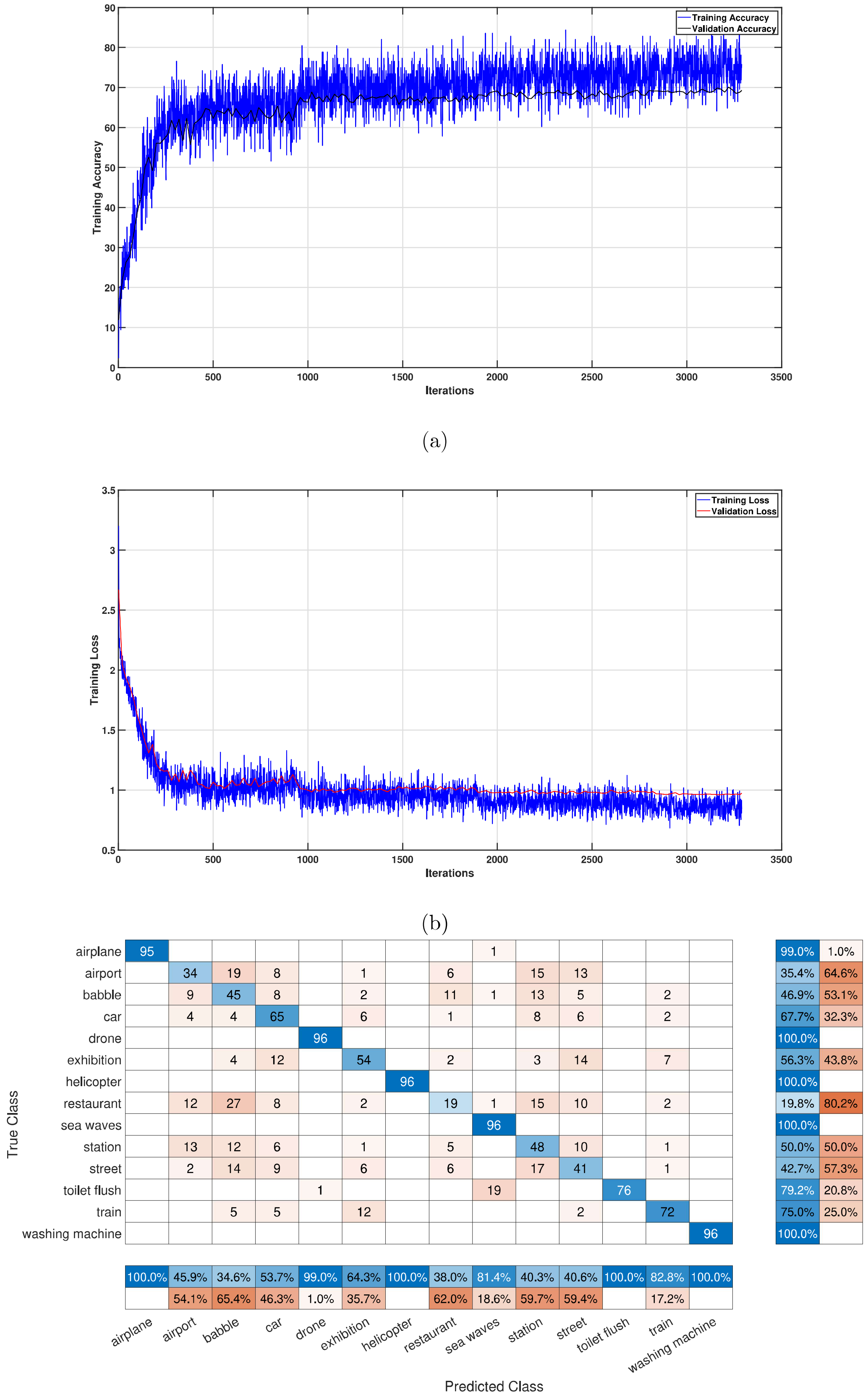
Signals that are used in noise classification testing are of different SNRs. Chosen SNR are 10, 0 and -5 dB. More weight is put on 10dB signal classification tests. This is due to the network struggling to correctly classify noise patterns at that SNR. Two methods are then tested to decrease classification failure. Increase the number of segments and increase/decrease the size of the frames.

The default size of the frames is chosen to be 1024. This size is used at every SNR testing. Frame sizes 512, 1024, and 2048 are used at 10dB SNR testing. Every test are done for both 50 and 100 segments per frame.

Table 4.1: Classification test accuracy values for different SNR, samples per frame and amount of segments.

SNR (dB)	Segments per frame	Frame Size	Test Accuracy (%)
10	50	512	69.42
10	100	512	71.58
10	50	1024	75.42
10	100	1024	77.68
10	50	2048	75.31
10	100	2048	77.98
0	50	1024	85.71
0	100	1024	88.84
-5	50	1024	100.00
-5	100	1024	100.00

Below the reader will find 10 figures of classification network training and its results. The order of figures is the same as in table (4.1).



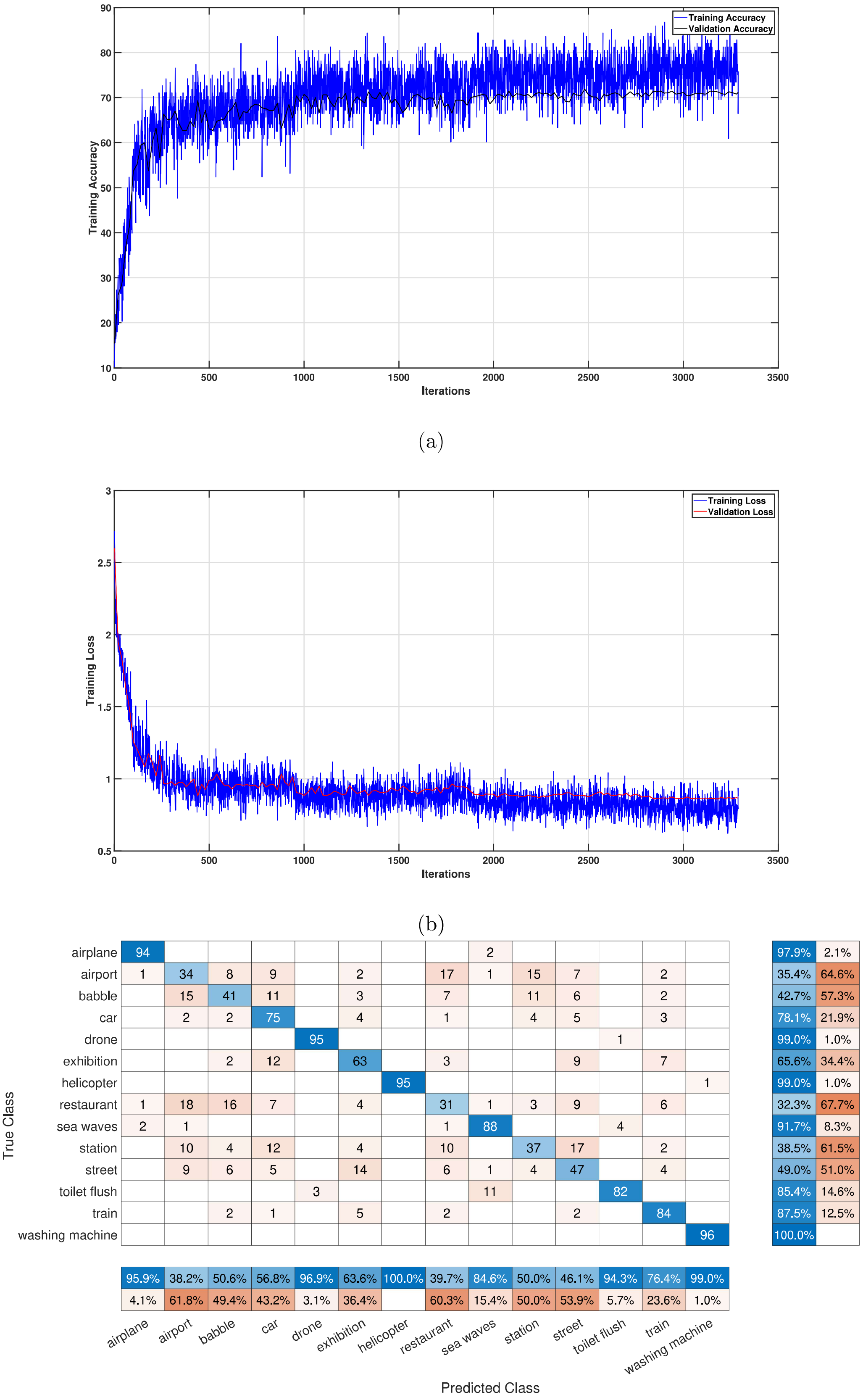
(a)

(b)

	airplane	airport	babble	car	drone	exhibition	helicopter	restaurant	sea waves	station	street	toilet flush	train	washing machine		
airplane	95								1						99.0%	1.0%
airport		34	19	8		1		6		15	13				35.4%	64.6%
babble		9	45	8		2		11	1	13	5		2		46.9%	53.1%
car		4	4	65		6		1		8	6		2		67.7%	32.3%
drone					96										100.0%	
exhibition			4	12		54		2		3	14		7		56.3%	43.8%
helicopter							96								100.0%	
restaurant		12	27	8		2		19	1	15	10		2		19.8%	80.2%
sea waves									96						100.0%	
station		13	12	6		1		5		48	10		1		50.0%	50.0%
street		2	14	9		6		6		17	41		1		42.7%	57.3%
toilet flush					1				19			76			79.2%	20.8%
train			5	5		12					2		72		75.0%	25.0%
washing machine														96	100.0%	
	100.0%	45.9%	34.6%	53.7%	99.0%	64.3%	100.0%	38.0%	81.4%	40.3%	40.6%	100.0%	82.8%	100.0%		
		54.1%	65.4%	46.3%	1.0%	35.7%		62.0%	18.6%	59.7%	59.4%		17.2%			

(c)

Figure 4.1: Training (a), Loss (b), and Confusion Matrix (c) results at 10dB SNR, 50 segments per frame, and 512 samples per frame.

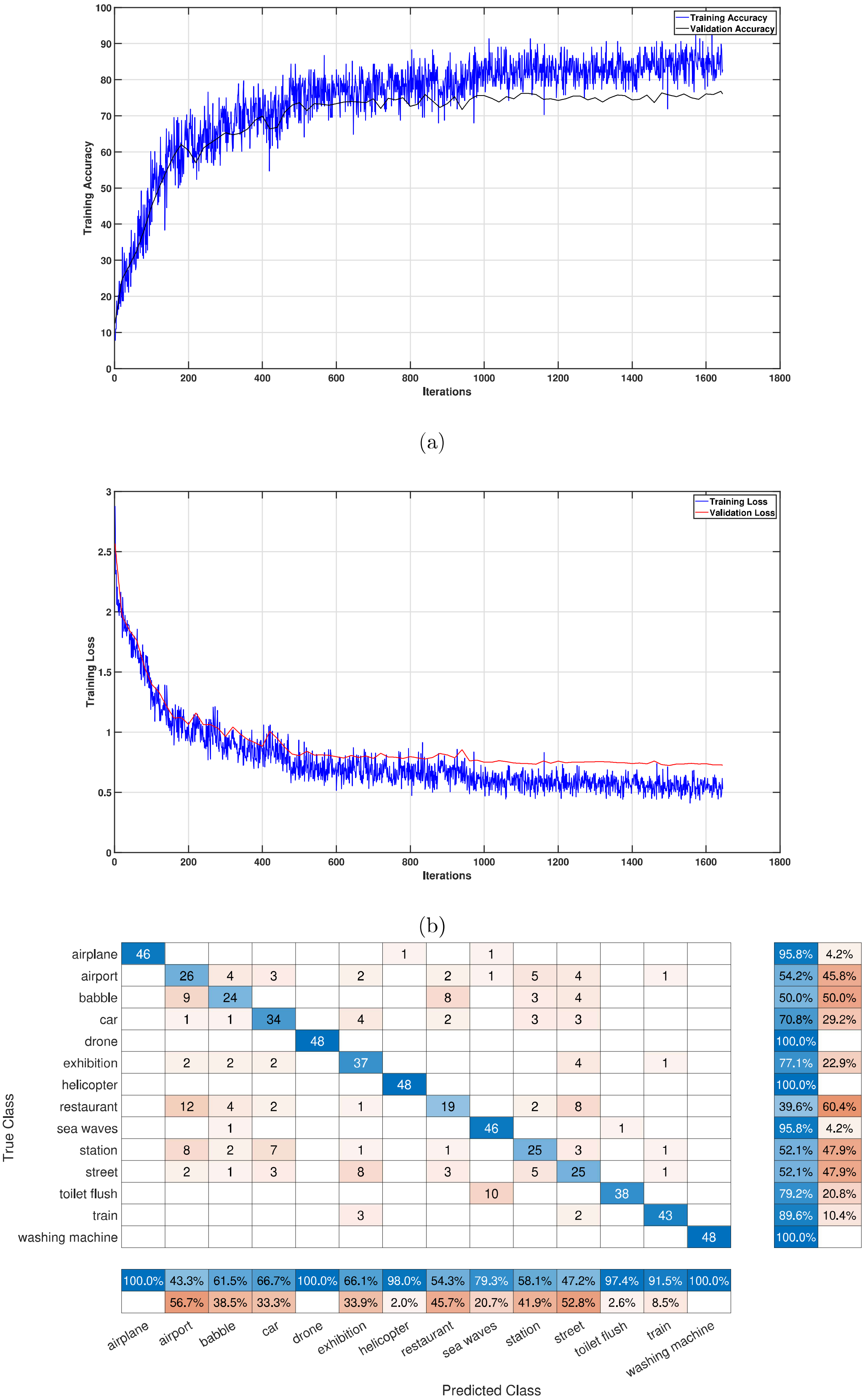


(a)

(b)

(c)

Figure 4.2: Training (a), Loss (b), and Confusion Matrix (c) results at 10dB SNR, 100 segments per frame, and 512 samples per frame.



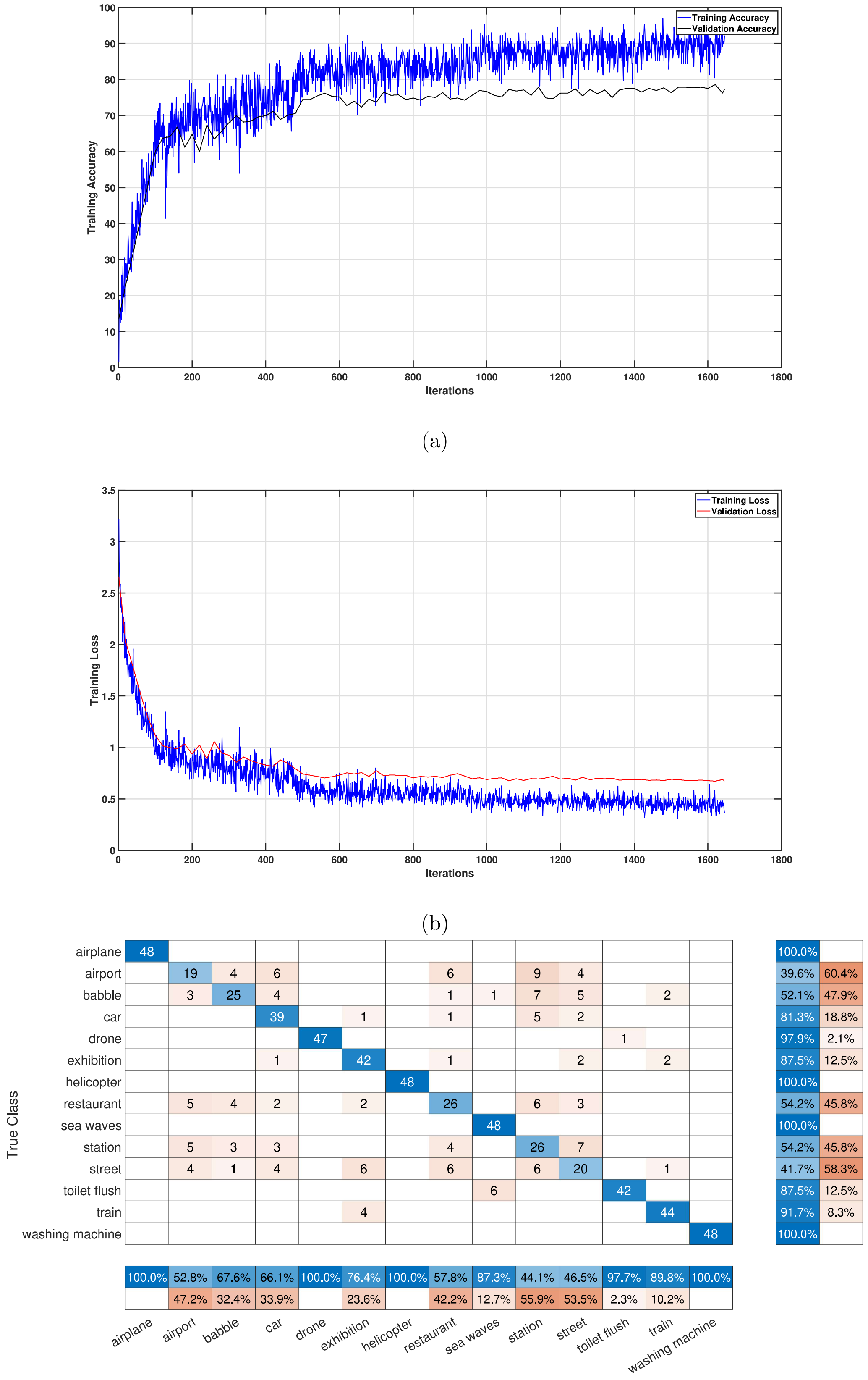
(a)

(b)

	airplane	airport	babble	car	drone	exhibition	helicopter	restaurant	sea waves	station	street	toilet flush	train	washing machine		
airplane	46						1		1						95.8%	4.2%
airport		26	4	3		2		2	1	5	4		1		54.2%	45.8%
babble		9	24					8		3	4				50.0%	50.0%
car		1	1	34		4		2		3	3				70.8%	29.2%
drone					48										100.0%	
exhibition		2	2	2		37					4		1		77.1%	22.9%
helicopter							48								100.0%	
restaurant		12	4	2		1		19		2	8				39.6%	60.4%
sea waves			1						46			1			95.8%	4.2%
station		8	2	7		1		1		25	3		1		52.1%	47.9%
street		2	1	3		8		3		5	25		1		52.1%	47.9%
toilet flush									10			38			79.2%	20.8%
train						3					2		43		89.6%	10.4%
washing machine														48	100.0%	
	100.0%	43.3%	61.5%	66.7%	100.0%	66.1%	98.0%	54.3%	79.3%	58.1%	47.2%	97.4%	91.5%	100.0%		
		56.7%	38.5%	33.3%		33.9%	2.0%	45.7%	20.7%	41.9%	52.8%	2.6%	8.5%			

(c)

Figure 4.3: Training (a), Loss (b), and Confusion Matrix (c) results at 10dB SNR, 50 segments per frame, and 1024 samples per frame.



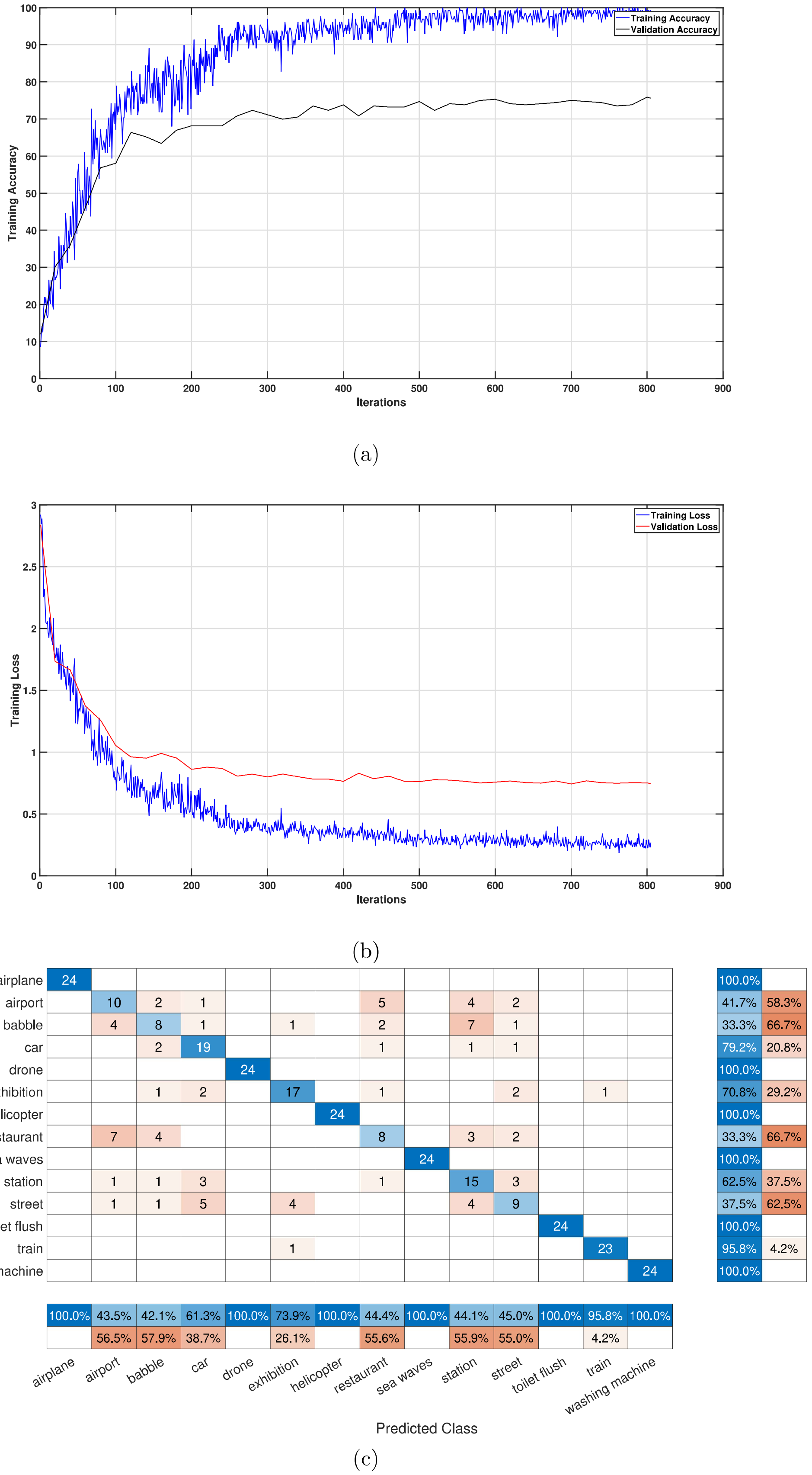
(a)

(b)

True Class	airplane	airport	babble	car	drone	exhibition	helicopter	restaurant	sea waves	station	street	toilet flush	train	washing machine		
airplane	48														100.0%	
airport		19	4	6				6		9	4				39.6%	60.4%
babble		3	25	4				1	1	7	5		2		52.1%	47.9%
car				39		1		1		5	2				81.3%	18.8%
drone					47							1			97.9%	2.1%
exhibition				1		42		1			2		2		87.5%	12.5%
helicopter							48								100.0%	
restaurant		5	4	2		2		26		6	3				54.2%	45.8%
sea waves									48						100.0%	
station		5	3	3				4		26	7				54.2%	45.8%
street		4	1	4		6		6		6	20		1		41.7%	58.3%
toilet flush									6			42			87.5%	12.5%
train						4							44		91.7%	8.3%
washing machine														48	100.0%	
	100.0%	52.8%	67.6%	66.1%	100.0%	76.4%	100.0%	57.8%	87.3%	44.1%	46.5%	97.7%	89.8%	100.0%		
		47.2%	32.4%	33.9%		23.6%		42.2%	12.7%	55.9%	53.5%	2.3%	10.2%			

(c)

Figure 4.4: Training (a), Loss (b), and Confusion Matrix (c) results at 10dB SNR, 100 segments per frame, and 1024 samples per frame.



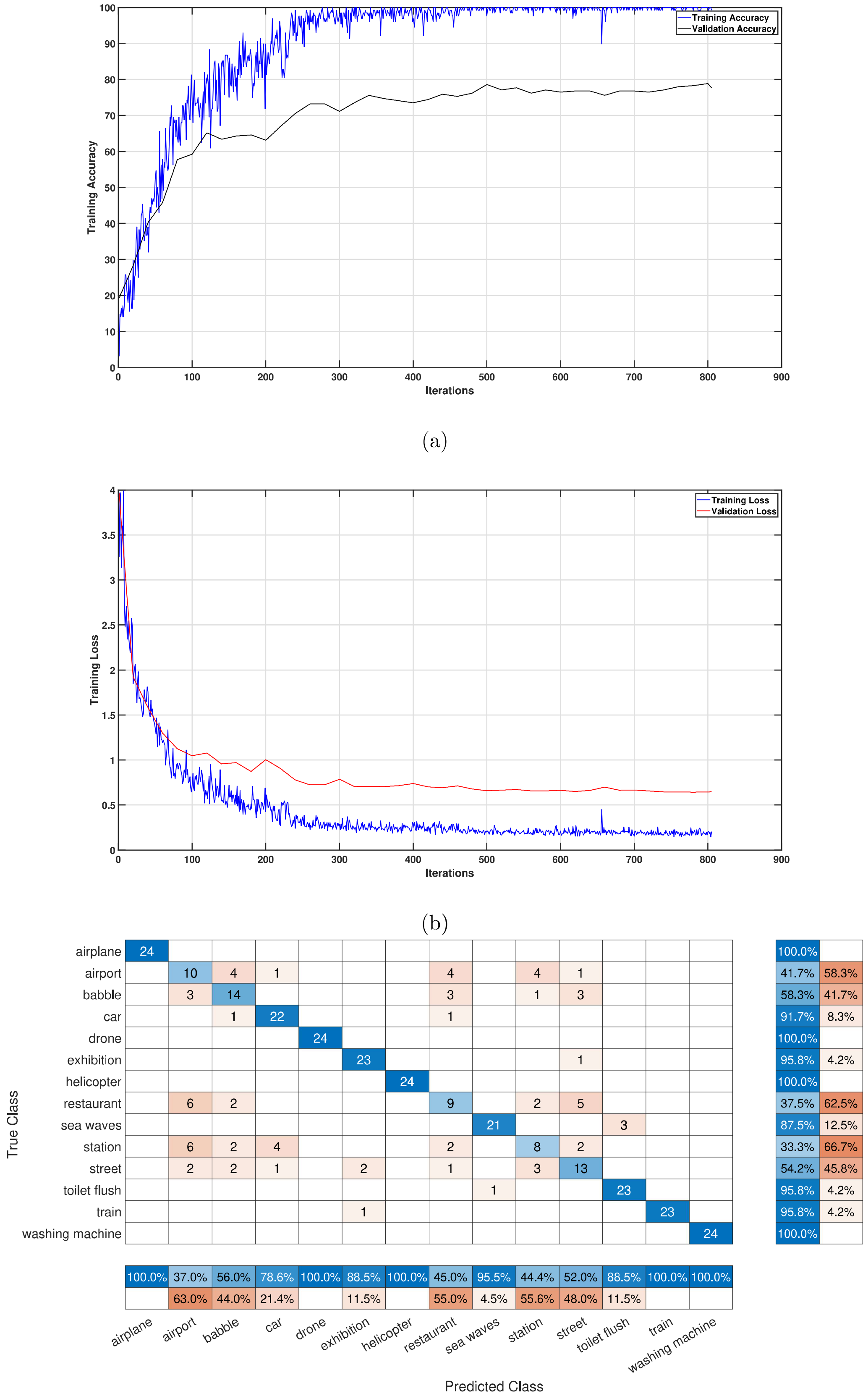
(a)

(b)

True Class	airplane	airport	babble	car	drone	exhibition	helicopter	restaurant	sea waves	station	street	toilet flush	train	washing machine		
airplane	24														100.0%	
airport		10	2	1				5		4	2				41.7%	58.3%
babble		4	8	1		1		2		7	1				33.3%	66.7%
car			2	19				1		1	1				79.2%	20.8%
drone					24										100.0%	
exhibition			1	2		17		1			2		1		70.8%	29.2%
helicopter							24								100.0%	
restaurant		7	4					8		3	2				33.3%	66.7%
sea waves									24						100.0%	
station		1	1	3				1		15	3				62.5%	37.5%
street		1	1	5		4				4	9				37.5%	62.5%
toilet flush												24			100.0%	
train						1							23		95.8%	4.2%
washing machine														24	100.0%	
	100.0%	43.5%	42.1%	61.3%	100.0%	73.9%	100.0%	44.4%	100.0%	44.1%	45.0%	100.0%	95.8%	100.0%		
		56.5%	57.9%	38.7%		26.1%		55.6%		55.9%	55.0%		4.2%			

(c)

Figure 4.5: Training (a), Loss (b), and Confusion Matrix (c) results at 10dB SNR, 50 segments per frame, and 2048 samples per frame.



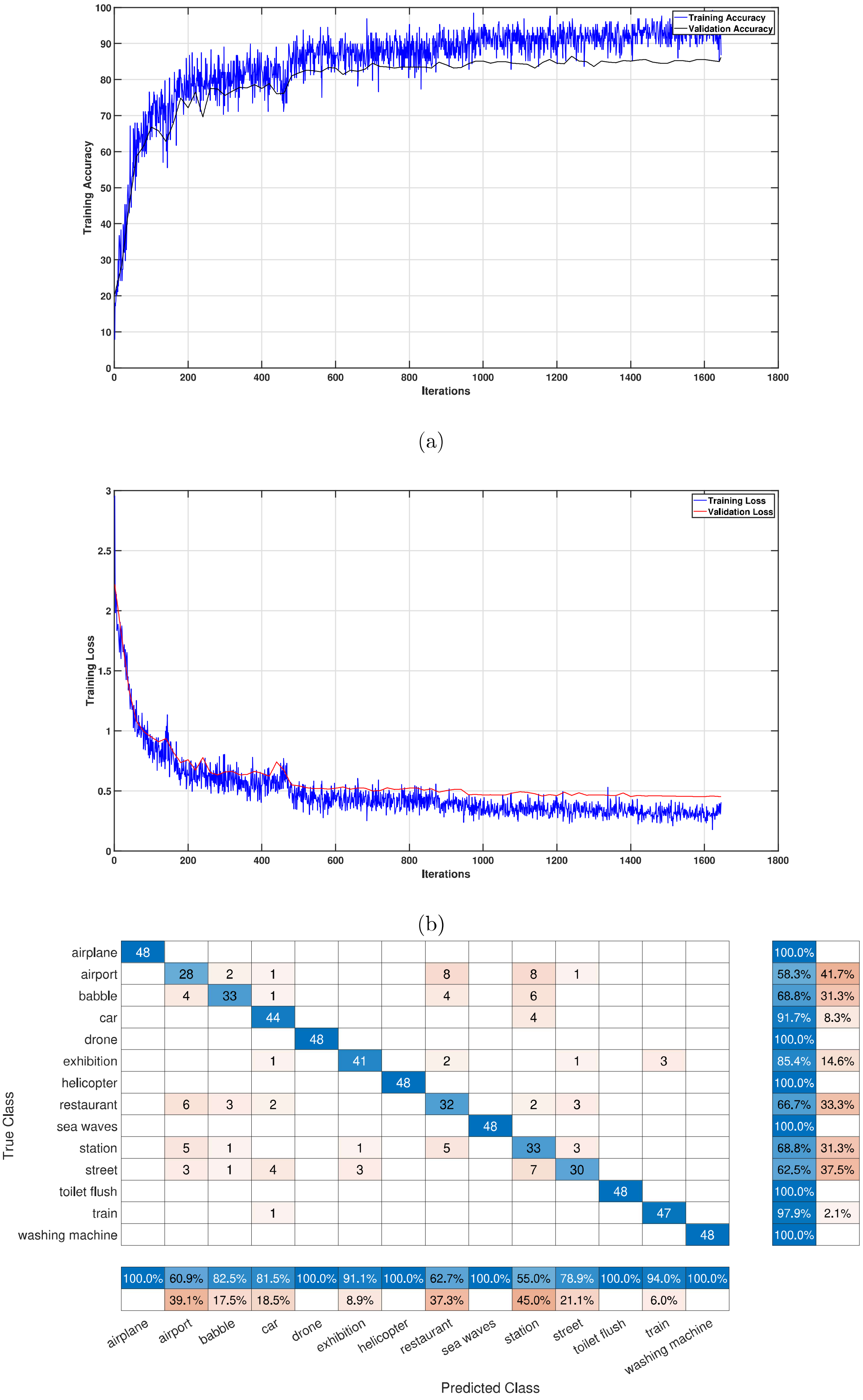
(a)

(b)

True Class \ Predicted Class	airplane	airport	babble	car	drone	exhibition	helicopter	restaurant	sea waves	station	street	toilet flush	train	washing machine		
airplane	24														100.0%	
airport		10	4	1				4		4	1				41.7%	58.3%
babble		3	14					3		1	3				58.3%	41.7%
car			1	22				1							91.7%	8.3%
drone					24										100.0%	
exhibition						23					1				95.8%	4.2%
helicopter							24								100.0%	
restaurant		6	2					9		2	5				37.5%	62.5%
sea waves									21			3			87.5%	12.5%
station		6	2	4				2		8	2				33.3%	66.7%
street		2	2	1		2		1		3	13				54.2%	45.8%
toilet flush									1			23			95.8%	4.2%
train						1							23		95.8%	4.2%
washing machine														24	100.0%	
	100.0%	37.0%	56.0%	78.6%	100.0%	88.5%	100.0%	45.0%	95.5%	44.4%	52.0%	88.5%	100.0%	100.0%		
		63.0%	44.0%	21.4%		11.5%		55.0%	4.5%	55.6%	48.0%	11.5%				

(c)

Figure 4.6: Training (a), Loss (b), and Confusion Matrix (c) results at 10dB SNR, 100 segments per frame, and 2048 samples per frame.



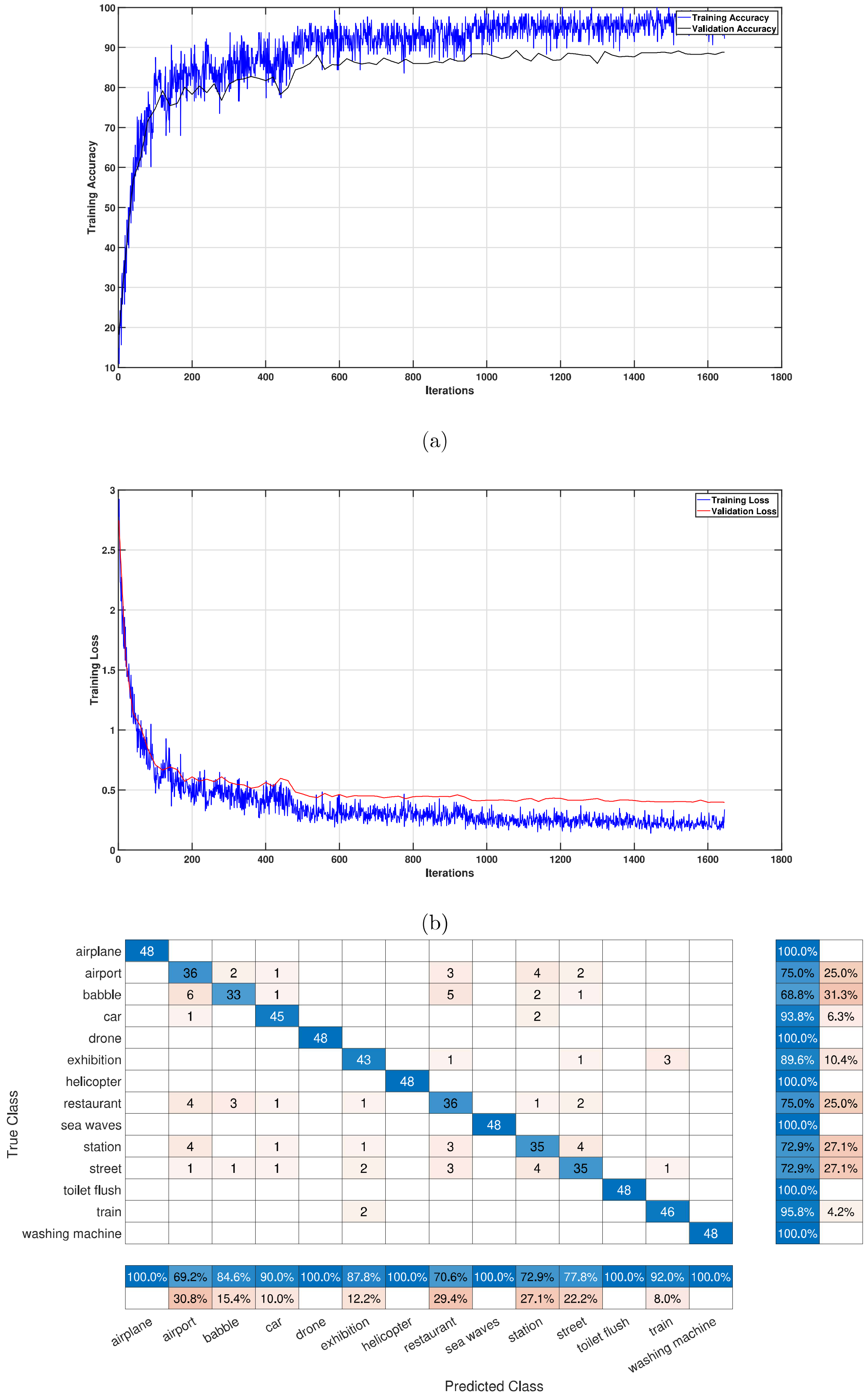
(a)

(b)

True Class	airplane	airport	babble	car	drone	exhibition	helicopter	restaurant	sea waves	station	street	toilet flush	train	washing machine		
airplane	48														100.0%	
airport		28	2	1				8		8	1				58.3%	41.7%
babble		4	33	1				4		6					68.8%	31.3%
car				44						4					91.7%	8.3%
drone					48										100.0%	
exhibition				1		41		2			1		3		85.4%	14.6%
helicopter							48								100.0%	
restaurant		6	3	2				32		2	3				66.7%	33.3%
sea waves									48						100.0%	
station		5	1			1		5		33	3				68.8%	31.3%
street		3	1	4		3				7	30				62.5%	37.5%
toilet flush												48			100.0%	
train				1									47		97.9%	2.1%
washing machine														48	100.0%	
	100.0%	60.9%	82.5%	81.5%	100.0%	91.1%	100.0%	62.7%	100.0%	55.0%	78.9%	100.0%	94.0%	100.0%		
		39.1%	17.5%	18.5%		8.9%		37.3%		45.0%	21.1%		6.0%			

(c)

Figure 4.7: Training (a), Loss (b), and Confusion Matrix (c) results at 0dB SNR, 50 segments per frame, and 1024 samples per frame.

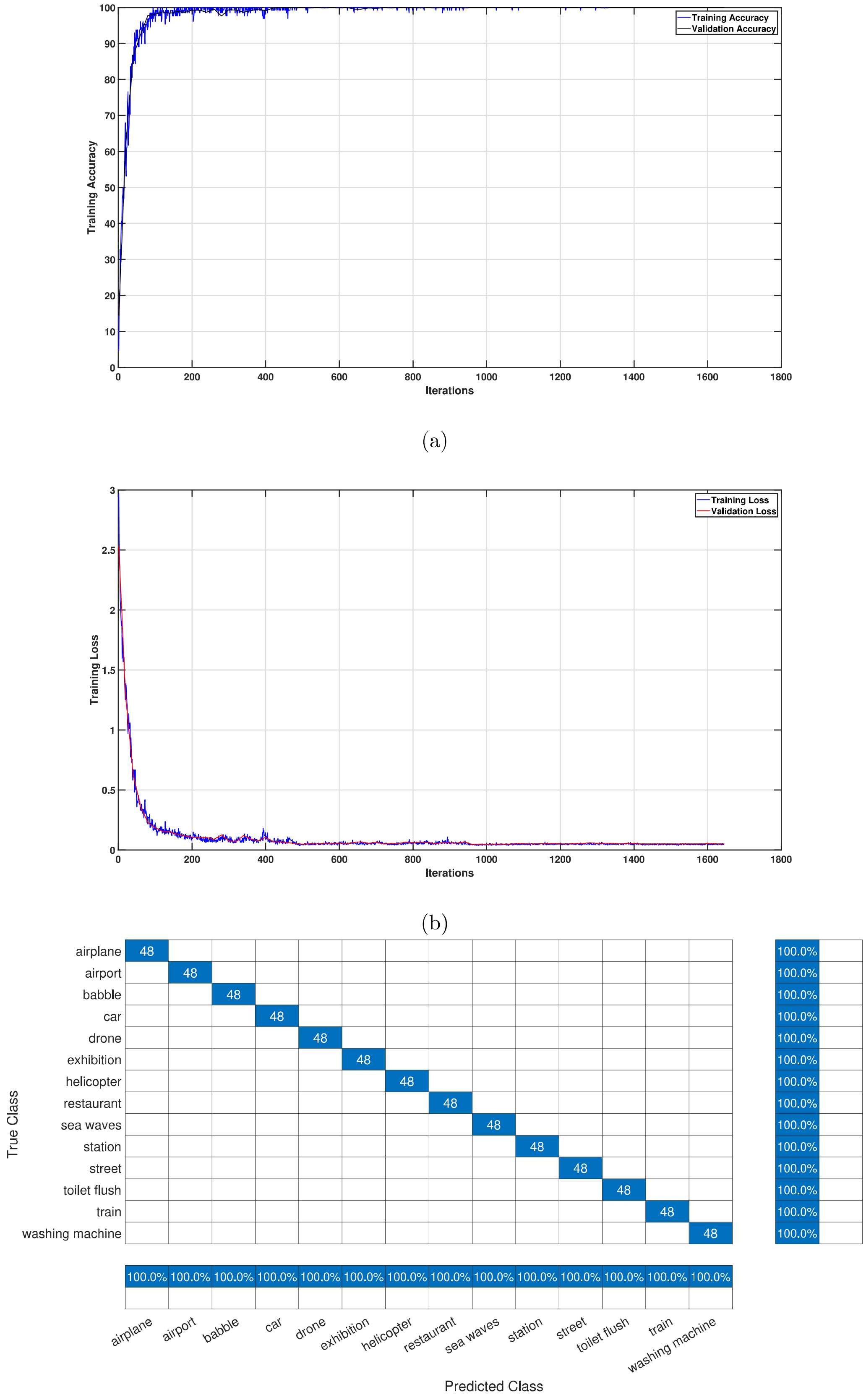


(a)

(b)

(c)

Figure 4.8: Training (a), Loss (b), and Confusion Matrix (c) results at 0dB SNR, 100 segments per frame, and 1024 samples per frame.

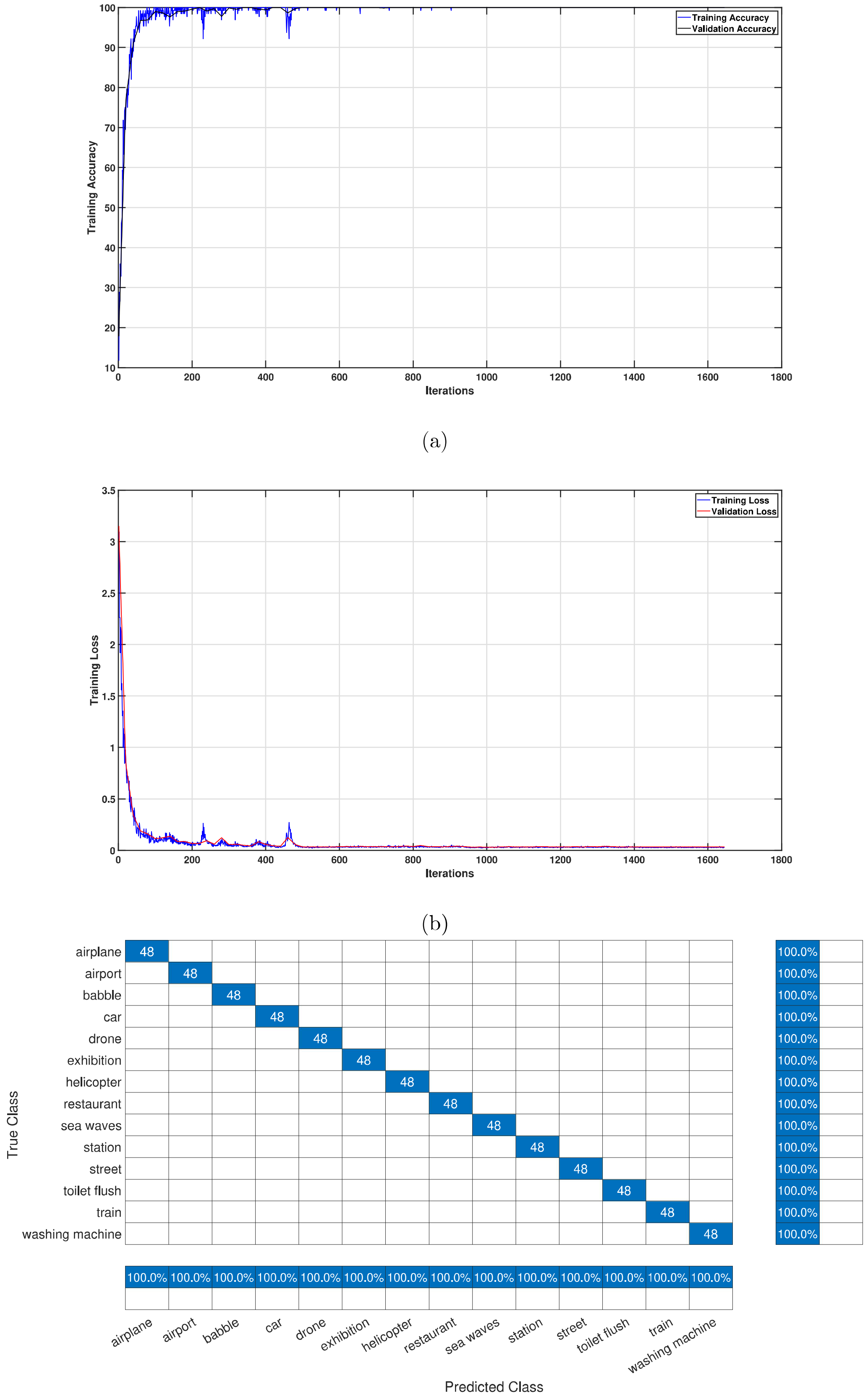


(a)

(b)

(c)

Figure 4.9: Training (a), Loss (b), and Confusion Matrix (c) results at -5dB SNR, 50 segments per frame, and 1024 samples per frame.



(a)

(b)

(c)

Figure 4.10: Training (a), Loss (b), and Confusion Matrix (c) results at -5dB SNR, 100 segments per frame and 1024 samples per frame.

4.2 Denoising

For denoising 3 classes of noise have been chosen, one for each type. For stationary noise type, it is airplane noise. For non-stationary it is drone noise. And for babble type, it is babble noise. Chosen SNR are 0, -5 and -10 dB. More weight is put on -10 dB signals. This is due to the denoising network starting to struggle. One method is tested to improve the quality of denoising. Increase number of segments.

LPC Cepstrum Distance is used as a metric for denoising quality. Due to the probabilistic nature of denoising network testing may vary up to 20% between each consecutive test. Therefore each test is performed 10 times and results are averaged.

The test is performed on no segmentation, 5 segments, 10 and 15 segments signals. Where 10 and 15 segments are only done for -10 dB SNR. Results are compared to find if signal decomposition improves the quality of denoising.

Even though increasing segments depict a decrease in metric quality, subjectively (by listening) signals denoised with 15 segments sound better than those with 5 segments. Babble noise is very difficult for CNN to denoise, processed signals at -10 dB are not comprehensible.

4.2.1 Airplane noise

Table 4.2: Denoising of Airplane class noise. Comparison between no EWT decomposition and 5 segments decomposition, based on averaged LPC CD coefficients. SNR 0 dB (a), SNR -5 dB (b), SNR -10 dB (c)

(a)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.69196	5.15678	22.94066	18.52538
2	5.56338	4.57319	17.79835	
3	6.4123	4.95135	22.78356	
4	6.01614	4.92341	18.16331	
5	5.64775	4.5246	19.88668	
6	5.84099	4.76392	18.43985	
7	5.80184	4.58657	20.94629	
8	5.6885	4.76491	16.23609	
9	5.87386	5.0368	14.2506	
10	6.01348	4.92392	18.11863	
11	6.01428	4.92242	18.15446	
12	5.36326	4.47164	16.62459	
13	6.17317	4.87162	21.08398	
14	6.23178	5.0707	18.63159	
15	5.70623	4.58737	19.60769	
16	5.3404	4.41421	17.34308	
17	5.35472	4.44089	17.06588	
18	5.30133	4.59871	13.25366	
19	6.24579	5.17228	17.18774	
20	5.52188	4.74649	14.04214	
21	5.96199	4.69401	21.26773	
22	6.25723	4.99173	20.2246	
23	6.16356	4.89583	20.56815	
24	7.51891	6.20148	17.52156	
25	6.69714	5.50145	17.85374	
26	5.47039	4.47716	18.15648	
27	5.70591	4.51376	20.89325	
28	5.25016	4.25036	19.04323	
29	5.64766	4.66833	17.34046	
30	5.54524	4.41771	20.33329	

(b)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.91916	5.48417	20.73937	17.25592
2	5.80327	4.88034	15.90362	
3	6.76724	5.46728	19.2096	
4	6.14542	5.11915	16.69975	
5	5.85814	4.86096	17.02213	
6	6.16258	5.14832	16.45837	
7	5.95252	4.9278	17.21489	
8	5.8751	4.9307	16.07462	
9	6.16322	5.17768	15.99067	
10	6.12634	5.21096	14.94171	
11	6.18126	5.09738	17.53494	
12	5.58113	4.59333	17.69892	
13	6.30895	5.09915	19.17593	
14	6.51399	5.38933	17.2653	
15	6.03323	4.89618	18.84646	
16	5.52295	4.43241	19.74561	
17	5.62732	4.69366	16.59156	
18	5.4516	4.82936	11.4139	
19	6.40929	5.30567	17.21907	
20	5.81191	5.00978	13.80149	
21	6.12239	4.67442	23.6504	
22	6.34902	5.34614	15.79582	
23	6.30762	5.19131	17.6978	
24	7.72838	6.46773	16.31196	
25	6.78611	5.53934	18.37238	
26	5.71563	4.80592	15.91618	
27	5.92939	4.82824	18.57105	
28	5.43814	4.45413	18.09461	
29	5.82381	4.89861	15.88651	
30	5.77153	4.7423	17.83288	

(c)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.61695	5.84265	11.70177	8.982815
2	5.69786	5.21811	8.419828	
3	6.71831	6.09096	9.337914	
4	5.9179	5.39936	8.76223	
5	5.7239	5.31891	7.075421	
6	6.00657	5.55744	7.477312	
7	5.93706	5.31856	10.41761	
8	5.59583	5.18	7.431069	
9	6.09594	5.54487	9.039951	
10	5.97124	5.64767	5.418807	
11	5.90591	5.3196	9.927513	
12	5.41413	4.90797	9.34887	
13	5.90355	5.32526	9.795631	
14	6.43419	5.7812	10.14875	
15	6.01998	5.3665	10.85519	
16	5.27828	4.63815	12.12762	
17	5.60767	5.0808	9.395524	
18	5.45661	5.10087	6.519432	
19	6.1926	5.58448	9.820108	
20	5.77266	5.35489	7.237045	
21	5.65106	4.83407	14.45729	
22	6.23896	5.73398	8.093977	
23	6.09877	5.56275	8.788985	
24	7.37549	6.60881	10.39497	
25	6.41884	5.71559	10.95603	
26	5.57322	5.15407	7.520787	
27	5.67895	5.25497	7.465817	
28	5.20083	4.77491	8.189462	
29	5.61021	5.28412	5.812438	
30	5.553	5.13391	7.547092	

Table 4.3: Denoising of Airplane class noise SNR -10 dB. Comparison between no EWT decomposition and 5, 10 and 15 segments decomposition, based on averaged LPC CD coefficients. 5 segments (a), 10 segments (b), 15 segments (c)

(a)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.61695	5.84265	11.70177	8.982815
2	5.69786	5.21811	8.419828	
3	6.71831	6.09096	9.337914	
4	5.9179	5.39936	8.76223	
5	5.7239	5.31891	7.075421	
6	6.00657	5.55744	7.477312	
7	5.93706	5.31856	10.41761	
8	5.59583	5.18	7.431069	
9	6.09594	5.54487	9.039951	
10	5.97124	5.64767	5.418807	
11	5.90591	5.3196	9.927513	
12	5.41413	4.90797	9.34887	
13	5.90355	5.32526	9.795631	
14	6.43419	5.7812	10.14875	
15	6.01998	5.3665	10.85519	
16	5.27828	4.63815	12.12762	
17	5.60767	5.0808	9.395524	
18	5.45661	5.10087	6.519432	
19	6.1926	5.58448	9.820108	
20	5.77266	5.35489	7.237045	
21	5.65106	4.83407	14.45729	
22	6.23896	5.73398	8.093977	
23	6.09877	5.56275	8.788985	
24	7.37549	6.60881	10.39497	
25	6.41884	5.71559	10.95603	
26	5.57322	5.15407	7.520787	
27	5.67895	5.25497	7.465817	
28	5.20083	4.77491	8.189462	
29	5.61021	5.28412	5.812438	
30	5.553	5.13391	7.547092	

(b)

Tr.	Single Track	10 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.61695	6.24986	5.547722	4.657982
2	5.69786	5.45035	4.343912	
3	6.71831	6.39954	4.744794	
4	5.9179	5.64579	4.598084	
5	5.7239	5.49255	4.041825	
6	6.00657	5.62258	6.392833	
7	5.93706	5.39888	9.064756	
8	5.59583	5.20996	6.895671	
9	6.09594	5.49555	9.849014	
10	5.97124	5.75377	3.641957	
11	5.90591	5.72058	3.138043	
12	5.41413	5.1811	4.304108	
13	5.90355	5.66805	3.989125	
14	6.43419	6.11268	4.996899	
15	6.01998	5.70308	5.264137	
16	5.27828	4.92723	6.650841	
17	5.60767	5.34408	4.700526	
18	5.45661	5.34857	1.979984	
19	6.1926	5.88794	4.919743	
20	5.77266	5.62846	2.497982	
21	5.65106	5.29536	6.294394	
22	6.23896	5.99149	3.966526	
23	6.09877	5.8204	4.564363	
24	7.37549	7.06991	4.143182	
25	6.41884	6.00361	6.468926	
26	5.57322	5.33077	4.350268	
27	5.67895	5.56231	2.053901	
28	5.20083	5.0569	2.767443	
29	5.61021	5.52961	1.436666	
30	5.553	5.43462	2.131821	

(c)

Tr.	Single Track	15 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.61695	6.3287	4.356237	4.296816
2	5.69786	5.36811	5.78726	
3	6.71831	6.48932	3.408446	
4	5.9179	5.71864	3.367073	
5	5.7239	5.49226	4.046891	
6	6.00657	5.59026	6.930911	
7	5.93706	5.47555	7.773376	
8	5.59583	5.2692	5.837025	
9	6.09594	5.35323	12.18368	
10	5.97124	5.82524	2.445053	
11	5.90591	5.77916	2.146155	
12	5.41413	5.185	4.232074	
13	5.90355	5.75558	2.506458	
14	6.43419	6.07947	5.513048	
15	6.01998	5.68316	5.595035	
16	5.27828	4.75467	9.920088	
17	5.60767	5.2118	7.059438	
18	5.45661	5.33885	2.158116	
19	6.1926	5.88103	5.031328	
20	5.77266	5.58285	3.288086	
21	5.65106	5.27925	6.579474	
22	6.23896	6.0184	3.535205	
23	6.09877	5.9031	3.208352	
24	7.37549	7.28059	1.286694	
25	6.41884	6.04818	5.774564	
26	5.57322	5.46795	1.888854	
27	5.67895	5.69676	-0.31361	
28	5.20083	5.10163	1.907388	
29	5.61021	5.57268	0.668959	
30	5.553	5.50953	0.78282	

4.2.2 Drone Noise

Table 4.4: Denoising of Drone class noise. Comparison between no EWT decomposition and 5 segments decomposition, based on averaged LPC CD coefficients. SNR 0 dB (a), SNR -5 dB (b), SNR -10 dB (c)

(a)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.79373	6.3806	6.081048	6.614969
2	5.76135	5.47673	4.940162	
3	6.36991	5.94734	6.633846	
4	6.46731	6.00248	7.187378	
5	5.7417	5.25753	8.43252	
6	5.9906	5.70653	4.741929	
7	5.92899	5.40899	8.770465	
8	5.89087	5.67541	3.657524	
9	6.3257	6.20151	1.963261	
10	6.16034	5.86476	4.798112	
11	6.24575	5.75769	7.814274	
12	5.68101	5.22691	7.993297	
13	6.31568	5.82536	7.763535	
14	6.58063	5.98415	9.064178	
15	5.96372	5.52315	7.387503	
16	5.44671	5.03859	7.492964	
17	5.44064	5.02677	7.607009	
18	5.50126	5.14946	6.394899	
19	6.31303	5.73432	9.166914	
20	5.73013	5.3277	7.023052	
21	6.04855	5.62935	6.930587	
22	6.51848	6.23798	4.30315	
23	6.24304	5.8293	6.62722	
24	7.57333	7.33598	3.134024	
25	7.00823	6.66658	4.874983	
26	5.68563	5.32882	6.275646	
27	5.77229	5.29826	8.212165	
28	5.44593	5.18274	4.832783	
29	5.91332	5.5218	6.620984	
30	5.69711	5.0292	11.72366	

(b)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.80801	6.41535	5.767618	6.613458
2	6.0692	5.74274	5.378963	
3	6.69315	6.14187	8.236481	
4	6.82308	6.22995	8.692995	
5	5.95593	5.45326	8.439824	
6	6.2703	5.8549	6.624882	
7	6.1821	5.60271	9.372058	
8	6.1386	5.86153	4.51357	
9	6.75853	6.4894	3.982079	
10	6.37098	6.12703	3.829081	
11	6.32812	5.89933	6.775946	
12	5.93241	5.47347	7.736148	
13	6.27414	5.7916	7.690935	
14	6.8002	6.22972	8.389165	
15	6.2157	5.67547	8.691378	
16	5.44428	5.06805	6.910556	
17	5.64188	5.31429	5.806398	
18	5.69266	5.39263	5.270471	
19	6.46246	5.86204	9.290889	
20	6.05203	5.6316	6.946925	
21	5.97263	5.65141	5.3782	
22	6.59931	6.32634	4.136342	
23	6.31704	5.96325	5.600566	
24	7.56571	7.41143	2.039201	
25	6.90745	6.72454	2.64801	
26	6.19543	5.59953	9.61838	
27	5.96614	5.42736	9.03063	
28	5.61158	5.36871	4.328015	
29	6.10809	5.68955	6.852224	
30	5.94131	5.32188	10.42582	

(c)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	7.12944	6.45516	9.457685	9.984589
2	6.54657	5.94429	9.199932	
3	7.15616	6.26812	12.40945	
4	7.13066	6.32542	11.29264	
5	6.32223	5.54326	12.32113	
6	6.50901	5.96203	8.403428	
7	6.55688	5.79231	11.66058	
8	6.74314	6.24145	7.440006	
9	7.06923	6.77053	4.225354	
10	6.76403	6.34041	6.262834	
11	6.71289	5.93832	11.53855	
12	6.38989	5.81613	8.979184	
13	6.55119	5.63607	13.96876	
14	7.17213	6.44444	10.14608	
15	6.608	5.91484	10.48971	
16	5.85642	5.42478	7.370373	
17	6.21754	5.79328	6.823599	
18	6.25065	5.80798	7.081983	
19	6.79141	5.90911	12.99141	
20	6.57847	5.87006	10.76861	
21	6.43577	5.68877	11.607	
22	7.01618	6.35518	9.421081	
23	6.75988	6.00664	11.1428	
24	7.9693	7.47648	6.183981	
25	7.21215	6.82513	5.366222	
26	6.81276	5.96408	12.45721	
27	6.44801	5.53469	14.16437	
28	6.09463	5.44475	10.66316	
29	6.52842	5.81001	11.00435	
30	6.46255	5.5128	14.69621	

Table 4.5: Denoising of Drone class noise SNR -10 dB. Comparison between no EWT decomposition and 5, 10, and 15 segments decomposition, based on averaged LPC CD coefficients. 5 segments (a), 10 segments (b), 15 segments (c)

(a)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	7.12944	6.45516	9.457685	9.984589
2	6.54657	5.94429	9.199932	
3	7.15616	6.26812	12.40945	
4	7.13066	6.32542	11.29264	
5	6.32223	5.54326	12.32113	
6	6.50901	5.96203	8.403428	
7	6.55688	5.79231	11.66058	
8	6.74314	6.24145	7.440006	
9	7.06923	6.77053	4.225354	
10	6.76403	6.34041	6.262834	
11	6.71289	5.93832	11.53855	
12	6.38989	5.81613	8.979184	
13	6.55119	5.63607	13.96876	
14	7.17213	6.44444	10.14608	
15	6.608	5.91484	10.48971	
16	5.85642	5.42478	7.370373	
17	6.21754	5.79328	6.823599	
18	6.25065	5.80798	7.081983	
19	6.79141	5.90911	12.99141	
20	6.57847	5.87006	10.76861	
21	6.43577	5.68877	11.607	
22	7.01618	6.35518	9.421081	
23	6.75988	6.00664	11.1428	
24	7.9693	7.47648	6.183981	
25	7.21215	6.82513	5.366222	
26	6.81276	5.96408	12.45721	
27	6.44801	5.53469	14.16437	
28	6.09463	5.44475	10.66316	
29	6.52842	5.81001	11.00435	
30	6.46255	5.5128	14.69621	

(b)

Tr.	Single Track	10 Segments	Improvement per track (%)	Averaged imp. (%)
1	7.12944	6.6321	6.975863	8.784869
2	6.54657	5.9969	8.396305	
3	7.15616	6.30044	11.95781	
4	7.13066	6.12419	14.11468	
5	6.32223	5.53418	12.46475	
6	6.50901	5.93547	8.811478	
7	6.55688	5.80539	11.46109	
8	6.74314	6.08555	9.751985	
9	7.06923	6.58343	6.872036	
10	6.76403	6.45733	4.534279	
11	6.71289	6.16369	8.181275	
12	6.38989	5.9293	7.208105	
13	6.55119	5.74429	12.31685	
14	7.17213	6.54783	8.704527	
15	6.608	6.00081	9.188711	
16	5.85642	5.54769	5.271651	
17	6.21754	6.01224	3.301949	
18	6.25065	5.85904	6.265108	
19	6.79141	6.00447	11.58728	
20	6.57847	6.14999	6.513369	
21	6.43577	5.78132	10.16895	
22	7.01618	6.55545	6.566679	
23	6.75988	6.17117	8.708882	
24	7.9693	7.66984	3.75767	
25	7.21215	6.95524	3.562183	
26	6.81276	6.01228	11.74972	
27	6.44801	5.55942	13.78084	
28	6.09463	5.49304	9.870821	
29	6.52842	5.93849	9.036337	
30	6.46255	5.657	12.46489	

(c)

Tr.	Single Track	15 Segments	Improvement per track (%)	Averaged imp. (%)
1	7.12944	6.68951	6.170611	6.926994
2	6.54657	6.17711	5.643566	
3	7.15616	6.5784	8.073604	
4	7.13066	6.31104	11.49431	
5	6.32223	5.67158	10.29146	
6	6.50901	5.94409	8.679046	
7	6.55688	5.90061	10.00888	
8	6.74314	6.13154	9.069959	
9	7.06923	6.64821	5.95567	
10	6.76403	6.60665	2.326719	
11	6.71289	6.32404	5.792587	
12	6.38989	5.99115	6.24017	
13	6.55119	5.85325	10.65364	
14	7.17213	6.6762	6.914682	
15	6.608	6.21627	5.928117	
16	5.85642	5.6888	2.862158	
17	6.21754	6.19612	0.344509	
18	6.25065	6.01222	3.814483	
19	6.79141	6.07418	10.56084	
20	6.57847	6.24634	5.048742	
21	6.43577	5.81827	9.594811	
22	7.01618	6.64125	5.343791	
23	6.75988	6.33346	6.3081	
24	7.9693	7.66565	3.810247	
25	7.21215	6.96563	3.418121	
26	6.81276	6.2766	7.869938	
27	6.44801	5.76384	10.61056	
28	6.09463	5.65925	7.143666	
29	6.52842	5.97577	8.465295	
30	6.46255	5.85691	9.371533	

4.2.3 Babble Noise

Table 4.6: Denoising of Babble class noise. Comparison between no EWT decomposition and 5 segments decomposition, based on averaged LPC CD coefficients. SNR 0 dB (a), SNR -5 dB (b), SNR -10 dB (c)

(a)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.47797	5.83786	6.081048	6.614969
2	5.84928	5.30932	4.940162	
3	6.95446	6.50204	6.633846	
4	6.28094	5.95653	7.187378	
5	5.59636	5.16818	8.43252	
6	5.9134	5.51596	4.741929	
7	6.00483	5.51142	8.770465	
8	5.68724	5.14775	3.657524	
9	6.49064	5.77762	1.963261	
10	6.18694	5.54205	4.798112	
11	5.53535	5.34895	7.814274	
12	5.32845	5.19925	7.993297	
13	5.41637	5.21062	7.763535	
14	5.97652	5.52056	9.064178	
15	5.64187	5.36703	7.387503	
16	4.95698	5.04072	7.492964	
17	5.37805	4.93195	7.607009	
18	5.3923	5.0729	6.394899	
19	5.92638	5.94159	9.166914	
20	5.638	5.5697	7.023052	
21	5.22278	4.84345	6.930587	
22	6.21792	5.70809	4.30315	
23	6.07987	5.68412	6.62722	
24	7.17554	6.70651	3.134024	
25	6.31402	5.58476	4.874983	
26	5.5935	5.25154	6.275646	
27	5.41028	5.30484	8.212165	
28	5.06198	4.92387	4.832783	
29	5.46788	5.20432	6.620984	
30	5.2997	5.06742	11.72366	

(b)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	7.42084	6.43815	13.2423	15.77578
2	6.71413	5.65763	15.73547	
3	7.78029	6.63226	14.75562	
4	6.94412	5.89191	15.15253	
5	6.55756	5.53487	15.59559	
6	6.88955	5.78616	16.01541	
7	6.81802	5.86899	13.91944	
8	6.55529	5.48377	16.34588	
9	7.00259	5.87746	16.06734	
10	6.84434	5.68863	16.88563	
11	6.82453	5.70955	16.33783	
12	6.45352	5.38167	16.60877	
13	6.875	5.60435	18.48218	
14	7.34871	6.16599	16.09425	
15	6.91747	5.7605	16.72533	
16	6.23768	5.204	16.57155	
17	6.73715	5.52992	17.919	
18	6.29327	5.49019	12.76093	
19	6.80258	5.77446	15.11368	
20	6.61591	5.9015	10.79836	
21	6.71476	5.40203	19.54992	
22	7.24857	6.07348	16.21134	
23	6.91549	5.7153	17.3551	
24	8.13183	7.10651	12.60872	
25	7.18975	6.03877	16.00862	
26	6.62545	5.53157	16.51027	
27	6.72061	5.62762	16.26326	
28	6.18542	5.19425	16.0243	
29	6.61985	5.58516	15.63011	
30	6.66017	5.59557	15.98458	

(c)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	7.64723	6.81138	10.9301	14.47673
2	7.00325	5.99649	14.37561	
3	7.81848	6.85293	12.34959	
4	7.08136	6.22421	12.10431	
5	6.9696	5.87937	15.64265	
6	6.94868	5.95919	14.23997	
7	6.93126	6.08383	12.2262	
8	6.7805	5.81288	14.27063	
9	7.20635	6.27994	12.85547	
10	7.04496	6.07406	13.78148	
11	6.95142	5.89063	15.26005	
12	6.69981	5.59644	16.46868	
13	6.98466	5.72424	18.04555	
14	7.4199	6.42473	13.41218	
15	7.12861	6.19506	13.09582	
16	6.46533	5.33067	17.54992	
17	7.16757	5.84301	18.4799	
18	6.44414	5.59694	13.14683	
19	6.94651	5.89138	15.18935	
20	6.94298	6.11278	11.9574	
21	6.76644	5.55343	17.92686	
22	7.45777	6.50444	12.78304	
23	7.11018	6.06454	14.70624	
24	8.20965	7.47607	8.935582	
25	7.31884	6.32126	13.6303	
26	6.92294	5.9374	14.23586	
27	7.03724	5.85481	16.80247	
28	6.51737	5.42015	16.83532	
29	6.81662	5.64149	17.23919	
30	6.90267	5.8103	15.82533	

Table 4.7: Denoising of Babble class noise SNR -10 dB. Comparison between no EWT decomposition and 5, 10, and 15 segments decomposition, based on averaged LPC CD coefficients. 5 segments (a), 10 segments (b), 15 segments (c)

(a)

Tr.	Single Track	5 Segments	Improvement per track (%)	Averaged imp. (%)
1	6.47797	5.83786	6.081048	6.614969
2	5.84928	5.30932	4.940162	
3	6.95446	6.50204	6.633846	
4	6.28094	5.95653	7.187378	
5	5.59636	5.16818	8.43252	
6	5.9134	5.51596	4.741929	
7	6.00483	5.51142	8.770465	
8	5.68724	5.14775	3.657524	
9	6.49064	5.77762	1.963261	
10	6.18694	5.54205	4.798112	
11	5.53535	5.34895	7.814274	
12	5.32845	5.19925	7.993297	
13	5.41637	5.21062	7.763535	
14	5.97652	5.52056	9.064178	
15	5.64187	5.36703	7.387503	
16	4.95698	5.04072	7.492964	
17	5.37805	4.93195	7.607009	
18	5.3923	5.0729	6.394899	
19	5.92638	5.94159	9.166914	
20	5.638	5.5697	7.023052	
21	5.22278	4.84345	6.930587	
22	6.21792	5.70809	4.30315	
23	6.07987	5.68412	6.62722	
24	7.17554	6.70651	3.134024	
25	6.31402	5.58476	4.874983	
26	5.5935	5.25154	6.275646	
27	5.41028	5.30484	8.212165	
28	5.06198	4.92387	4.832783	
29	5.46788	5.20432	6.620984	
30	5.2997	5.06742	11.72366	

(b)

Tr.	Single Track	10 Segments	Improvement per track (%)	Averaged imp. (%)
1	7.64723	6.89355	9.855595	12.94053
2	7.00325	6.02255	14.0035	
3	7.81848	6.92817	11.38725	
4	7.08136	6.31965	10.75655	
5	6.9696	6.03414	13.422	
6	6.94868	6.10184	12.18706	
7	6.93126	6.09924	12.00388	
8	6.7805	5.89023	13.12986	
9	7.20635	6.32604	12.21575	
10	7.04496	6.1448	12.77736	
11	6.95142	6.17147	11.22001	
12	6.69981	5.65361	15.61537	
13	6.98466	5.90737	15.42366	
14	7.4199	6.50483	12.33265	
15	7.12861	6.2782	11.92953	
16	6.46533	5.45264	15.66339	
17	7.16757	5.98168	16.54522	
18	6.44414	5.68978	11.70614	
19	6.94651	6.0189	13.35361	
20	6.94298	6.19669	10.74884	
21	6.76644	5.70887	15.62964	
22	7.45777	6.64214	10.93665	
23	7.11018	6.22149	12.49884	
24	8.20965	7.54825	8.056373	
25	7.31884	6.51162	11.02934	
26	6.92294	5.93325	14.2958	
27	7.03724	6.03881	14.18781	
28	6.51737	5.49164	15.7384	
29	6.81662	5.81535	14.68866	
30	6.90267	5.87575	14.87714	

(c)

Tr.	Single Track	15 Segments	Improvement per track (%)	Averaged imp. (%)
1	7.64723	7.08909	7.29859	10.25242
2	7.00325	6.22433	11.12226	
3	7.81848	7.1199	8.934985	
4	7.08136	6.36571	10.10611	
5	6.9696	6.15961	11.62176	
6	6.94868	6.2049	10.7039	
7	6.93126	6.35294	8.343649	
8	6.7805	6.02471	11.14652	
9	7.20635	6.40006	11.1886	
10	7.04496	6.34306	9.963151	
11	6.95142	6.33999	8.795757	
12	6.69981	5.85307	12.63827	
13	6.98466	6.16446	11.74288	
14	7.4199	6.74937	9.036914	
15	7.12861	6.49775	8.849692	
16	6.46533	5.63131	12.89988	
17	7.16757	6.19501	13.56889	
18	6.44414	5.79903	10.0108	
19	6.94651	6.18668	10.9383	
20	6.94298	6.39936	7.829779	
21	6.76644	5.93542	12.2815	
22	7.45777	6.85306	8.108456	
23	7.11018	6.4966	8.629599	
24	8.20965	7.80634	4.912633	
25	7.31884	6.70717	8.357472	
26	6.92294	6.14129	11.29072	
27	7.03724	6.24848	11.20837	
28	6.51737	5.68167	12.82266	
29	6.81662	6.01046	11.82639	
30	6.90267	6.11618	11.394	

Chapter 5

Discussions

5.1 Motivation

The idea of using autonomous drones in search and rescue operations motivates the decision to work with noise classification and denoising. A flying drone capable of picking up human speech in low-visibility areas such as above forests, fogs, or at night. The first practical challenge would be to eliminate drone noise from recorded signals. That is what I have been working on in preparation for my master's thesis. For a long time, there has been a solution to this problem. Every mobile phone has two microphones, one for picking up ambient noise and the other for picking up contaminated speech. The algorithm for removing noise is proprietary and not disclosed to us, but spectral subtraction is a simple solution that works well.

Wavelet Empirical The transform decomposes the signal and provides time-frequency analysis in terms of subband signals. The idea of splitting a noisy signal into segments and then simply removing the noisy ones was too difficult to swallow. For segment removal, the naive approach was to use the iterative method. While it did work in some ways, each signal removed segment had to be individually selected. Machine learning evolved into a general solution that could handle any signal. Denoising research naturally led to the classification of noise patterns.

5.2 Process

Because I have previously worked with EWT in Matlab, it was decided that I would continue to use Matlab. Because of my lack of experience with CNN, the first task I wanted to complete was noise classification. This is a well-defined machine learning task, and almost all teaching resources demonstrate CNN using image classification examples. With the help of EWT decomposition, audio signals resemble images. This worked extremely well and aided me in learning CNN by example.

The second task was to solve the audio signal denoising. First, I attempted to solve the problem by decomposing signals, removing segments, and then recomposing segments into the modified signals. This really didn't work out, and my boss insisted that I look for a machine learning-based solution. Because Matlab already has such solutions and examples. It was decided to use it as a starting point and improve on it.

5.3 Implementation

GPU was used for classification network training and testing. This greatly accelerated the entire process and enabled the use of higher resolution "images." The number of layers and filter size were determined empirically. Good results have already been obtained after 40 epochs. This equates to approximately 6 minutes for 50 segments. However, as the number of segments doubles, so does the time required to train the network.

GPU is also used for denoising network training. I thoroughly tested various network modifications, but the Matlab solution produced the best results in terms of performance vs hardware requirements. Because several networks are trained one after the other, each network is saved on the hard drive and then erased to make room for RAM. Each network is trained over a period of two epochs.

5.4 Limitations

The classification network has a hardware limitation. Signal decomposition necessitates a large amount of RAM and VRAM, but a larger number of segments improves accuracy. With 16 GB of RAM and 8 GB of VRAM, I was able to reliably classify 100 segments. At 150 segments, I'm already at risk of running out of memory. Because it was deemed untrustworthy, the results of 150 segments are not included in this report.

The main limitation of denoising networks is their high latency; it takes an average of 1.3 seconds for the K-step Ahead prediction algorithm to run its course. And that's without accounting for data preparation (decomposition and STFT), which takes about 4 seconds. It could take up to 13 seconds to denoise 5 segments (considering OS lag). The total time for ten segments is 22 seconds.

Chapter 6

Conclusions

Convolutional Neural Networks for noise classification and denoising of audio signals have been presented in this thesis. Three types of noises have been used for testing networks, stationary, non-stationary, and babble type noise. The total number of noise classes is 14 with a signal-to-noise ratio ranging from 10 dB to -10 dB. The noise classification network is able to classify all 14 classes of noise with 100% accuracy at -5 dB SNR. Denoising network showing up to 18% improvement compared to denoising solution by STFT method. To achieve these results, audio signals are decomposed by Empirical Wavelet Transform before being processed by CNN in both cases. Although signals decomposition improves the quality of denoising, this is not a real-time solution due to high latency.

6.1 Further Work

In real-world application scenarios, use noise classification and denoising. A search and rescue flying drone, for example, could capture and transmit surrounding sound to a server via cellular network, Lora, Wimax, or Bluetooth. The server could be a cloud solution like Google Colab or Microsoft Azure, or it could simply be a personal computer. To make that work, the code must be rewritten in Python.

Increase the amount of training data, and include voices with various accents and backgrounds. Increase the number of noise classes even more. Perform a more in-depth level of segmentation on a more powerful PC to see if and when saturation occurs. Currently, increasing the number of segments from 50 to 100 improves classification accuracy slightly.

Bibliography

[1] J. Singh and R. Joshi, "Background sound classification in speech audio segments," in *Proc. International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, Timisoara, Romania, Oct. 2019, pp. 1–6.

[2] T. T. Nguyen, A. Fuchs, and F. Pernkopf, "Acoustic scene classification using deep mixtures of pre-trained convolutional neural networks," *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 871–875, 2019.

[3] M. Okaba and T. Tuncer, "An automated location detection method in multi-storey buildings using environmental sound classification based on a new center symmetric nonlinear pattern: Cs-lblock-pat," *Automation in Construction*, vol. 125, p. 103645, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580521000960

[4] I. Martín-Morató, M. Cobos, and F. J. Ferri, "Adaptive mid-term representations for robust audio event classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 12, pp. 2381–2392, 2018.

[5] H. Seker and O. Inik, "Cnnsound: Convolutional neural networks for the classification of environmental sounds," in *2020 The 4th International Conference on Advances in Artificial Intelligence*, ser. ICAAI 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 79–84. [Online]. Available: https://doi.org/10.1145/3441417.3441431

[6] S. Li, F. Li, S. Tang, and F. Luo, "Heart sounds classification based on feature fusion using lightweight neural networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–9, 2021.

[7] L. Huang and C.-M. Pun, "Audio replay spoof attack detection by joint segment-based linear filter bank feature extraction and attention-enhanced densenet-bilstm network," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1813–1825, 2020.

[8] W. Yang and S. Krishnan, "Combining temporal features by local binary pattern for acoustic scene classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1315–1321, 2017.

[9] H. Phan, O. Y. Chén, L. Pham, P. Koch, M. De Vos, I. McLoughlin, and A. Mertins, "Spatio-temporal attention pooling for audio scene classification," *arXiv preprint arXiv:1904.03543*, 2019.

[10] S. Chandrakala and S. L. Jayalakshmi, "Generative model driven representation learning in a hybrid framework for environmental audio scene and sound event recognition," *IEEE Transactions on Multimedia*, vol. 22, no. 1, pp. 3–14, 2020.

[11] S. Waldekar and G. Saha, "Wavelet transform based mel-scaled features for acoustic scene classification," in *Proc. Interspeech 2018*, 2018, pp. 3323–3327. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-2083

[12] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Leveraging deep neural networks with non-negative representations for improved environmental sound classification," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017, pp. 1–6.

[13] S. Ryu and S.-C. Kim, "Impact sound-based surface identification using smart audio sensors with deep neural networks," *IEEE Sensors Journal*, vol. 20, no. 18, pp. 10 936–10 944, 2020.

[14] T. Spadini, D. L. de Oliveira Silva, and R. Suyama, "Sound event recognition in a smart city surveillance context," *CoRR*, vol. abs/1910.12369, 2019. [Online]. Available: http://arxiv.org/abs/1910.12369

[15] A. Luo, E. Li, Y. Liu, X. Kang, and Z. J. Wang, "A capsule network based approach for detection of audio spoofing attacks," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6359–6363.

[16] R. Leonardo, M. Barandas, and H. Gamboa, "A framework for infrastructure-free indoor localization based on pervasive sound analysis," *IEEE Sensors Journal*, vol. 18, no. 10, pp. 4136–4144, 2018.

[17] V. Nivitha Varghees and K. I. Ramachandran, "Effective heart sound segmentation and murmur classification using empirical wavelet transform and instantaneous phase for electronic stethoscope," *IEEE Sensors Journal*, vol. 17, no. 12, pp. 3861–3872, 2017.

[18] K. A. Babu, B. Ramkumar, and M. S. Manikandan, "Automatic identification of s1 and s2 heart sounds using simultaneous pcg and ppg recordings," *IEEE Sensors Journal*, vol. 18, no. 22, pp. 9430–9440, 2018.

[19] Y. Qian, Z. Chen, and S. Wang, "Audio-visual deep neural network for robust person verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1079–1092, 2021.

[20] M. Mishra, H. Menon, and A. Mukherjee, "Characterization of s_1 and s_2 heart sounds using stacked autoencoder and convolutional neural network," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 9, pp. 3211–3220, 2019.

[21] A. Gomez-Alanis, A. M. Peinado, J. A. Gonzalez, and A. M. Gomez, "A gated recurrent convolutional neural network for robust spoofing detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 1985–1999, 2019.

[22] H. A. Sánchez-Hevia, D. Ayllón, R. Gil-Pita, and M. Rosa-Zurera, "Maximum likelihood decision fusion for weapon classification in wireless acoustic sensor networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1172–1182, 2017.

[23] D. Zhao, H. Zhang, S. Liu, Y. Wei, and S. Xiao, "Deep rational attention network with threshold strategy embedded for mechanical fault diagnosis," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–15, 2021.

[24] H. Dinkel, Y. Qian, and K. Yu, "Investigating raw wave deep neural networks for end-to-end speaker spoofing detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2002–2014, 2018.

[25] D. Salvati, C. Drioli, and G. Foresti, "End-to-end speaker identification in noisy and reverberant environments using raw waveform convolutional neural networks," in *INTERSPEECH*, 2019.

[26] S. Ahmad, S. Agrawal, S. Joshi, S. Taran, V. Bajaj, F. Demir, and A. Sengur, "Environmental sound classification using optimum allocation sampling based empirical mode decomposition," *Physica A: Statistical Mechanics and its Applications*, vol. 537, p. 122613, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437119314955

[27] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Boston, MA, USA, Sep. 2015, pp. 1–6.

[28] S. Abdoli, P. Cardinal, and A. Lameiras Koerich, “End-to-end environmental sound classification using a 1d convolutional neural network,” *Expert Systems with Applications*, vol. 136, pp. 252–263, Dec. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419304403

[29] J. Lu, R. Ma, G. Liu, and Z. Qin, “Deep convolutional neural network with transfer learning for environmental sound classification,” in *Proc. International Conference on Computer, Control and Robotics (ICCCR)*, Shanghai, China, Jan. 2021, pp. 242–245.

[30] Y. Chen, Q. Guo, X. Liang, J. Wang, and Y. Qian, “Environmental sound classification with dilated convolutions,” *Applied Acoustics*, vol. 148, pp. 123–132, May 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003682X18306121

[31] G. Hua and H. Zhang, “Enf signal enhancement in audio recordings,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1868–1878, Nov. 2020.

[32] D. Luo, P. Korus, and J. Huang, “Band energy difference for source attribution in audio forensics,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2179–2189, Mar. 2018.

[33] J. Salamon and J. P. Bello, “Deep convolutional neural networks and data augmentation for environmental sound classification,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, Jan. 2017.

[34] N. Davis and K. Suresh, “Environmental sound classification using deep convolutional neural networks and data augmentation,” in *Proc. IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, Thiruvananthapuram, India, Dec. 2018, pp. 41–45.

[35] X. Zhang, Y. Zou, and W. Wang, “Ld-cnn: A lightweight dilated convolutional neural network for environmental sound classification,” in *Proc. International Conference on Pattern Recognition (ICPR)*, Beijing, China, Aug. 2018, pp. 373–378.

[36] R. G. Lyons, *Understanding Digital Signal Processing*, 3rd ed. Pearson Education, Inc.

[37] E. Sejdić, I. Djurović, and J. Jiang, “Time–frequency feature representation using energy concentration: An overview of recent advances,” vol. 19, no. 1, pp. 153–183. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S105120040800002X

[38] N. Kehtarnavaz, *Digital Signal Processing System Design*, 2nd ed. Academic Press.

[39] V. Velardo, “AudioSignalProcessingForML,” original-date: 2020-06-18T11:54:37Z. [Online]. Available: https://github.com/musikalkemist/AudioSignalProcessingForML/blob/2b7efd37305a10d0059fdd0a41f011fd0218771a/6-%20How%20to%20extract%20audio%20features/How%20to%20extract%20audio%20features%20.pdf

[40] ——, “AudioSignalProcessingForML,” original-date: 2020-06-18T11:54:37Z. [Online]. Available: https://github.com/musikalkemist/AudioSignalProcessingForML/blob/2b7efd37305a10d0059fdd0a41f011fd0218771a/15%20-%20Short-Time%20Fourier%20Transform%20explained%20easily/Short-Time%20Fourier%20Transform%20Explained%20Easily.pdf

[41] Short-time fourier transform - MATLAB stft. [Online]. Available: https://www.mathworks.com/help/signal/ref/stft.html

[42] D. Gabor, “Theory of communication. part 1: The analysis of information,” vol. 93, no. 26, pp. 429–441, publisher: IET Digital Library. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/ji-3-2.1946.0074

[43] D. H. von Seggern, *CRC Standard Curves and Surfaces with Mathematica*, 3rd ed. CRC Press.

[44] J. K. Hammond and P. R. White, “THE ANALYSIS OF NON-STATIONARY SIGNALS USING TIME-FREQUENCY METHODS,” vol. 190, no. 3, pp. 419–447. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022460X96900723

[45] A. Graps, “An introduction to wavelets,” vol. 2, no. 2, pp. 50–61, conference Name: IEEE Computational Science and Engineering.

[46] N. M. Astaf’eva, “Wavelet analysis: basic theory and some applications,” vol. 39, no. 11, p. 1085, publisher: IOP Publishing. [Online]. Available: https://iopscience.iop.org/article/10.1070/PU1996v039n11ABEH000177/meta

[47] Choose a wavelet - MATLAB & simulink. [Online]. Available: https://www.mathworks.com/help/wavelet/gs/choose-a-wavelet.html

[48] M. Hanteh, O. Rezaifar, and M. Gholhaki, “Selecting the appropriate wavelet function in the damage detection of precast full panel building based on experimental results and wavelet analysis | request PDF.” [Online]. Available: https://www.researchgate.net/publication/352080994_Selecting_the_appropriate_wavelet_function_in_the_damage_detection_of_precast_full_panel_building_based_on_experimental_results_and_wavelet_analysis

[49] Ryan, *Linear Algebra, Signal Processing, and Wavelets - A Unified Approach*, matlab version ed., ser. Springer Undergraduate Texts in Mathematics and Technology. Springer. [Online]. Available: https://link.springer.com/book/10.1007/978-3-030-01812-2#siteedition-academic-link

[50] S. P. Nanavati and P. K. Panigrahi, “Wavelet transform,” vol. 9, no. 3, pp. 50–64. [Online]. Available: https://doi.org/10.1007/BF02834988

[51] C. Burrus, R. Gopinath, and H. Guo, “Introduction to wavelets and wavelet transform—a primer,” vol. 67. [Online]. Available: https://www.researchgate.net/publication/246532602_Introduction_to_Wavelets_and_Wavelet_Transform-A_Primer

[52] C. Valens, “A really friendly guide to wavelets.” [Online]. Available: http://agl.cs.unm.edu/~williams/cs530/arfgtw.pdf

[53] Continuous wavelet transform and scale-based analysis - MATLAB & simulink. [Online]. Available: https://www.mathworks.com/help/wavelet/gs/continuous-wavelet-transform-and-scale-based-analysis.html

[54] J. Gilles, “Empirical wavelet transform,” vol. 61, no. 16, pp. 3999–4010, conference Name: IEEE Transactions on Signal Processing.

[55] R. Panda, S. Jain, R. Tripathy, and U. R. Acharya, “Detection of shockable ventricular cardiac arrhythmias from ECG signals using FFREWT filter-bank and deep convolutional neural network,” vol. 124, p. 103939. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482520302742

[56] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6.

[57] S. Saha. A comprehensive guide to convolutional neural networks — the ELI5 way. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[58] Learn about convolutional neural networks - MATLAB & simulink. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/introduction-to-convolutional-neural-networks.html

[59] A. Bonner. The complete beginner’s guide to deep learning: Convolutional neural networks. [Online]. Available: https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb

[60] What are convolutional neural networks? [Online]. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks

[61] J. Feng, X. He, Q. Teng, C. Ren, H. Chen, and Y. Li, "Reconstruction of porous media from extremely limited information using conditional generative adversarial networks," vol. 100.

[62] CS231n convolutional neural networks for visual recognition. [Online]. Available: https://cs231n.github.io/convolutional-networks/

[63] NOIZEUS: Noisy speech corpus - univ. texas-dallas. [Online]. Available: https://ecs.utdallas.edu/loizou/speech/noizeus/

[64] aurora - aurora speech recognition experimental framework. [Online]. Available: http://aurora.hsnr.de/index-2.html

[65] Denoise speech using deep learning networks - MATLAB & simulink. [Online]. Available: https://www.mathworks.com/help/audio/ug/denoise-speech-using-deep-learning-networks.html#mw_rtc_DenoiseSpeechUsingDeepLearningNetworksExample_EE4C0B08

[66] J. Kim, "Speech enhancement toolkit," original-date: 2018-06-04T03:53:41Z. [Online]. Available: https://github.com/jtkim-kaist/Speech-enhancement/blob/84f1a3c1273fb4952522b911dd62cbb4476a534d/SE/lib/sub_lib/MATLAB_code/objective_measures/quality/comp_cep.m

[67] D. O'Shaughnessy, "Linear predictive coding," vol. 7, no. 1, pp. 29–32, conference Name: IEEE Potentials.

[68] R. B. Randall, "A history of cepstrum analysis and its application to mechanical problems," vol. 97, pp. 3–19. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327016305556

[69] N. Kitawaki, H. Nagabuchi, and K. Itoh, "Objective quality evaluation for low-bit-rate speech coding systems," vol. 6, no. 2, pp. 242–248, conference Name: IEEE Journal on Selected Areas in Communications.