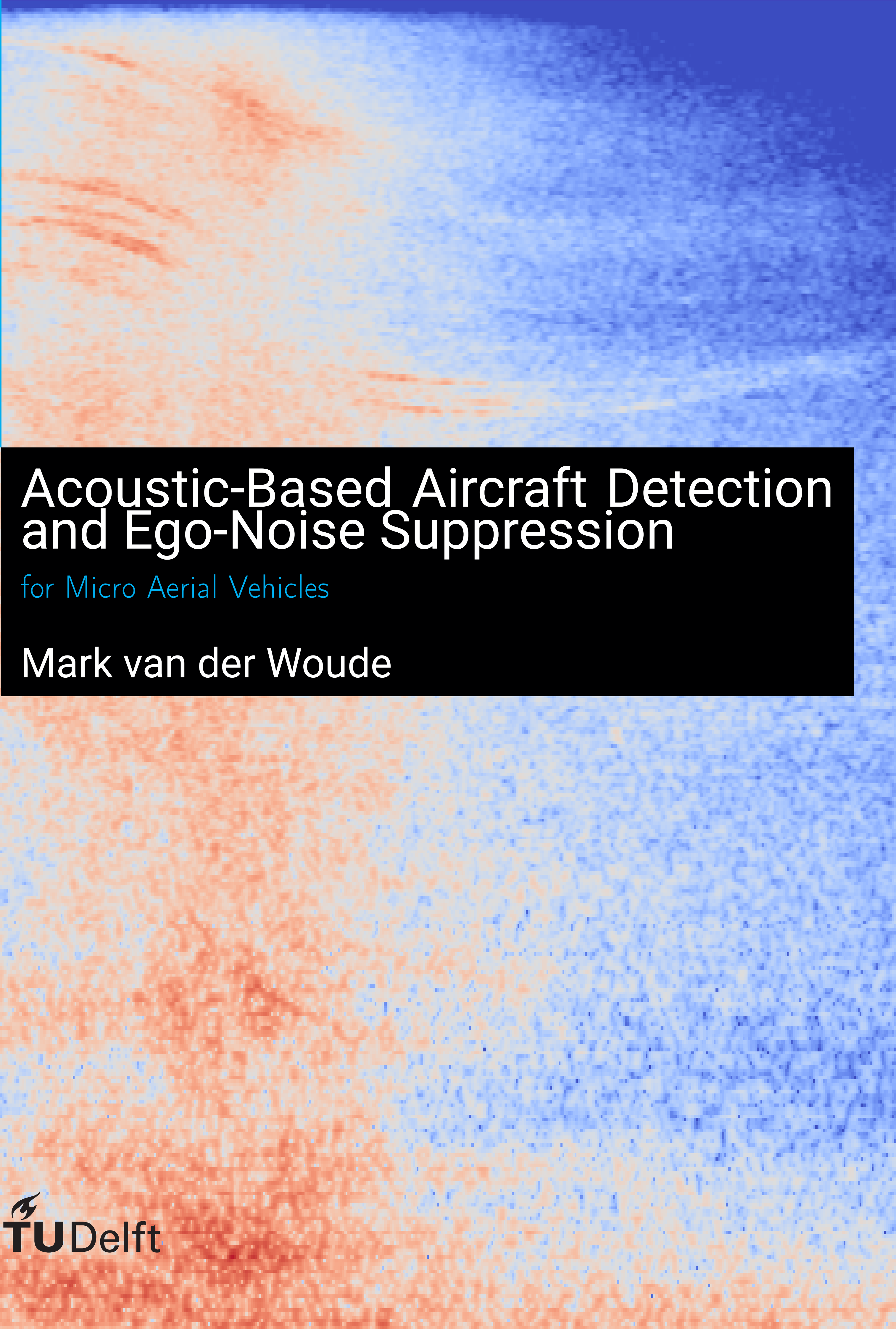

Faculty of Aerospace Engineering

Acoustic-Based Aircraft Detection and Ego-Noise Suppression

for Micro Aerial Vehicles

Mark van der Woude

TUDelft

Acoustic-Based Aircraft Detection and Ego-Noise Suppression

for Micro Aerial Vehicles

by

Mark van der Woude

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on March 4th, 2021.

Student number:	4100379	
Thesis committee:	Prof. Dr. G. C. H. E. de Croon,	TU Delft, supervisor
	Prof. Dr. Ir. M. Snellen,	TU Delft, external member
	Ir. C. De Wagter,	TU Delft, external member

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Contents

List of Figures

List of Tables

List of Abbreviations

ABF	Adaptive Beamformer
ADC	Analog-to-Digital Converter
ADS-B	Automatic Dependent Surveillance Broadcast
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
BF	Beamformer
B-LSTM	Bidirectional Long Short-Term Memory
BPTT	Back-propagation through Time
BSS	Blind Source Separation
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
CQT	Constant-Q Transform
DAA	Detect and Avoid
DAE	Denoising Autoencoder
DBN	Deep Belief Network
DCT	Direct Cosine Transform
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
DOA	Direction of Arrival
DSBF	Delay-and-Sum Beamformer
FFT	Fast Fourier Transform
FOV	Field of View
FN	False Negative
FP	False Positive
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HOG	Histogram of Oriented Gradients
HMM	Hidden Markov Model
ICA	Independent Component Analysis
I-DFT	Inverse Discrete Fourier Transform
LIDAR	Light Detection and Ranging
LSTM	Long Short-Term Memory

MAV	Micro Aerial Vehicle
MFC	Mel Frequency Cepstrum
MFCC	Mel Frequency Cepstral Coefficient
MLP	Multi-Layer Perceptron
NMF	Nonnegative Matrix Factorization
OCC	One-Class Classification
RBM	Restricted Boltzmann Machine
ReLu	Rectified Linear Unit
RMS	Root-Mean-Square
RNN	Recurrent Neural Network
ROC	Receiver Operation Characteristic
RPM	Revolutions per Minute
SAR	Synthetic Aperture Radar
SEC	Sound Event Classification
SED	Sound Event Detection
SGD	Stochastic Gradient Descent
SIF	Spectrogram Image Feature
SNR	Signal-to-Noise Ratio
STE	Short-Time Energy
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine
TCAS	Traffic Alert and Collision Avoidance System
T-F	Time-Frequency
TN	True Negative
TP	True Positive
UAV	Unmanned Aerial Vehicle
VLOS	Visual Line-of-Sight
ZCR	Zero-Crossing Rate

List of Symbols

Greek Symbols

ϵ	Learning rate of an optimizer
$\boldsymbol{\theta}$	Set of learnable parameters
θ_d	Direction of arrival
μ_k	mean of the k^{th} Gaussian of a GMM
ϕ_k	mixture coefficient of the k^{th} Gaussian of a GMM
ρ	Activation function of a layer in an ANN
$\boldsymbol{\Sigma_k}$	covariance matrix of the k^{th} Gaussian of a GMM
τ	Transmission delay

Roman Symbols

$\boldsymbol{b}$	Bias vector of a classifier
B	Cut-off Frequency
c	Speed of sound
$\boldsymbol{c}$	Hidden-to-output bias vector of a recurrent layer
$c[t]$	New cell state of an LSTM cell at time step t
$\tilde{c}[t]$	Proposed new cell state of an LSTM cell at time step t.
$E[m]$	Short-time energy
$E[k,m]$	Band energy
f	Frequency
f'	Observed frequency
f_c	Cylinder firing frequency of a piston engine
f_e	Exhaust frequency of a piston engine
f_{mel}	Mel-scale frequency
f_s	Sample frequency
f_1	Blade passage frequency
$F[m]$	Spectral flux
$f[t]$	Forget gate of an LSTM cell at time step t
$f(\boldsymbol{x},\boldsymbol{y}\|\boldsymbol{\theta})$	Surrogate function
$g(\boldsymbol{\theta})$	Regularization term
$g[t]$	Additional hidden units of a bi-directional recurrent layer at time step t

$h[t]$	Hidden units of a recurrent layer at time step t
$h(\boldsymbol{x}\|\boldsymbol{\theta})$	Predictor for input $\boldsymbol{x}$, given parameters $\boldsymbol{\theta}$
$i[t]$	Input gate of an LSTM cell at time step t
$L(h(\boldsymbol{x}\|\boldsymbol{\theta}),\boldsymbol{y})$	Loss function
$o[t]$	Output gate of an LSTM cell at time step t
p	Number of rotors on an MAV
$P_{\boldsymbol{\theta}}$	Probability density w.r.t. $\boldsymbol{\theta}$
$P(c_j)$	Prior probability of c_j
$p\left(\boldsymbol{x},\boldsymbol{y}\right)$	Joint probability distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$
$p\left(x_i\|c_j\right)$	Conditional probability of x_i, given c_j
Q	Quality factor
r	Mix ratio
$\boldsymbol{R}[k,m]$	Correlation matrix of a time-frequency signal
$s[k,m]$	Discrete time-frequency signal of the target sound
$s[n]$	Discrete time-domain signal of the target sound
t	Time
$\boldsymbol{U}$	Input-to-hidden weight matrix of a recurrent layer
$\boldsymbol{V}$	Hidden-to-output weight matrix of a recurrent layer
$v[n]$	Discrete time-domain signal of the ego-noise
$\boldsymbol{W}$	Weight matrix of a classifier
$w[k,m]$	Spatial filter
$w[k,n]$	Windowing function (CQT)
$w[n]$	Windowing function (STFT)
$\boldsymbol{x}$	Set of inputs to classifier
$X[k]$	Discrete frequency-domain signal
$X[k,m]$	Discrete time-frequency signal
$x[n]$	Discrete time-domain signal
$x(t)$	Continuous time-domain signal
$x[t]$	Input units of a recurrent layer at time step t
$X_c[n]$	Cepstrum of $X[k]$
$\boldsymbol{y}$	Labels corresponding to $\boldsymbol{x}$
z	Layer in an ANN
$z[t]$	Hidden state of an HMM at time step t

Superscripts

H	Hermitian Transpose
k	k^{th} layer in an ANN
T	Transpose

Chapter 1

Introduction

The widespread usage of Micro Air Vehicles (MAVs) has led to various airspace safety breaches, such as unintentional near mid-air collisions with other aircraft [6]. Because these MAVs are typically too small to be detected by manned aircraft, they must be equipped with autonomous Detect and Avoid (DAA) systems when operating outside the visual line-of-sight (VLOS) of the operator, to ensure safe integration into general aviation[1]. The purpose of a DAA system, consisting of a detection and an avoidance subsystem, is thus to provide surveillance, alerting and ultimately guidance to the UAV [7]. A separation distance of 500ft is required to prevent (near) mid-air collisions [8].

A DAA system can be categorized as cooperative or non-cooperative [9, 10, 11]. A cooperative detection method requires the invading aircraft to be equipped with a system compatible with the DAA system on the MAV, while a non-cooperative method does not have this requirement. A non-cooperative detection method is considered active when the DAA emits a signal, and passive when it does not.

Examples of cooperative detection methods are the Traffic Alert and Collision Avoidance System (TCAS) and Automatic Dependent Surveillance Broadcast (ADS-B). While both transponder-based methods are reliable, they are not feasible as a detection method for MAVs. TCAS is too heavy, too large and has a too high power consumption to be equipped on any MAV. On the other hand, ADS-B transceivers such as the *pingRx*[2] are light-weight with low power consumption, but light-weight aircraft are not (required to be) equipped with the system.

Active non-cooperative detection methods include radar and laser systems. Several small-scale radar systems have been developed to be equipped on an MAV[12, 13]. In [12], aircraft are differentiated by their Doppler signature, which depends on engine dimensions of the target aircraft. Other small-scale systems use a Synthetic Aperture Radar (SAR), but mainly for terrain mapping and not for aircraft detection [13]. Detection based on laser methods such as LIDAR (Light Detection and Ranging) suffer from a narrow field-of-view [10].

Passive non-cooperative detection include optical and acoustic systems. An Optical system must be combined with a thermal (infrared) camera to provide coverage in non-clear weather conditions. Detection methods for optical system include sky segmentation[14, 15], motion-based approaches[16] or classification using deep neural networks [17].

Acoustic systems require a single microphone for aircraft detection, or a microphone array for localization. An advantage of a such a system is that a single microphone can capture sound from any direction, whereas an optical system would require an array of cameras to obtain a full Field of View (FOV). While an acoustic vector sensor, mounted on an MAV, is used in [18] to localize nearby sounds, general research regarding the feasibility of aircraft detection based on their acoustic signature is limited.

[1] https://www.easa.europa.eu/sites/default/files/dfu/Introduction%20of%20a%20regulatory%20framework%20for%20the%20operation%20of%20unmanned%20aircraft.pdf

[2] https://uavionix.com/products/pingrx/

1.1 Motivation and Research Question

Therefore, the objective of this thesis is to design an autonomous system which enables an in-flight MAV to register the presence or absence of aircraft in real-time by analyzing the sound in its environment. While research specifically targeting aircraft sound is limited, more general topics such as Sound Event Detection (SED) or -Classification (SEC) have been researched for decades. Research into robust SEC is especially of interest, as the introduction of random noise to the dataset more accurately reflects the noise-corrupted sound recorded by the MAV. In the preliminary work of this thesis, the methodology and performance of recent sound classification methods will be analyzed, such that a selection can be implemented during the thesis.

The main research question is formulated as follows:

How can an in-flight Micro Aerial Vehicle, equipped with an acoustic sensor, detect the presence of nearby aircraft?

1.2 Structure

The report is divided into three main parts, structured as follows. The main contributions of the thesis are presented in the scientific paper in Part I. The paper, which can be read as a standalone document, is divided into five sections. The first section gives a brief overview of current aircraft detection methods and their applicability to MAVs. Section II of the paper summarizes work related to sound classification and ego-noise reduction, applicable to aircraft detection and/or MAVs where possible. Section III outlines the data acquisition and feature extraction/pre-processing of the acquired data. Section IV describes the experiments conducted and their outcome. Finally, section V lists the conclusions of the research and suggestions regarding future aircraft detection research.

Part II, consisting of chapters 2 to 5, concerns an extensive literature review. Chapter 2 provides a brief summary of aircraft sound characteristics, as well as an overview of common sound representations suitable for classification and detection. In Chapter 3, sound classifiers are introduced and analyzed. Chapter 4 reviews approaches for ego-noise removal. Finally, Chapter 5 provides a synthesis regarding the literature survey.

In Part III, consisting of chapters 6 to 8, the preliminary analysis is conducted. Chapter 6 investigates the performance of various established feature representations in combination with Convolutional Neural Network architectures based on prior research. Chapter 7 assesses whether ego-noise prediction based on MAV flight data is a viable approach. Finally, Chapter 8 summarizes and discusses the findings of both analyses.

I

Scientific Paper

Acoustic-Based Aircraft Detection and Ego-Noise Suppression for Micro Aerial Vehicles

C.A.M. van der Woude*[‡], J.C. van Dijk[†‡], G.C.H.E. de Croon[†‡]
*MSc.student, [†]Supervisor,
[‡] Control and Simulation, Department of Control and Operations
Delft University of Technology, Delft, The Netherlands

***Abstract* — Widespread usage of Micro Aerial Vehicles (MAVs) has led to various airspace safety breaches, including near mid-air collisions with other aircraft. To ensure safe integration into general aviation, it is paramount that MAVs are equipped with an autonomous detect and avoid system when flying beyond the visual line-of-sight of the operator. The purpose of this research is to investigate the feasibility of acoustic-based aircraft detection, which has generally been overlooked in favor of optical or radar-based technology. Effective sound-based aircraft detection on-board an MAV requires suppressing the dynamic ego-noise it generates during flight, which would otherwise pollute the recorded environmental sound. This paper proposes using a recurrent neural network to predict the generated noise, given a sequence of MAV flight data, so that it can be effectively removed from noisy recordings. For aircraft detection, a convolutional neural network in combination with Mel spectrogram features is designed to classify noise-free environmental sound as either aircraft or non-aircraft, achieving 97.5% accuracy. To reconstruct the noisy environment of an MAV flight, these noise-free sounds are mixed with ego-noise at mix ratios up to 1.00. When evaluating in these mismatched conditions, accuracy decreases to 95.0% and 47.5% with- and without ego-noise suppression, respectively. Although ego-noise suppression can not prevent a drop in performance, the large difference between the mismatched conditions does demonstrate the benefits of the proposed denoising approach on aircraft detection.**

***Index Terms* — Aircraft Detection, Sound Event Classification, Ego-Noise Suppression, Mel Spectrogram, Artificial Neural Networks**

I Introduction

Widespread usage of Micro Aerial Vehicles (MAVs) has led to various airspace safety breaches, such as near mid-air collisions with other aircraft [1]. As these MAVs are too small to be detected by manned aircraft, they must be equipped with autonomous Detect and Avoid (DAA) systems when operating outside the visual line-of-sight of the operator, to ensure safe integration into general aviation. The purpose of the DAA system is then to provide surveillance, alerting and ultimately guidance to the MAV [2]. This section aims to summarize current and potential aircraft detection systems and their applicability to MAVs. The design of an avoidance system is a different matter and beyond the scope of this research; the proposed maneuver when a threat is detected is simply to land immediately.

The detection system can be categorized as either cooperative or non-cooperative. As the name implies, a cooperative module requires both the invading aircraft and the MAV to be equipped with a compatible detection method. Examples of robust cooperative detection modules in modern aviation are the Traffic alert and Collision Avoidance System (TCAS) and Automatic Dependent Surveillance-Broadcast (ADS-B). Naturally, systems such as TCAS are too heavy, large, and power-consuming to be equipped on any MAV. On the other hand, light-weight low-power ADS-B transceivers are commercially available. However, lighter aircraft such as emergency helicopters and general aviation, are not required to be equipped with an ADS-B system, and as such compliance with ADS-B detection can not be guaranteed.

Non-cooperative methods do not required the invading aircraft to be equipped with it, and can be further divided into active and passive modules. Active modules, such as laser or radar systems, transmit a signal to the environment. Although light-weight radar systems appropriate for MAVs exists, and have been used successfully for terrain imaging [3], the technology is still in an early stage. Passive modules include optical and acoustic sensors. Optical systems are often combined with thermal sensors to provide coverage in non-clear weather conditions. Detection methods using optical sensors include sky segmentation[4, 5], motion-based approaches[6] and classification with deep neural networks[7].

Compared to optical systems, research regarding acoustic detection systems is still in an early stage. In [8],a feasibility study was conducted regarding sound localization from an MAV with an acoustic vector sensor. Since then, various approaches to detect intruding MAV using sound have been proposed, most of which involve the design of a spatial filter using a microphone array.

The objective of this research is to investigate the feasibility of acoustic-based aircraft detection on an in-flight MAV, using only a single microphone. As the microphone is mounted close to the body of the vehicle, non-stationary noise generated by the moving MAV (ego-noise) will be present in the recordings. The approach taken during this research is to employ an Artificial Neural Network (ANN) to *learn* the relation between data measured from the MAV (e.g. motor speed, velocity) and ego-noise, such that the latter can be effectively filtered out. The noise-free remainder of the sound can then be used for aircraft detection, by classifying the incoming sound as either aircraft or non-aircraft.

Two scenarios have been considered in this research. In the first scenario, a separate ANN is designed for the ego-noise prediction stage and the aircraft classification stage. The denoised input for the aircraft classifier is then obtained by subtracting the predicted ego-noise from the noise-corrupted signal. In the second scenario, the ego-noise removal and aircraft classification is performed simultaneously in a single ANN, which is given both the noise-corrupted audio and MAV data.

The structure of this article is as follows. Section II discusses research performed in the audio classification- and ego-noise removal domains, respectively. Section III elaborates on the acquisition and processing of the data required for the two test scenarios. Section IV explains the model selection process and describes the results for both test scenarios. Finally, the conclusions of the research are outlined in Section V,

II Related Work

A Audio Classification with Convolutional Neural Networks

Research into audio processing tasks (e.g. automatic speech recognition, sound event detection) has been conducted for decades. Initially, established classifiers like Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) were used in combination with manually engineered features such as the Mel-Frequency Cepstral Coefficients (MFCCs). In recent years, a shift has taken place in which feature learning using ANNs has become favored over these traditional classifiers [9].

Convolutional Neural Networks (CNNs) in particular achieve state-of-the-art performance in visual recognition tasks by taking advantage of the data locality in images [10]. This also translates to the audio processing domain, where the best performing input feature is usually an image representation, i.e. a time-frequency spectrogram [11]. The most frequently used variation is the log-power Mel-frequency spectrogram (referred to as Mel spectrogram for brevity). Although the Mel scale has historically been used for speech processing tasks, as it is based on human perception of frequencies, the higher frequency resolution at low frequencies has proven beneficial for general audio processing tasks as well [9]. Variations of the Mel spectrogram are used regularly, such as a frequency smoothed spectrum[11] or augmentation with the time-derivative of the spectrum [10, 12].

Network architectures have remained relatively shallow, with 2-3 convolutional followed by 2-3 fully-connected (FC) layers [10, 11, 12, 13]. In [10], a tall kernel is used in the first convolutional layer to learn filters spanning almost the entire frequency band. Kernels with a small receptive field are used in [13] to learn small, localized patterns which can be fused in subsequent layers to detect larger time-frequency signatures. In [12] it is suggested that by making use of dilated kernels, the extracted features are more separable than those extracted from traditional kernels, which implies a better capability for providing discriminative high-level features.

The majority of the research regarding audio classification using ANNs focuses on multi-class classification, on datasets with 10-50 classes such as UrbanSound8k [14] or ESC-50[15]. Binary classification tasks are usually limited to speech- or music onset detection. Research regarding aircraft/non-aircraft classification specifically is especially limited. In [16], a GMM classifier using MFCC features is used to detect airplane take-off sound events in noisy environments, with 94% accuracy. To the authors best knowledge, the only study regarding aircraft detection using MAVs is conducted in [17]. In [17], a CNN in combination with Mel spectrogram features is used to detect aircraft presence in audio recorded from an airport runway. Although MAV noise is mixed onto the recordings to simulate its ego-noise, this is not accounted for during training. This work is intended as a continuation of [17], with the aim of predicting the ego-noise generated by an MAV such that it can be effectively filtered from noisy input data to improve classification.

B Ego-noise Suppression

The noise emitted by a multi-rotor UAV can be considered a mixture of narrow-band harmonic noise and broadband noise [18]. The harmonic component consists of the mechanical noise generated by the rotating motors, with the fundamental frequencies of these components proportional to the rotational speed of the rotor, while the broadband component is comprised of noise generated by the propeller blades cutting through the air [19]. They also deduce that the ego-noise of a quad-rotor UAV can be modeled as the sum of four directional point-sources and one directionless diffuse noise which results from superposition of the point-source and broadband noise. Although the noise spectra vary dynamically with the motor rotation speed, the mixing network itself can be assumed stationary even on a moving MAV, as long as the relative positions between microphone(s) and motors is kept constant [19].

Ego-noise suppression methods can be divided into self-contained and dependent methods. Self-contained methods rely solely on information from an array of microphones, usually mounted on or around the MAV in a circular configuration. In [20], a delay-and-sum beamformer consisting of 16 microphones, mounted around a hovering MAV, identifies the location of a speaker based on the sound intensity in the captured environment. In [19], blind source separation is applied to denoise a speech signal contaminated with ego-noise of a moving quad-rotor, resulting in Signal-to-Noise Ratio(SNR) improvements up to 20dB for configurations with at least six microphones. In [21], ego-noise is reduced by exploiting the time-frequency sparsity of the energy peaks of speech and MAV noise in a mixed recording. After estimating the direction of arrival of the sound in each bin via a spatial likelihood function, the bins belonging to the target sound are extracted based on their proximity to the target direction. Although this method provides better noise attenuation than blind source separation, it does require the target direction to be known a priori. This method is extended in [22]: instead of providing the target DOA beforehand, it is estimated under the assumption that the output of a spatial filter which successfully extracts the target sound shows a higher non-Gaussianity at low SNR scenarios. In [23], this method is applied to a moving sound source by tracking the noisy estimations of the DOA with a particle filter.

Dependent approaches require additional sensory information, such as the rotational speed of the motors. Research regarding such approaches for denoising of multi-rotor MAVs is rather limited. An order analysis-based approach is suggested in [24] for a single-propeller fixed-wing UAV, where the noise spectrum is represented in a revolution-order domain instead of the time-frequency domain. Because only a single motor is present, the ego-noise generated occurs in narrow bands at orders of its fundamental frequency. The mixed signal is then denoised by subtracting all energy at the harmonic orders from the total energy within the spectrum. A drawback of this method is that it does not discriminate between target sound and unwanted noise; any target sound present at the orders will not be observable in the noise-free signal. This may especially be a problem for a moving multi-rotor MAV, where the bands are less narrow due to deviations between the rotational speed of each motor. In [25], the ego-noise spectrum of a ground-based robot is predicted via template-based online learning. Prior to the experiments, a template database is generated containing pairs of joint states and ego-noise spectra. For each new recording, the ego-noise spectrum is estimated by retrieving the noise template corresponding to the state vector in the database which is closest to the recorded states. If the difference between these state vectors exceeds a given threshold and no other sounds are present, the pair is added to the database instead.

This paper uses a dependent ego-noise filtering approach because the extra sensor data provided by the MAV is likely to improve ego-noise prediction. The template learning method in [25] is not too dissimilar to the method used in this paper; the key difference is that the neural network-based ego-noise prediction is not bounded by the finite size of a database.

III Methodology

This section contains the methodology of the report, which is divided into two parts. Section III-A outlines the data acquisition process for ego-noise prediction and aircraft classification, while Section III-B details the preprocessing and feature extraction of the acquired data.

A Data Acquisition

Ego-Noise Prediction

The data required for ego-noise prediction has been obtained by logging short flights of a Parrot Bebop2 drone within an 8-by-8 meter indoor area. Commands to the MAV (e.g. take-off, flight, landing) were given via Paparazzi UAV[3], which was running on a nearby Ubuntu 18.04 laptop. In flight, the MAV navigated autonomously through the arena by guiding itself towards waypoints, which were generated randomly and on-the-fly. The horizontal components of each waypoint were within 3 meters of the center of the area; the height of each waypoint was constrained between 0.5 and 3.5 meters.

[3]https://github.com/paparazzi/paparazzi

The data gathered during each flight contains a sequence of state vectors, measured at approximately 10 ms intervals. Each state vector contains seven subgroups, totaling 23 states: the rotational speed of each of the four rotors (RPM), the four stabilization commands (thrust, roll, pitch and yaw), the position (NED), velocity (NED), acceleration (NED), body attitude angles (pitch, roll, yaw) and body rotational rates (pitch rate, roll rate, yaw rate). With the exception of position and velocity, which were relayed to the drone via a motion capture system within the arena, all aforementioned states were measured on-board the MAV itself.

The in-flight MAV noise has been recorded at a sample rate of 44.1 kHz using a USB microphone, connected to a Raspberry Pi 3 (Pi), which itself was powered by the battery of the drone. The Pi was mounted on the bottom of the drone, allowing for battery swapping without needing to remove the device. This ensured that the microphone remained in a fixed position w.r.t the MAV, which is crucial for obtaining consistent noise measurements. A drawback of this configuration is the increase in noise in the recordings due to the downward airflow generated by the propellers.

The data collection process is done automatically. As soon as the motors are turned on using Paparazzi, the MAV starts transmitting state data via UDP to the ground station (laptop). In turn, the ground station signals the Pi to begin recording and transmitting audio. The time delay between the MAV and microphone recordings caused by this signal transmission is negligible ($\ll$ 1 ms). Likewise, data transmission ends when the motors are turned off after landing. 14 recordings were obtained in total, with an approximate duration of 50 seconds each. These were split into a training, validation and test set with a {0.6, 0.2, 0.2} split between the respective sets. With 14 recordings available, this resulted in a division of {8, 3, 3} for the dataset.

Aircraft Classification

The ESC-50 dataset [15] is used for training and evaluating the aircraft classifier, as it is one of the few datasets containing environmental recordings of helicopters. This dataset comprises 50 classes, each class itself containing 40 five-second recordings. ESC-50 was preferred over the much larger dataset AudioSet [26], since the labeling in the latter was less accurate.

For aircraft classification, the classes 'airplane' and 'helicopter' are labeled as 'aircraft' (1), while the classes 'engine', 'train' and 'wind' are labeled as 'non-aircraft' (0). The 'wind' class was chosen as a counter-example because the sound recorded by an outdoor MAV will likely be contaminated with wind, which should not result in false positives. Learning to classify wind as non-aircraft sound is especially important because all MAV recordings have taken place indoors, thus it will not be detected by the ego-noise suppressor. The 'engine' class was chosen since engine noise is often periodic in nature, as is the rotor blade noise found in the recordings belonging to the 'helicopter' class. The engines in this class belong predominantly to road vehicles; no aircraft engines appear within the class. Finally, the 'train' class was selected because the passing by of trains and aircraft both result in an observed change in frequency (Doppler shift) in the recording. The remaining classes have been discarded as they should rarely occur during an MAV flight. Silent fragments within recordings, most of which were present in the 'engine' and 'helicopter' classes, have been pruned as well.

The dataset was then split into a training, validation and test set with a ratio of {0.64, 0.16, 0.20} between sets and even distribution between categories ({25, 7, 8} recordings per category per set). The training set is augmented in order to increase the variety of available training data, without altering the semantics of each label. Three types of data augmentation were applied separately to each recording of the training set, increasing the size of the training set from 125 to 1,625 recordings:

- Intra-class mixing: mix each recording with a random recording of the same class (within the training set). The mix ratio is randomly chosen between 0.2 and 0.5, and a random offset is applied to the chosen recording. The mixing is done four times, with a different recording each time. The objective of intra-class mixing is to ensure that a mix of two similar events occurring within one recording, e.g. two airplane fly-overs, should still be identified appropriately.

- Time stretching: slow down or speed up the recording. Each recording was time stretched by the factors (0.70, 0.85, 1.15, 1.30), resulting in four additional recordings.

- Pitch shifting: lower or raise the pitch of the recording. Each recording was pitch shifted by (-2, -1, 1, 2) semitones, resulting in four additional recordings. In [13], a significant increase in classification accuracy was reported with pitch shift augmentation.

B Preprocessing and Feature Extraction

This section describes the steps necessary to make the raw data suitable for analysis in the two scenarios.

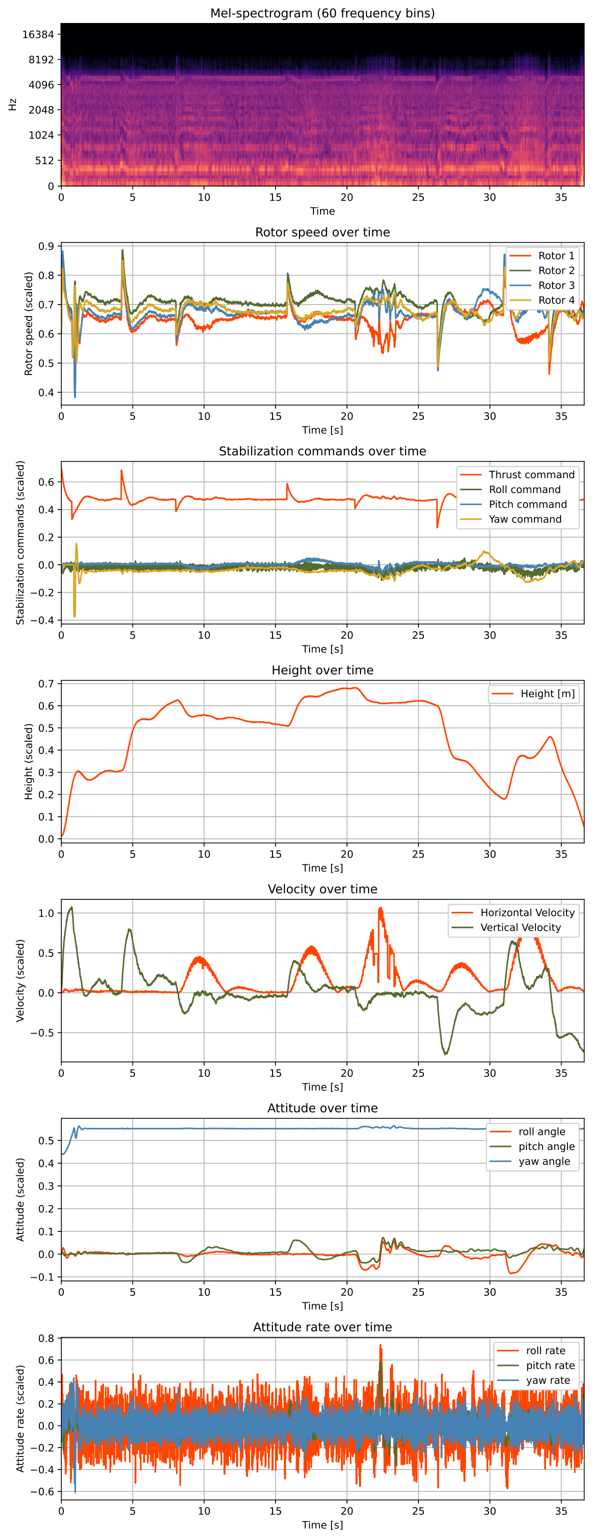

Figure 1: Example of the noise spectrum of an MAV flight (top), and several corresponding states.

Scenario 1: Ego-Noise Prediction

The data required for ego-noise prediction consists of state data as well as audio data. The preprocessing of the MAV states consists of three main adjustments to the gathered data. The first adjustment concerns the pruning of unnecessary states and the merging of redundant states. The x- and y-coordinate of the position vector are discarded, as they should have no influence on the ego-noise. The z-coordinate ('down') is kept, since ground effects may influence the noise footprint when the MAV is close to the ground. It is negated and referred to as height The x- and y-coordinate of velocity and acceleration are combined into horizontal velocity and acceleration, respectively. For consistency, downward velocity and acceleration are negated and referred to as vertical velocity and vertical acceleration. The second adjustment is the scaling of all states to an interval close to [0, 1] or [-0.5, 0.5] using a constant factor for each state. The third and final adjustment is the augmentation of the state vector with the time-derivative of the motor speed and stabilization commands. The aim of these two augmentations is to provide additional dynamics to the network. All aforementioned changes, including the scaling factors, are summarized in Table 1.

Measured [Unit]	Altered	Scaling
Motor speed [rpm]	-	12,000
-	Motor speed (delta)	-
Stabilization commands [-]	-	12,000
-	Stabilization commands (delta)	-
Position (z) [m]	Height	3.5
Velocity (x, y) [m/s]	Velocity (horizontal)	$\sqrt{8}$
Velocity (z) [m/s]	Velocity (vertical)	2
Acceleration (x, y,) [m/s^2]	Acceleration (horizontal)	$\sqrt{800}$
Acceleration (z) [m/s^2]	Acceleration (vertical)	20
Attitude angles (ϕ, θ, ψ) [rad]	-	π
Attitude rates (p, q, r) [rad/s]	-	π

Table 1: The measured state vector (left), alterations made to the components of the measured state vector (middle), and scaling applied to the altered state vector (right).

The predicted ego-noise must be subtracted from the noise-contaminated data obtained during regular flight, so that a denoised feature can be fed to the aircraft classifier. It is therefore logical to predict the noise in a format compatible with the features extracted from the ESC-50 dataset, which are ultimately used to train the classifier. The chosen time-frequency representation is a Mel spectrogram with 60 frequency bins, extracted using a window length of 1024 samples (23.2 ms) and a hop length of 512 (11.6 ms). The rationale behind choosing this representation is described in detail in the next section.

The state vectors and time-frequency spectra must now be synchronized, i.e. each state vector (input) must be coupled to one frame of the spectrogram (target). A new state vector is obtained every 10 ms, while a new spectrogram frame is generated approximately every 11.6 ms. Synchronization is done by matching each frame with the state vector closest in time, while the state vectors without a matching frame are discarded. An overview of the obtained ego-noise and corresponding scaled states, with the exception of acceleration, delta-rpm and delta-commands due to space limitations, is given in Figure 1

Two more adjustments are made to finalize the dataset. The first is to omit the take-off phase, identified by a spike in the rpm and delta-rpm early on in the recording. In Figure 1, this spike occurs around the 1-second mark. The second adjustment stems from the assumption that noise induced by the MAV does not vanish instantaneously; instead it gradually dissipates over subsequent frames. This context in time is added to the dataset by transforming each input-output pair into a sequence which includes the previous six pairs (i.e. the sequence length is 7). In other words, state vector 1-7 are used to predict the ego-noise at frame 7, state vector 2-8 predict the ego-noise at frame 8, etc. Although the first six pairs of each recording (roughly 70 ms) must be discarded because they contain an incomplete sequence, this is a negligibly small loss of data. The final dataset contains approximately {29k, 11k, 10k} samples for the training, validation and test set, respectively.

Scenario 1: Aircraft Classification

Prior to being fed to the classifier, the audio waveform is converted to a time-frequency representation via the Short-Term Fourier Transform (STFT). The Mel spectrogram is the representation of choice, as it outperformed both linear and constant-Q frequency scaling in a preliminary analysis conducted. Most noteworthy was the abysmal performance of the constant-Q spectra, despite having the highest frequency revolution at lower frequencies, where vehicle sounds are expected to dominate. The Mel scale, based on human perception of speech, is linear until 700 Hz and then scales logarithmically with frequency:

$$f_{mel} = 2595 \log_{10}\left(1 + \frac{f_{Hz}}{700}\right). \tag{1.1}$$

Extracting the feature from each recording is as follows:

1. The recording is (re-)sampled at 44.1 kHz and silent fragments at the start and/or end of the recording are trimmed. A preliminary analysis indicated that maintaining a high sample rate was beneficial for performance, despite aircraft sound dominating at frequencies below 12 kHz (i.e. a cut-off sample rate of 24 kHz).

2. The Mel spectrogram of each recording is extracted, where the STFT uses a window length of 1,024 (≈23.2 ms), hop size of 512 (≈11.6 ms), which are fairly standard values. 60 frequency bins were chosen for the Mel filterbank, as this led to the best performance under noise-free conditions in scenario 1. The obtained power spectrum is then converted to a logarithmic scale to magnify the differences found in the spectra, which is especially important for classification. This is step 1 in Figure 2

3. The Mel spectrum of the recording is then split up into segments of 60 frames each (approximately 0.7 seconds), with 50 % overlap between consecutive segments. As with the number of Mel filterbank bins, this segment length was chosen because it led to the best performance under clean conditions. By using an overlap between consecutive segments, the size of the resulting dataset is increased. The splitting of segments is centered w.r.t. the recording: a five-second recording consists of 431 frames, and is split using frame indices {6-65, 36-95, ..., 366-425}. This is step 2 in Figure 2.

4. The segment is smoothed in frequency via a Savitzky–Golay filter with a width of 3 and padding with first-order interpolation. After the initial experiments, this led to an additional 1.5% increase in accuracy. [11] report similar findings after frequency smoothing.

5. The spectrum is re-scaled to a [0, 1] interval, serving as the first input channel to the network.

6. The second input channel is the smoothed time-derivative of the first channel (delta for brevity). The delta is extracted via a first-order Savitzky-Golay filter with a width of 9 and 'mirror' padding, similar to [10] and [12]. The above three steps are summarized as step 3 in Figure 2.

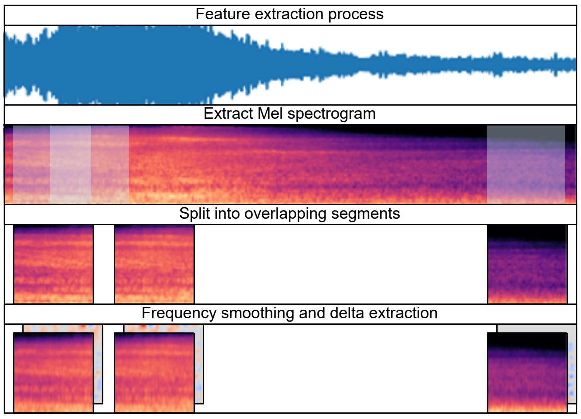
Feature extraction process
Extract Mel spectrogram
Split into overlapping segments
Frequency smoothing and delta extraction

Figure 2: Overview of the feature extraction process for aircraft classification in scenario 1.

Scenario 2: Aircraft Classification

In the second scenario, a single network is trained to simultaneously filter noise and classify incoming sound. This network requires two inputs; the first is the noisy Mel spectrum containing aircraft/non-aircraft sound and MAV ego-noise, the second the sequence of state vectors corresponding to the ego-noise. To simulate noise-contaminated recordings, the noise-free spectra extracted from the ESC-50 subset for scenario 1 (obtained from step 1 in Figure 2) are mixed with MAV noise. Each 'clean' spectrum is mixed with a random segment of a random ego-noise spectrum from the same respective subset (train/validation/test) at a constant mix ratio. Given a clean spectrum **C** and an ego-noise spectrum **R**, mixed at a ratio r, the mix **M** is generated as follows:

$$\mathbf{M} = \mathbf{C} + r \cdot \mathbf{N}. \tag{1.2}$$

Unlike the spectra generated for the clean dataset (Figure 2), the resulting mix is not scaled back to a [0, 1] interval. Studies concerning robust audio classification, which investigate classification performance in mismatched conditions, commonly vary their signal-to-noise ratio (SNR) from 20 to as low as 0 dB [11]. Because the mixing in scenario 2 is done with scaled log-power spectra and not audio waveforms, the term SNR is not applicable here. Instead an r of 1 is chosen to resemble 0 dB SNR.

The mixed recording and corresponding state sequence are then split into segments, as per step 2 of Figure 2. Unlike the spectra generated for the clean dataset, the resulting mix is not scaled back to a [0, 1] interval. Frequency smoothing and delta extraction are applied to each segment as per step 3 in Figure 2. Examples of noise-free and mixed features belonging to the 'airplane' and 'engine' class are shown in Figure 3 and Figure 4, respectively.

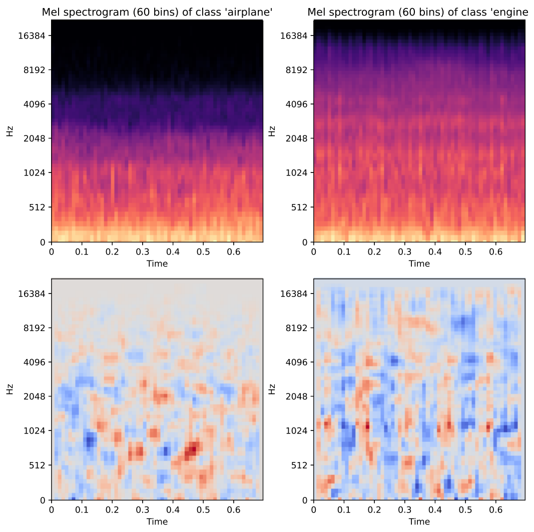

Figure 3: Example features of an aircraft segment (airplane) and a non-aircraft segment (engine), without added MAV noise.

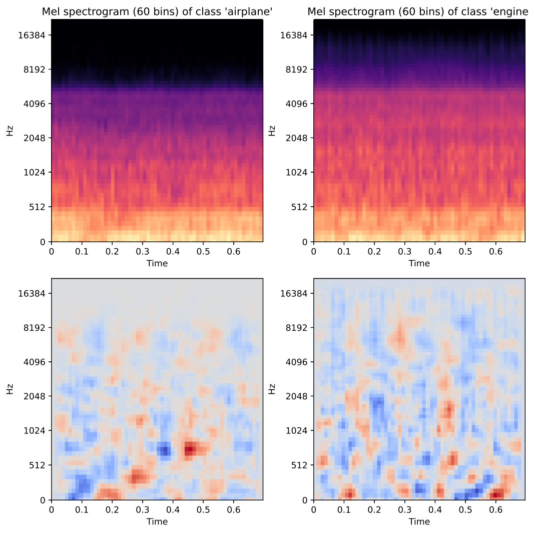

Figure 4: Example features of an aircraft segment (airplane) and a non-aircraft segment (engine), mixed with MAV noise at a ratio of 1.00.

Scenario 1 and 2: Mismatched Conditions

To test the robustness of the classifier, additional evaluation sets are created in mismatched noise conditions. In mismatched conditions, the amount of noise present during training (if any) differs from the noise at evaluation. For scenario 2, where the network is trained at r=1.00, additional evaluation sets with r = {0.75, 0.50, 0.25} are created.

Because the classifier from scenario 1 is trained on noise-free input (r=0) and does not have access to the state vector to suppress the additional noise, mismatched conditions are created by evaluating on the denoised estimate of the clean spectrum. This estimate $\hat{\mathbf{C}}$ is obtained by subtracting the predicted ego-noise spectrum $\mathbf{P}$ from the mixture $\mathbf{M}$ (obtained via Equation (1.2)), at the appropriate noise ratio r:

$$\hat{\mathbf{C}} = \mathbf{M} - r \cdot \mathbf{P}. \tag{1.3}$$

To compare the robustness with scenario 2, $r = \{0.25, 0.50, 0.75, 1.00\}$ is used to generate the mismatched evaluation datasets for scenario 1. Additionally, a fourth augmentation set is generated solely for scenario 1, consisting of a denoised mixture at r=1.00.

IV Experiments

This section outlines the chosen model architecture and assesses the outcome of the experiments regarding scenario 1 and 2. Section IV-A covers scenario 1, which consists of separate aircraft classification and ego-noise prediction. Scenario 2, where aircraft classification and ego-noise suppression is done simultaneously, is covered in section IV-B.

A Scenario 1

Ego-Noise Prediction

Two types of networks have been considered for the ego-noise prediction. The first is a Multi-Layer Perceptron (MLP), a regular feedforward network consisting only of fully-connected (dense) layers. The second is a Recurrent Neural Network (RNN), which consists of recurrent layers followed by fully-connected layers. The recurrent layer allows for learning long-term temporal dependencies found in the data by sharing parameters across several time steps. In the context of ego-noise prediction, a time sequence containing the previous and current state vectors can be used to predict the current ego-noise spectrum. Because the standard recurrent layer suffers from vanishing and exploding gradients, its activation function is replaced by a memory cell to mitigate these problems. Two common cells are the Gated Recurrent Unit (GRU) [27] and the Long Short Term Memory (LSTM) [28], both of which will be investigated during the experiment.

Bayesian Optimization (BO) [29] has been used to find the optimal sequence length, network architecture (layer width, depth) and network type (MLP, GRU or LSTM). The advantage of BO over a grid search is that observations from previous evaluations determine the next parameterization to be evaluated. For each sequence length (1-10) and network type, 20 searches were conducted through the network depth and width. Each recurrent layer was bounded between 40 and 120 units, while each fully-connected layer was bounded between 40 and 200 units. A layer depth of 1-3 was considered for the MLP, while the GRU and LSTM would consist of 1-2 recurrent layers followed by 1-2 fully-connected layers.

The training for each parameterization was identical. The AdamW algorithm[30] was used with a learning rate of 0.001 and weight decay of 0.01. Training was done for 20 epochs, using a batch size of 256. The lowest validation loss achieved by each combination of sequence length and network type after 20 epochs is tabulated in Table 2. A recurrent network using GRU cells in the recurrent layer, with a sequence length of 7 leads to the best performance. This network contains two recurrent layers with 120 and 90 units, followed by two dense layers containing 190 and 160 units, respectively. To provide the predicted ego-noise spectra, the network was trained for 200 epochs, with an early stopping patience of 25 epochs.

Ego-noise spectra are illustrated in Figure 5. The top row indicates the original spectrum, the middle row the predicted spectrum, and the bottom row is the residual, obtained by subtracting the predicted spectrum from the original. It appears the model estimates the ego-noise considerably well. The dominant frequency, fluctuating around 400 Hz, as well as higher order harmonics are followed adequately. The model also notices the instances of broadband noise around the 15- and 36-second mark. The abrupt changes in direction caused by the autopilot setting a new waypoint, most notable around the 4-, 13-, 19- and 23-second mark, are also compensated for by the model.

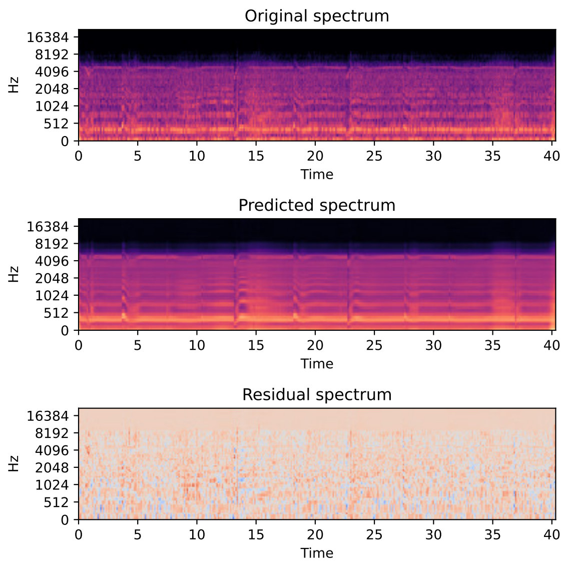

Figure 5: MAV ego-noise spectra: original (top), predicted (middle) and residual (bottom).

	1	2	3	4	5	6	7	8	9	10
MLP	1.535	1.549	1.572	1.562	1.576	1.558	1.569	1.575	1.561	1.565
GRU	1.561	1.561	1.546	1.524	1.552	1.517	**1.512**	1.540	1.550	1.519
LSTM	1.567	1.550	1.537	1.540	1.533	1.534	1.531	1.520	1.536	1.523

Table 2: Overview of the lowest MSE loss (1e-3) of each type of ego-noise model for each state vector sequence length.

Aircraft Classification

The network chosen for the aircraft classification is naturally a CNN, as this type of model has achieved state-of-the-art performance in audio classification problems when combined with spectrogram input. While many network configurations are readily available through prior research, these networks are typically used for multi-class or multi-label classification of datasets orders of magnitude larger than the one used in this article, and are therefore far more complex than required. Instead, the architecture of the final network was obtained via trial and error by manually testing combinations with varying input shape, filter size, layer depth and layer width. To speed up this process, the optimal network configuration was selected by evaluating on the dataset without augmentation, which is approximately 14 times smaller than the fully augmented dataset.

The best performing architecture consist of three convolutional layers, followed by two fully-connected layers and one linear output layer. The first two convolutional layer contain 16 output channels, with a kernel size of (5, 5) and a kernel dilation of (2, 2). An advantage of the dilated kernels is that they offer an increased receptive field (9 by 9) without the increase in model complexity that comes with a larger kernel [12]. Both layers are also followed by (2, 2) max pooling with identical stride, which is a common procedure to down-sample the layer output while retaining the features that trigger a strong response. The third and final convolutional layer contains 32 output channels and also has a kernel shape of (5, 5), but contains neither dilation nor is it followed by pooling. All convolutional layers use the Rectified Linear Unit (ReLU) as the activation function between layers. Additionally, batch normalization[31] is applied before the ReLU operation of each convolutional layer, speeding up the learning. The two fully-connected layers that follow contain 128 and 32 output units, respectively. Both layers also use ReLU activation, as well as a dropout of 0.5 during training to mitigate over-fitting. The output layer is linear, with a sigmoid function which transforms the output to a score between 0 and 1. An overview of the model is given in the top section of Figure 6.

The experiment considers the performance and robustness of the classifier with and without data augmentation. After optimization using limited training data, the network is re-trained once for each of the four individual augmentations, and once with all augmentation data, to assess whether the augmentations are beneficial. Each of these six models was then evaluated in mismatched denoised conditions, which were the result of the mismatch between predicted and actual ego-noise. These tests were also repeated under regular mixed conditions (i.e. without ego-noise reduction) to demonstrate the effects of the ego-noise reduction on classification performance.

Each model is trained in identical fashion. The AdamW algorithm[30] is used for training, which is an adaptation of the established Adam algorithm that decouples the gradient update and weight decay. Training is done for 100 epochs with a learning rate of 0.0001 and a weight decay of 0.01. Early stopping terminates training if the validation loss does not improve after 25 epochs. The batch size used for training is 256.

The models that performed best in at least one of the evaluation conditions (clean or denoised) are tabulated in Table 3. In the table, two metrics of accuracy are given. The first is the segment-based accuracy, which is simply the percentage of the input data that is classified correctly. The second metric is the recording-based accuracy. The classification score of a recording is obtained by averaging the classification scores of its individual segments. For example, if one or two out of the 13 segments in a recording are incorrectly classified, the complete recording is still likely to be classified correctly. As such, the recording-based accuracy is generally higher than the segment-based accuracy.

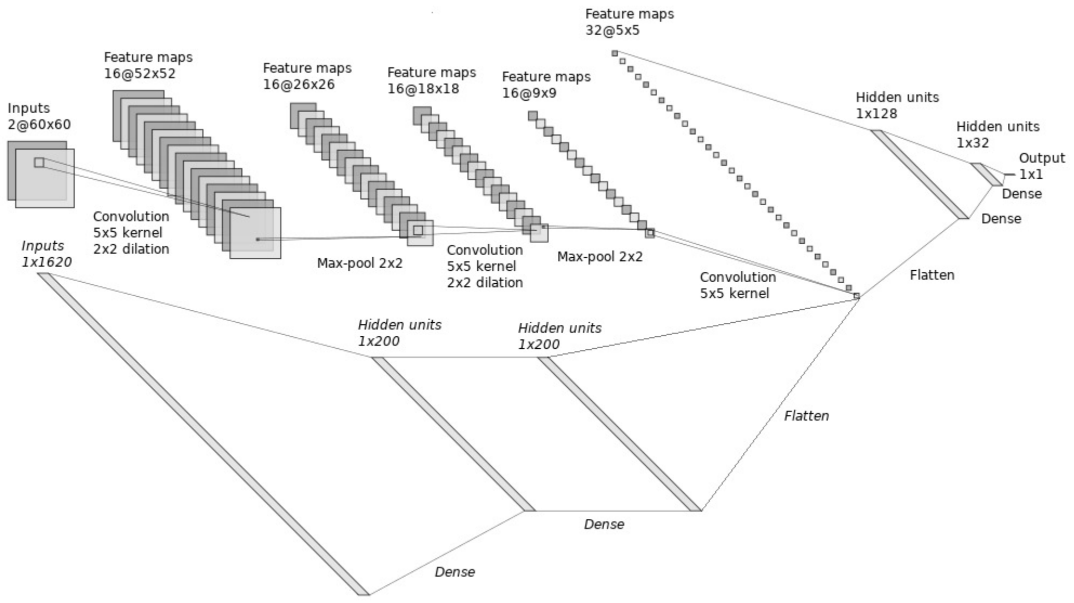

Figure 6: Architecture of the CNN used for aircraft classification in scenario 1 and 2. The second input, indicated in cursive, is only used in scenario 2.

	None	Class Mix	All
Clean	**95.0% (97.5%)**	91.0% (95.0%)	91.5% (92.5%)
0.25D	**95.0% (97.5%)**	92.2% (95.0%)	91.5% (92.5%)
0.50D	**93.7% (97.5%)**	91.7% (95.0%)	91.5% (92.5%)
0.75D	**90.5% (97.5%)**	**90.7% (95.0%)**	**91.2% (92.5%)**
1.00D	83.9% (95.0%)	**87.4% (95.0%)**	**88.7% (92.5%)**
0.25M	89.2% (90.0%)	86.2% (87.5%)	85.2% (87.5%)
0.50M	77.9% (77.5%)	80.4% (82.5%)	76.1% (77.5%)
0.75M	68.6% (70.0%)	74.4% (75.0%)	71.4% (67.5%)
1.00M	56.5% (47.5%)	64.8% (65.0%)	64.3% (57.5%)

Table 3: Influence of data augmentation on model performance under clean, denoised (D) and mixed (M) conditions.

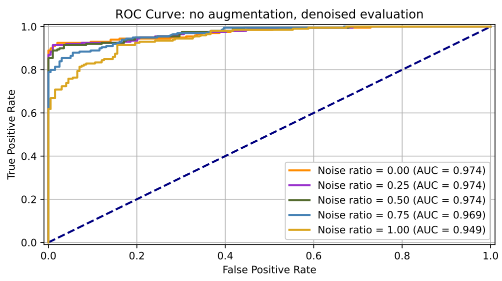

Figure 7: Scenario 1: ROC curves for segment-based performance, trained without data augmentation and evaluated on denoised data.

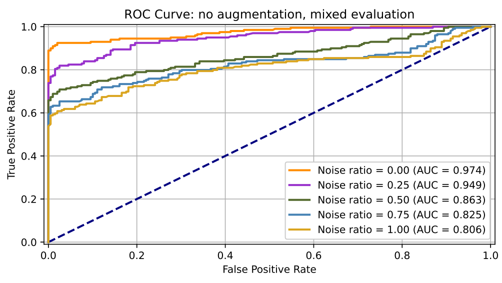

Figure 8: Scenario 1: ROC curves for segment-based performance, trained without data augmentation and evaluated on mixed data.

The network trained without data augmentation achieves the best performance for the clean dataset, with 95.0% and 97.5% accuracy for segment- and recording-based accuracy, respectively. As the mismatch increases, the models trained with intra-class mixing augmentation and all augmentations overtake its segment-based accuracy. By comparing the top- and bottom half of Table 3, the benefits of the denoising approach become clear. Without denoising, the classifier would only achieve a recording-based accuracy of 47.5% at the highest noise setting, compared to 95.0% at the equivalent denoised setting.

The ROC curves of the classifier trained without any augmentation are given in Figure 7 and Figure 8, evaluated on denoised and mixed settings, respectively. Comparing both figures reaffirms the benefits of denoising the mixed data prior to classification.

Although binary classification is used, some insight into classification of the original class labels is still retained. A detailed breakdown of classification performance per category (i.e. 'airplane' classified as 'aircraft', 'engine' classified as 'non-aircraft', etc.) is provided in appendix A for each of the six models at all noise settings.

B Scenario 2

In this scenario, ego-noise filtering and aircraft classification is done in a single network. This network consists of two inputs: the two-channel spectrum contaminated with ego-noise and the states corresponding to the ego-noise. The architecture of the network is depicted in Figure 6. The top half of the figure indicates the forward pass of the spectral input, which is identical to the network used in scenario 1. The state input is flattened (27x60 → 1620) and passes through two fully-connected layers, both with ReLU activations, with a dropout of 0.2 on the first layer and 0.5 on the second layer. The state input then merges with the flattened spectral input after the third convolutional layer.

The network is trained and evaluated five times: once without augmentation, once for each augmentation, and once with all augmentations given in Section III-A. Each network is trained with a mix ratio r of 1.00, and evaluated in the mismatched conditions $r = \{0.75, 0.50, 0.25\}$. The training configuration is identical to scenario 1: AdamW is used with a learning rate of 0.0001 and a weight decay of 0.01 and training ends after 100 epochs with an early stopping patience of 25.

The network trained with all augmentations achieved the best results in matching conditions, and its performance is tabulated in Table 4. With a segment-based accuracy of 91.7% and a recording-based accuracy of 92.5% in matching conditions, its performance is not dissimilar to the networks of scenario 1 in Table 3. The ROC curve belonging to this network is plotted in Figure 9.

	All augmentations
r=1.00	91.7% (92.5%)
r=0.75	77.1% (75.0%)
r=0.50	63.8% (65.0%)
r=0.25	60.6% (57.5%)

Table 4: Scenario 2: classification performance in matching and mismatched conditions.

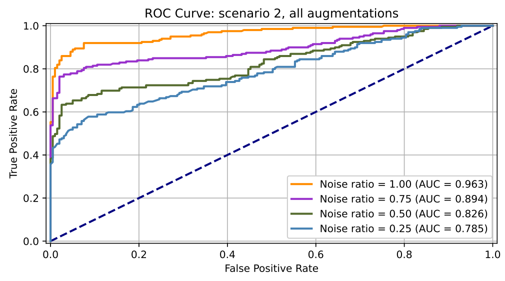

Figure 9: Scenario 2: ROC curves for segment-based performance, trained with all augmentations and evaluated in matching and mismatched conditions.

Unfortunately, the network is unable to adapt when faced with data generated at a different noise setting and Table 4 shows that its performance quickly degrades. In the evaluations presented here, the network likely continues to overcompensate for the noise level it learned during training. While the spectrum of each input feature changes at different noise settings, the state vector remains constant. Thus, the network has no way of knowing that the incoming spectrum contains less noise, making the implicit denoising network infeasible in practice. When denoising and classification is separated as in scenario 1, any increase or decrease in noise is accompanied by a change in the state vector, and the noise can continue being attenuated by the denoising network.

V Conclusions

In this paper, a novel ANN-based approach has been presented which mitigates the effects of MAV ego-noise on aircraft detection by using on-board sensor data to predict the noise spectrum recorded by a single microphone. To the author's best knowledge, this is the first research which applies ANNs to reduce ego-noise of an in-flight MAV.

Two aircraft detection pipelines are outlined in the paper. In the first approach, noise reduction and aircraft detection is done separately. An RNN with GRU cells is used to predict the Mel spectrum corresponding to a sequence of MAV flight data. This predicted spectrum is then subtracted from the spectrum of a mixed recording, to obtain a denoised recording that is split into shorter segments and fed into a CNN-based classifier. The second approach merges the denoising and classification stages into a single ANN. The mixed spectra are fed through convolutional layers, while the corresponding MAV flight data is fed through regular feedforward layers and then merged with the output of the final convolutional layer.

Experiments have been performed in matching and mismatched noise conditions. The first approach achieves an accuracy of 97.5% in the clean configuration, which decreases to 95.0% on the highest noise setting when denoising is applied. Without the denoising step, accuracy is a mere 47.5%, signifying the importance of denoising the noise-contaminated input prior to classification. The second approach achieves an accuracy of 92.5% in matching conditions, but is unable to adapt to the mismatched conditions.

The mismatched conditions evaluated in this article are not dissimilar to robust SED with background noise. There, the SNR typically does not go lower than 0dB, which is approximated by setting the highest noise ratio to 1.00. It is unknown what a realistic SNR (or mix ratio) is for in-flight aircraft classification. Moreover, recording with an in-flight MAV in the proximity of other aircraft composes a serious safety hazard. The primary contribution of this paper is therefore to illustrate the benefits of ANN-based MAV ego-noise reduction on aircraft detection.

Lastly, the created ego-noise dataset, as well as the code used to generate the results is publicly available on https://github.com/mvanderwoude/aircraft_detector.

Bibliography

[1] G. Wild, J. Murray, and G. Baxter, “Exploring civil drone accidents and incidents to help prevent potential air disasters,” *Aerospace*, vol. 3, no. 3, p. 22, 2016.

[2] M. G. Wu, A. C. Cone, S. Lee, C. Chen, M. W. Edwards, and D. P. Jack, “Well clear trade study for unmanned aircraft system detect and avoid with non-cooperative aircraft,” in *2018 Aviation Technology, Integration, and Operations Conference*, p. 2876, 2018.

[3] G. Ludeno, I. Catapano, G. Gennarelli, F. Soldovieri, A. R. Vetrella, A. Renga, and G. Fasano, “A micro-uav-borne system for radar imaging: A feasibility study,” in *2017 9th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*, pp. 1–4, June 2017.

[4] T. G. McGee, R. Sengupta, and K. Hedrick, “Obstacle detection for small autonomous aircraft using sky segmentation,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 4679–4684, April 2005.

[5] T. Zsedrovits, A. Zarandy, B. Vanek, T. Peni, J. Bokor, and T. Roska, “Visual detection and implementation aspects of a uav see and avoid system,” in *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, pp. 472–475, IEEE, 2011.

[6] A. Rozantsev, V. Lepetit, and P. Fua, “Detecting flying objects using a single moving camera,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 879–892, May 2017.

[7] S. Hwang, J. Lee, H. Shin, S. Cho, and D. Shim, “Aircraft detection using deep convolutional neural network in small unmanned aircraft systems,” in *2018 AIAA Information Systems-AIAA Infotech Aerospace*, 01 2018.

[8] E. Tijs, G. de Croon, J. Wind, B. Remes, C. De Wagter, H. de Bree, and R. Ruijsink, “Hear-and-avoid for micro air vehicles,” in *Proceedings of the International Micro Air Vehicle Conference and Competitions (IMAV), Braunschweig, Germany*, vol. 69, 2010.

[9] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*. Springer, 2018.

[10] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2015.

[11] H. Zhang, I. McLoughlin, and Y. Song, “Robust sound event recognition using convolutional neural networks,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 559–563, April 2015.

[12] X. Zhang, Y. Zou, and W. Shi, “Dilated convolution neural network with leakyrelu for environmental sound classification,” in *2017 22nd International Conference on Digital Signal Processing (DSP)*, pp. 1–5, IEEE, 2017.

[13] 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, 2017.

[14] J. Salamon, C. Jacoby, and J. P. Bello, “A dataset and taxonomy for urban sound research,” in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 1041–1044, 2014.

[15] K. J. Piczak, “Esc: Dataset for environmental sound classification,” in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018, 2015.

[16] C. Asensio, M. Ruiz, and M. Recuero, “Real-time aircraft noise likeness detector,” *Applied Acoustics*, vol. 71, no. 6, pp. 539–545, 2010.

[17] D. Wijnker, T. van Dijk, M. Snellen, G. de Croon, and C. De Wagter, “Hear-and-avoid for uavs using convolutional neural networks,” in *Proceedings of the 11th International Micro Air Vehicle Competition and Conference (IMAV2019), Madrid, Spain*, vol. 30, 2019.

[18] G. Sinibaldi and L. Marino, “Experimental analysis on the noise of propellers for small uav,” *Applied Acoustics*, vol. 74, no. 1, pp. 79 – 88, 2013.

[19] L. Wang and A. Cavallaro, “Ear in the sky: Ego-noise reduction for auditory micro aerial vehicles,” in *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 152–158, Aug 2016.

[20] T. Ishiki and M. Kumon, “Design model of microphone arrays for multirotor helicopters,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6143–6148, Sep. 2015.

[21] L. Wang and A. Cavallaro, “Microphone-array ego-noise reduction algorithms for auditory micro aerial vehicles,” *IEEE Sensors Journal*, vol. 17, no. 8, pp. 2447–2455, 2017.

[22] L. Wang and A. Cavallaro, “Acoustic sensing from a multi-rotor drone,” *IEEE Sensors Journal*, vol. 18, no. 11, pp. 4570–4582, 2018.

[23] L. Wang, R. Sanchez-Matilla, and A. Cavallaro, “Tracking a moving sound source from a multirotor drone,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2511–2516, IEEE, 2018.

[24] P. Marmaroli, X. Falourd, and H. Lissek, “A uav motor denoising technique to improve localization of surrounding noisy aircrafts: proof of concept for anti-collision systems,” in *Acoustics 2012*, 2012.

[25] G. Ince, K. Nakadai, and K. Nakamura, “Online learning for template-based multi-channel ego noise estimation,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3282–3287, IEEE, 2012.

[26] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780, IEEE, 2017.

[27] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[28] F. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” *Neural computation*, vol. 12, pp. 2451–71, 10 2000.

[29] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, “BoTorch: Programmable Bayesian Optimization in PyTorch,” *arXiv e-prints*, p. arXiv:1910.06403, Oct. 2019.

[30] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.

[31] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

Appendix A

Detailed Classification Breakdown

The tables found in this appendix show a detailed breakdown of the classification accuracy of the networks trained for scenario 1. In Table A.1, performance of all network in clean and denoised settings is summarized. Table A.2 to Table A.7 show the detailed classification breakdown for the network with no augmentation, class mix augmentation, pitch shift augmentation, time stretch augmentation, denoised augmentation and all augmentations, respectively. Generally speaking, the networks have the most trouble detecting helicopters, especially as the mismatch increases.

	None	Class Mix	Pitch Shift	Time Stretch	Denoised	All
Clean	95.0% (97.5%)	91.0% (95.0%)	91.5% (95.0%)	92.0% (95.0%)	89.7% (92.5%)	91.5% (92.5%)
0.25D	95.0% (97.5%)	92.2% (95.0%)	90.5% (95.0%)	92.2% (95.0%)	90.5% (95.0%)	91.5% (92.5%)
0.50D	93.7% (97.5%)	91.7% (95.0%)	87.9% (95.0%)	90.7% (95.0%)	89.7% (95.0%)	91.5% (92.5%)
0.75D	90.5% (97.5%)	90.7% (95.0%)	85.4% (92.5%)	88.4% (95.0%)	88.2% (92.5%)	91.2% (92.5%)
1.00D	83.9% (95.0%)	87.4% (95.0%)	79.9% (85.0%)	83.4% (90.0%)	86.2% (87.5%)	88.7% (92.5%)

Table A.1: General overview of the classification performance for various data augmentations.

	Overall	Airplane	Helicopter	Engine	Train	Wind
Clean	95.0% (97.5%)	97.1% (100%)	84.2% (87.5%)	100%	100%	98.1% (100%)
0.25D	95.0% (97.5%)	97.1% (100%)	84.2% (87.5%)	100%	100%	98.1% (100%)
0.50D	93.7% (97.5%)	94.2% (100%)	82.1% (87.5%)	100%	100%	98.1% (100%)
0.75D	90.5% (97.5%)	91.4% (100%)	73.7% (87.5%)	100%	100%	95.2% (100%)
1.00D	83.9% (95.0%)	74.0% (87.5%)	68.4% (87.5%)	100%	100%	95.2% (100%)
0.25M	89.2% (90.0%)	88.5% (87.5%)	70.5% (62.5%)	100%	95.7% (100%)	99.0% (100%)
0.50M	77.9% (77.5%)	86.5% (87.5%)	71.6% (75.0%)	95.8% (100%)	57.5% (50.0%)	76.0% (75.0%)
0.75M	68.6% (70.0%)	84.6% (87.5%)	83.2% (87.5%)	79.2% (25.0%)	21.3% (25.0%)	55.8% (62.5%)
1.00M	56.5% (47.5%)	85.6% (87.5%)	86.3% (87.5%)	56.3% (37.5%)	10.6% (12.5%)	21.2% (12.5%)

Table A.2: Detailed classification performance of the network trained without augmentation at clean, denoised (D) and mixed (M) settings.

	Overall	Airplane	Helicopter	Engine	Train	Wind
Clean	91.0% (95.0%)	88.5% (87.5%)	80.0% (87.5%)	95.8% (100%)	100%	97.1% (100%)
0.25D	92.2% (95.0%)	88.5% (87.5%)	82.1% (87.5%)	95.8% (100%)	100%	100%
0.50D	91.7% (95.0%)	86.5% (87.5%)	81.0% (87.5%)	97.9% (100%)	100%	100%
0.75D	90.7% (95.0%)	83.7% (87.5%)	80.0% (87.5%)	97.9% (100%)	100%	100%
1.00D	87.4% (95.0%)	76.9% (87.5%)	79.0% (87.5%)	97.9% (100%)	100%	95.2% (100%)
0.25M	86.2% (87.5%)	87.5%	56.8% (62.5%)	100%	97.9% (87.5%)	100%
0.50M	80.4% (82.5%)	85.6% (87.5%)	65.3% (62.5%)	100%	74.5% (87.5%)	82.7% (75.0%)
0.75M	74.4% (75.0%)	84.6% (87.5%)	77.9% (87.5%)	91.7% (87.5%)	36.2% (37.5%)	70.2% (75.0%)
1.00M	64.8% (65.0%)	84.6% (87.5%)	85.3% (87.5%)	70.8% (75.0%)	17.0% (25.0%)	45.2% (50.0%)

Table A.3: Detailed classification performance of the network trained with class mix augmentation at clean, denoised (D) and mixed (M) settings.

	Overall	Airplane	Helicopter	Engine	Train	Wind
Clean	91.5% (95.0%)	87.5%	82.1% (87.5%)	100%	100%	96.2% (100%)
0.25D	90.5% (95.0%)	87.5%	77.9% (87.5%)	100%	100%	96.2% (100%)
0.50D	87.9% (95.0%)	85.6% (87.5%)	69.5% (87.5%)	100%	100%	96.2% (100%)
0.75D	85.4% (92.5%)	76% (87.5%)	67.4% (75.0%)	100%	100%	98.1% (100%)
1.00D	79.9% (85.0%)	64.4% (75.0%)	56.8% (50.0%)	100%	100%	98.1% (100%)
0.25M	85.7% (87.5%)	87.5% (87.5%)	66.3% (62.5%)	100%	85.1% (87.5%)	95.2% (100%)
0.50M	75.6% (72.5%)	87.5% (87.5%)	76.8% (75.0%)	93.8% (100%)	38.3% (37.5%)	71.2% (62.5%)
0.75M	63.8% (62.5%)	87.5% (87.5%)	83.2% (87.5%)	60.4% (62.5%)	14.9% (12.5%)	46.2% (62.5%)
1.00M	54.3% (47.5%)	86.5% (87.5%)	86.3% (87.5%)	45.8% (37.5%)	6.38% (12.5%)	18.3% (12.5%)

Table A.4: Detailed classification performance of the network trained with pitch shift augmentation at clean, denoised (D) and mixed (M) settings.

	Overall	Airplane	Helicopter	Engine	Train	Wind
Clean	92.0% (95.0%)	88.5% (87.5%)	80.0% (87.5%)	100%	100%	99.0% (100%)
0.25D	92.2% (95.0%)	88.5% (87.5%)	81.1% (87.5%)	100%	100%	99.0% (100%)
0.50D	90.7% (95.0%)	87.5%	76.8% (87.5%)	100%	100%	98.1% (100%)
0.75D	88.4% (95.0%)	86.5% (87.5%)	68.4% (87.5%)	100%	100%	98.1% (100%)
1.00D	83.4% (90.0%)	75.0% (87.5%)	61.1% (62.5%)	100%	100%	97.1% (100%)
0.25M	86.9% (87.5%)	87.5% (87.5%)	61.1% (62.5%)	100%	95.7% (87.5%)	100%
0.50M	79.6% (82.5%)	84.6% (87.5%)	71.6% (75.0%)	95.8% (100%)	48.9% (62.5%)	88.5% (87.5%)
0.75M	71.1% (72.5%)	84.6% (87.5%)	78.9% (87.5%)	79.2% (87.5%)	19.1% (25.0%)	70.2% (75.0%)
1.00M	60.8% (55.0%)	84.6% (87.5%)	84.2% (87.5%)	52.1% (37.5%)	12.8% (12.5%)	41.3% (50.0%)

Table A.5: Detailed classification performance of the network trained with time stretch augmentation at clean, denoised (D) and mixed (M) settings.

	Overall	Airplane	Helicopter	Engine	Train	Wind
Clean	89.7% (92.5%)	94.2% (100%)	70.5% (75.0%)	100%	100%	93.3% (87.5%)
0.25D	90.5% (95.0%)	94.2% (100%)	73.7% (87.5%)	100%	100%	93.3% (87.5%)
0.50D	89.7% (95.0%)	93.3% (100%)	73.7% (87.5%)	100%	100%	91.3% (87.5%)
0.75D	88.2% (92.5%)	89.4% (100%)	71.6% (75.0%)	100%	100%	91.3% (87.5%)
1.00D	86.2% (87.5%)	84.6% (87.5%)	69.5% (62.5%)	100%	100%	90.4% (87.5%)
0.25M	86.2% (90.0%)	85.6% (87.5%)	61.1% (62.5%)	100%	100%	97.1% (100%)
0.50M	78.9% (82.5%)	83.7% (87.5%)	71.6% (75.0%)	95.8% (100%)	57.4% (62.5%)	82.7% (87.5%)
0.75M	65.8% (62.5%)	84.6% (87.5%)	76.8% (75.0%)	66.7% (62.5%)	19.1% (25.0%)	57.7% (62.5%)
1.00M	56.8% (52.5%)	84.6% (87.5%)	85.3% (87.5%)	47.9% (50.0%)	10.6% (12.5%)	27.9% (25.0%)

Table A.6: Detailed classification performance of the network trained with denoised augmentation at clean, denoised (D) and mixed (M) settings.

	Overall	Airplane	Helicopter	Engine	Train	Wind
Clean	91.5% (92.5%)	88.5% (87.5%)	76.8% (75.0%)	100%	100%	100%
0.25D	91.5% (92.5%)	88.5% (87.5%)	76.8% (75.0%)	100%	100%	100%
0.50D	91.5% (92.5%)	88.5% (87.5%)	76.8% (75.0%)	100%	100%	100%
0.75D	91.2% (92.5%)	88.5% (87.5%)	75.8% (75.0%)	100%	100%	100%
1.00D	88.7% (92.5%)	83.7% (87.5%)	70.5% (75.0%)	100%	100%	100%
0.25M	85.2% (87.5%)	87.5%	54.7% (62.5%)	100%	93.6% (87.5%)	100%
0.50M	76.1% (77.5%)	86.5% (87.5%)	64.2% (75.0%)	85.4% (100%)	48.9% (50.0%)	84.6% (75.0%)
0.75M	71.4% (67.5%)	84.6% (87.5%)	78.9% (87.5%)	68.8% (62.5%)	25.5% (25.0%)	73.1% (75.0%)
1.00M	64.3% (57.5%)	85.6% (87.5%)	84.2% (87.5%)	50% (37.5%)	12.8% (12.5%)	54.8% (62.5%)

Table A.7: Detailed classification performance of the network trained with all augmentations at clean, denoised (D) and mixed (M) settings.

Appendix B

Simplifying the Ego-Noise Predictor

This appendix briefly investigates the influence of model simplification on the ability to predict ego-noise, and the influence of the subsequent change in ego-noise prediction on the classification performance in mismatched denoised conditions.

The proposed network is a simple feedforward network consisting of only 1 hidden layer with 100 hidden units, which only uses the 4 rotor rpms to predict the ego-noise. In comparison, the ideal network obtained through Bayesian Optimization, as defined in the article, consists of 2 GRU layers and 2 dense layers, and uses all 27 available states as input.

It is not surprising that the simple network performs worse, with an MSE loss of 1.630e-3 compared to an MSE loss of 1.507e-3 for the optimized network. Nevertheless, the simple network (Figure B.1) still predicts adequately. For comparison, the optimal network is shown in Figure B.2.

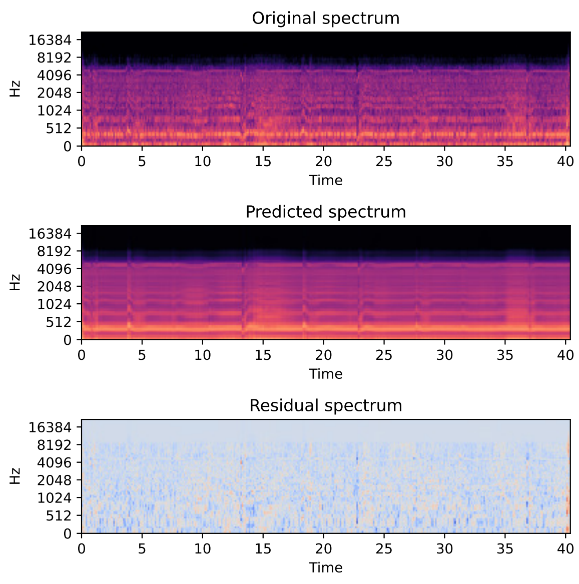

Figure B.1: MAV ego-noise spectra obtained from the simple network.

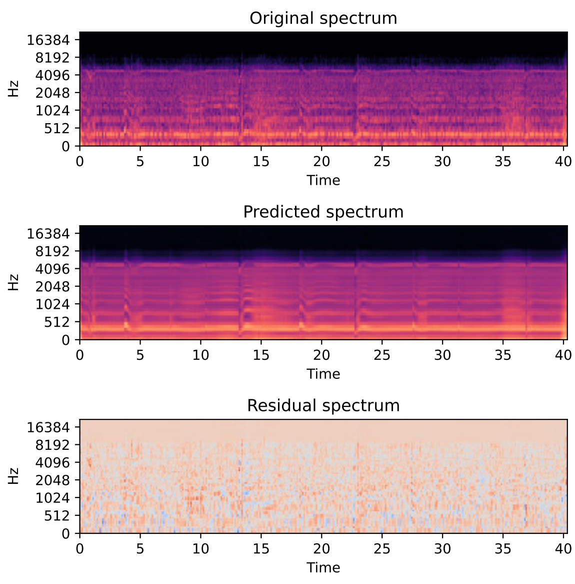

Figure B.2: MAV ego-noise spectra obtained from the optimized network.

The spectra predicted by the simplified network can now be subtracted from the mixed dataset at the given noise ratios (r={0.25, 0.50, 0.75, 1.00}) to obtain unique sets of denoised spectra. A comparison in classification performance between these sets and the denoised sets used in the article is displayed in Table B.1. From this it becomes apparent that the simplified model does hinder the accuracy as the mismatch increases.

	Simplified	Optimized
0.25D	95.0% (97.5%)	95.0% (97.5%)
0.50D	93.2% (97.5%)	93.7% (97.5%)
0.75D	88.9% (97.5%)	90.5% (97.5%)
1.00D	83.7% (92.5%)	83.9% (95.0%)

Table B.1: Classification accuracy in mismatched conditions using denoised data obtained from the simplified network (left) and optimized network (right).

II

Literature Review

Chapter 2

Properties and Representation of Aircraft Sound

Sound is essentially a vibration propagating through a medium as a weak, longitudinal wave of pressure. Since the particle displacements occur in the same direction as the movement of the wave, this produces a series of local compressions and rarefactions. The difference between the ambient pressure and instantaneous pressure caused by these disturbances is denoted as the sound pressure. A sound signal is considered discrete when it is dominant at one or several frequencies, while broadband sound has a varying frequency distribution.

A brief overview of the generation of aircraft sound is given in Section 2.1, while Section 2.2 contains required background knowledge for the digital processing of sound. Finally, Section 2.3 gives an overview of historical and state-of-the-art feature extraction methods for sound classification.

2.1 Aircraft Sound Sources

The aircraft sound perceived on the ground is generated primarily by its engine, while airframe noise is also of significance [19]. The noise characteristic thus depend on the engine type, of which several classes can be distinguished. Small aircraft for general aviation generally make use of a piston engine in combination with a propeller, while commercial and military aircraft use some kind of turbo engine (jet, prop or fan).

As noted by [19], an N-cylinder piston engine operating at an RPM of n_e generates noise predominantly at the exhaust frequency

$$f_e = \frac{Nn_e}{120}, \tag{2.1}$$

as well as at integer multiples (harmonics) of the cylinder firing frequency

$$f_c = \frac{n_e}{120}. \tag{2.2}$$

Each piston engine consists of either 4 or 6 cylinders, while the RPM in cruise typically lies between 2-3,000.

A propeller generates discrete rotational noise, caused by the periodic excitation of the air within the propeller disk by the blades. This noise occurs at harmonics of its blade passage frequency

$$f_1 = \frac{Bn_p}{60}, \tag{2.3}$$

where B indicates the number of blades and n_p the propeller rpm. The propeller also causes weak broadband vortex noise, due to random airflow disturbances caused by the blades. The total noise generated by a piston-prop engine can be considered omnidirectional.

Turbojet and turbofan engines generate noise via the compressor, fan, turbine, combustor and exhaust jet. All noise sources are highly directional noise, at ± 45deg relative to the direction of motion of the aircraft. As a result, noise of a turbojet/fan aircraft appears attenuated when directly overhead. On approach, fan noise is dominant, while exhaust noise is dominant when the aircraft recedes. A fan with subsonic blade tip speed generates predominantly discrete noise at harmonics of the fan blade passage frequency (Equation (2.3). When supersonic tip speeds occur, shock waves are created which cause additional noise, harmonic with the engine rpm. The noise of the jet exhaust is broadband, and generated from the high-velocity exhaust air mixing with the ambient air.

Airframe noise is aerodynamic noise generated by the turbulent flow of air over the surface of the aircraft, with cavities such as high-lift devices causing substantial radiation. As such, airframe noise dominates during landing approaches.

2.1.1 Doppler Effect

A moving sound source such as an aircraft causes an observed frequency which differs from the actual frequency of the sound emitted. This phenomenon, dubbed the Doppler effect, is caused by the continuously changing distance between sound source and receiver. The ratio between observed frequency f' and true frequency f can be expressed as follows:

$$\frac{f'}{f} = \frac{1}{1 + \frac{dr/dt}{c}}, \tag{2.4}$$

where dr/dt indicates the (time) rate of change in distance from source to receiver, and c is the speed of sound. As such, f' increases when the source moves towards the observer, and decreases when it moves away from the observer. The change in observed frequency $(f' - f)$ is referred to as the Doppler shift.

2.2 Digital Representation of Sound

Sound is captured by an electroacoustic transducer such as a microphone as an analog (i.e. continuous) signal $x(t)$. To analyze this sound digitally, $x(t)$ must be converted to a digital (discrete) signal $x[n]$ which has finite intervals between measurements, a finite bandwidth and a finite range of values. This is done via an Analog-to-digital converter (ADC) and occurs in three steps.

First, the bandwidth of $x(t)$ is made limited via a low-pass filter, such that it falls between 0 and a cut-off frequency B. Second, the filtered signal is sampled with a sample frequency $f_s >= 2B$, as defined by the Nyquist theorem, to avoid aliasing. Finally, the obtained digital signal $x[n]$ is quantized, such that its amplitude can take on a finite number of values. Sampling an audio signal with a bandwidth which covers the audible spectrum (20-20,000Hz) thus requires an f_s of at least 40,000 Hz. Typical sampling rates for audio signals are 44,100 Hz or 48,000 Hz, with 16-bit quantization (i.e. the amplitude has 65,536 possible values) [20].

2.2.1 Frequency Domain Representation

A time-domain signal $x[n]$ can be converted to a frequency-domain signal $X[k]$ via the Discrete Fourier Transform (DFT):

$$X[k] = \sum_{n=0}^{N-1} x[n] \mathrm{e}^{-\frac{j2\pi}{N}kn}. \tag{2.5}$$

The frequency bins in $X[k]$ are linearly separated, with its frequency resolution determined by the sample frequency f_s divided by the number of samples N used for the DFT. For a real-valued time signal such as an audio signal, only the first $\frac{N}{2}+1$ frequency bins of the DFT are of interest; the others are redundant complex conjugates. The DFT is invertible, and as such a reconstruction of $x[n]$ can be obtained from $X[k]$ via the Inverse Discrete Fourier Transform (I-DFT):

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \mathrm{e}^{\frac{j2\pi}{N}kn}. \tag{2.6}$$

In practice, the (I-)DFT is implemented via a Fast Fourier Transform (FFT) which essentially de-composes it into smaller (I-)DFTs, greatly reducing computational time. A typical FFT algorithm (e.g. radix-2) requires 2^k samples, with k a strictly positive integer, otherwise zero padding is applied which can result in spectral leakage [20].

2.2.2 Time-Frequency Representation

The DFT as described above captures global properties of the signal, and preserves no temporal information. For the localization of audio events within the signal, a time-frequency representation which divides it into M frames each containing K frequency bins may provide more information. A single frame typically covers 20-40 ms [21, 22].

Short-Time Fourier Transform

The time-frequency signal $X[k,m]$ is obtained by applying the (discrete) Short-time Fourier Transform (STFT) to $x[n]$, which is essentially a DFT applied to N samples of $x[n]$, multiplied by a windowing function $w[n]$:

$$X[k,m] = \sum_{n=0}^{N-1} x[n]w[n]\mathrm{e}^{-\frac{j2\pi}{N}kn}. \tag{2.7}$$

The windowing function in Equation (2.7) is usually a raised cosine variant such as the Hann or Hamming window, since they prevent discontinuity between STFT frames [23]. The amount of samples between successive windows (hop size) determines the frame length of the STFT. The hop size is usually less than N, resulting in overlapping samples between windows, which account for the decreased weight of the samples near the border of the window. Hop size is generally dependent on the windowing function; a narrow window requires larger overlap (i.e. a smaller hop size) and vice versa [23].

$X[k,m]$ is often represented visually in the form a spectrogram, which is defined as the squared magnitude of the STFT of $x[n]$:

$$\text{Spectrogram}[k,m] = |\text{STFT}(x[n])|^2, \tag{2.8}$$

with the time/sample index m indicated on the horizontal axis, frequency k indicated on the vertical axis, and amplitude or power denoted by intensity. Although this transformation discards the phase information and is therefore not invertible, the original signal can be approximated via e.g. the Griffin-Lim algorithm [24]

Constant-Q Transform

The Constant-Q Transform (CQT), as introduced by [25], is closely related to the STFT, but instead uses logarithmically spaced frequency bins when transforming the signal $x[n]$ to $X[m,k]$:

$$X[k,m] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} x[n]w[n,k]e^{-\frac{j2\pi Qn}{N[k]}} \tag{2.9}$$

Here, Q is the (constant) quality factor, defined as the k-th center frequency divided by the k-th filter width. Because of the varying filter width, the window size $N[k]$ also varies for each filter. As a result of the logarithmic spacing, the CQT has a higher frequency resolution at low frequencies than an STFT with an identical number of bins. The idea of the CQT stems from the Western musical scale, which uses the same logarithmic spacing [25]. As such, it is widely used in Music Information Retrieval (MIR) and SEC tasks [26, 27].

Mel Scale

The human ear does not perceive sound linearly with frequency over the audible spectrum, but finds equally-spaced low-frequency sounds easier to differentiate than high-frequency sounds. When a detection problem involves sounds that are easily recognized by humans, such as aircraft noise, a detector may benefit from a frequency scaling more suitable to the human ear [28]. The Mel scale [29] scales the conventional spectrum such that each pitch is perceived equidistant from its neighbors:

$$f_{mel} = 2595\log_{10}\left(1 + \frac{f}{700}\right). \tag{2.10}$$

Figure 2.1 displays several spectra of an aircraft fly-over. All spectra have been logarithmically scaled, standardized and scaled to a 0-1 interval. The top-left image displays the spectrogram (obtained from the STFT) an a linear frequency scale. On the bottom-left, the same spectrum is displayed on a logarithmic scale. The

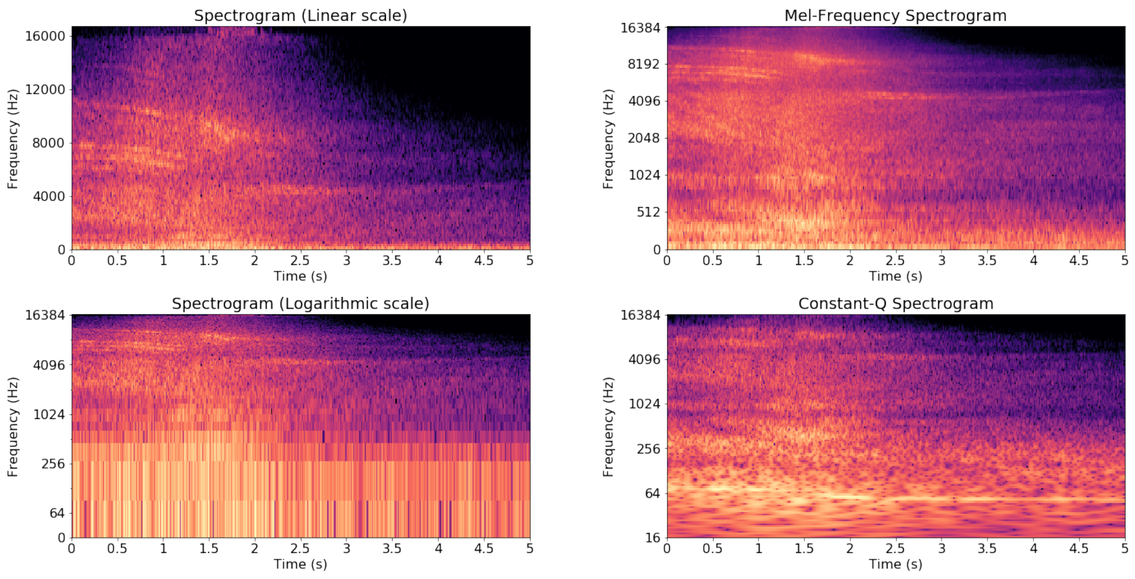

Figure 2.1: Various spectrograms of an aircraft fly-over. Aircraft audio obtained from the ESC-50 dataset [1].

top and bottom right image show the Mel-frequency spectrogram and constant-Q spectrogram, respectively. This figure illustrates the inherent advantage of (pseudo-)logarithmically scaled frequency bins for SEC: the lower frequency components, where most of the characteristics of the signal appear to be contained, are sampled at a higher resolution so that more information is captured.

2.3 Feature Extraction

Although any of the above representations for the audio signal can be used as direct input into the aircraft detector, they may not always serve as a good representation of the data. Instead, features can be extracted, reducing the dimensionality and redundancy of the input while still retaining useful information. Feature selection depends heavily on the field and the problem at hand. Feature engineering involves the extraction of features based on field-specific knowledge. Alternatively, feature learning allows the detector to learn which type of feature is best suitable for the problem. This section provides an overview of features used frequently in audio processing tasks.

2.3.1 Time-Domain Features

Time-domain features work directly on the raw audio signal in the time domain, without directly using any spectral properties of the signal.

Short-Time Energy

The Short-Time Energy (STE) is a simple feature which describes the amplitude variations of a signal over time, by dividing it into frames via a fixed window w of size N, and computing the total energy within each frame:

$$E[m] = \frac{1}{N}\sum_{n}^{N}(x[n]\,w[m-n])^2. \tag{2.11}$$

Zero-Crossing Rate

The (short-time) Zero-Crossing Rate (ZCR) is the number of times the amplitude of the time signal changes sign (i.e. crosses zero) within a fixed time frame. The ZCR of a signal frame can thus be considered a rough estimate of its frequency.

Due to their simplicity, applications using solely the STE and ZCR of a signal are rather limited. [30] combine the STE and ZCR to detect the presence of speech within a frame. A frame with high STE and low ZCR is assumed to contain speech, while a frame with low STE and high ZCR is assumed to contain predominantly noise. [31] use the ZCR in combination with the frame root-means-square (RMS) to discriminate between music, speech and silence.

2.3.2 Frequency-Domain Features

Frequency-based features are extracted from the frames obtained by transforming the time signal with an STFT. They may also use Mel or constant Q frequency scaling and/or logarithmic amplitude scaling instead of linear scaling.

Band Energy

The band energy is the total energy contained in a given frequency band k of a frame m in the time-frequency representation of a signal $X[k,m]$, defined as the square of its magnitude:

$$E[k,m] = |X[k,m]|^2. \tag{2.12}$$

A powerful feature related to the band energy is the log Mel-band energy, which uses Mel frequency scaling and logarithmic amplitude scaling. It is one of the few purely spectral features suitable for more complex tasks such as polyphonic SED [32] and multi-channel SED [33]. Another feature related to the band energy is the band energy ratio, defined as the band energy divided by the total energy in the frame [34].

Spectral Flux

The spectral flux measures the spectral change between successive frames. It is defined as the square of the difference between the normalized magnitude $\hat{X}$ of two frames:

$$F[m] = \sum_{k=1}^{N} \left(\hat{X}[k,m] - \hat{X}[k,m-1]\right)^2 \tag{2.13}$$

As a stand-alone feature, [35] use the Mel spectral flux of a signal for speech activity detection.

Spectral Roll-off

The spectral roll-off of a frame indicates the skewness of its spectral shape, and is defined as the frequency below which a predefined amount (usually 85% to 99% of the total energy) of spectral energy is contained [2].

As most of these individual spectral features are simplistic, they are usually combined into a single feature vector which provides more information [34, 36, 37]. This combination of time- and frequency domain features is often referred to as the low-level feature vector of a signal.

2.3.3 Cepstral Features

Cepstral features are features which operate in the cepstral domain, which is related to the frequency spectrum of a signal. The cepstrum $X_c[n]$ of the spectrum $X[k]$ is obtained by applying the inverse DFT to the logarithm of its power spectrum:

$$X_c[n] = \text{I-DFT}(\log(|X[k]|^2)). \tag{2.14}$$

According to the source-filter model of speech generation [38], a speech signal is generated in the vocal chords, and then convoluted in time via the vocal tract and tongue which act as linear filters. The advantage of the cepstral domain is that it allows for separation of the source and filter(s), since a convolution in time is represented as an addition in the cepstral domain (due to the log-operation on the spectrum). As such, cepstral features have been widely used in Automatic Speech Recognition (ASR) and speech synthesis tasks [39].

Mel Frequency Cepstrum Coefficients

The Mel Frequency Cepstrum (MFC), as introduced by [40], represents the cepstrum of a signal on the Mel scale. The coefficients which make up the MFC are defined as the Mel-Frequency Cepstral Coefficients (MFCCs). They have been widely used in speech processing applications as well as general sound scene analysis over the past decades, and were in the past regarded as the baseline when evaluating the performance of a new acoustic feature [26].

The MFCCs can be obtained from a time signal $x[n]$ as follows [41]. First, the signal is divided in overlapping frames of which the power spectrum estimate is obtained. This is equivalent to the squared magnitude of the STFT of $x[n]$ divided by the number of samples per frame. A Mel-scaled triangular filterbank with 50% overlap is then applied to the power spectrum. Finally, the discrete cosine transform (DCT) is applied to the logarithm of the filterbank coefficients to obtain the (real-valued) MFCCs. The DCT expresses a signal as a sum of cosines:

$$x[n] = \sum_{k=0}^{K-1} X[k] \cos\left(\frac{(2k+1)n}{2K}\pi\right). \tag{2.15}$$

In essence, the MFCCs look for periodicity within the log-spectrum. The application of the DCT instead of the I-DFT has the advantage of decorrelating the filterbank coefficients, which became correlated due to the overlapping STFT windows. The MFCCs are also real-valued coefficients due to the DCT.

For a signal with a bandwidth of 16 kHz, the number of filters in the Mel filterbank usually ranges from 12 to 30 [2]. The MFCC feature vector may also be truncated by discarding the first coefficient, which represents the signal energy [42], and/or by discarding high coefficients that are deemed unnecessary [43]. The MFCC feature vector can also be augmented with the delta-MFCCs and delta-delta-MFCCs, defined as respectively the first- and second derivate of the (truncated) MFCCs. These delta-coefficients model the dynamics of the coefficients and increase classification accuracy [42]. [36] compared MFCCs with a low-level audio feature vector, and found that the former had a higher classification accuracy for audio classification as well as music classification tasks.

2.3.4 Spectrogram Image Features

A spectrogram image feature (SIF) treats the recorded sound as a time-frequency structure, rather than a time series of frequency bins. Inspired by computer vision techniques, these features intend to obtain characteristics regarding the shape and evolution of the time-frequency contents of the signal [2]. To obtain a grey-scale spectrogram image from a spectrogram, the spectrum energies must first be mapped to a 0-1 scale. [44] add an additional step which quantizes the grey-scale image into several monochrome channels via pseudo-colormapping, and found it improves the classification accuracy. They also observe that SIFs vastly outperform MFCCs under noisy conditions, because the presence of noise causes mismatches, and thus inaccuracies, in the MFC coefficients. An alternative approach by [45] is to extract the region surrounding the three maximum energy peaks of a smoothed, de-noised spectrogram image. They report that it outperforms the SIF introduced by [44] under clean and noisy conditions.

Histogram of Orientated Gradients

The Histogram of Orientated Gradients (HOG), as introduced by [46], is a feature used in computer vision applications for object detection in images. It uses the intensity and orientation of image gradients to provide local shape information, and bins them into a histogram. In audio scene classification, the objective of the HOG is to extract the shape of certain time-frequency structures which may be characteristic for an audio scene [26]. [26] note that the HOG has several advantages over power spectrum-based features such as MFCCs. Not only are they invariant to small time- and frequency translations, they also provide information regarding the local direction of variation of the power spectrum.

2.3.5 Feature Learning

The learning of features has become a popular and viable alternative to selecting manually engineered features for sound analysis in recent years [2]. This section lists some notable feature learning algorithms. Artificial neural networks, of which the hidden layers are essentially feature learners, are discussed in Section 3.4.

Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF) is a feature learning technique popularized by [47], where it is used to learn a part-based representation of human faces in grey-scale images. Given a set of N images which each contain M pixels, the NMF decomposes the NxM data matrix **V** into two non-negative matrices **W** and **H**, with $\mathbf{W} \in \mathbb{R}^{NxK}$ and $\mathbf{H} \in \mathbb{R}^{KxM}$. The basis images are contained in the K columns of **W**, while the image encoding is present in the M columns of **H**. Due to the nonnegativity constraint, each human face is thus approximated as a linear summation of all basis images, according to the encoding in the columns of **H**.

Naturally, the NMF can also be applied to time-frequency images obtained from audio signals, such as the spectrogram image. [48] introduce the convolutive NMF, which accounts for the temporal relationship that is often found between nearby time-frequency frames. This temporal dependency is modelled by making the basis images and encoding time-variant. [49] have found that the convolutive NMF outperformed the NMF for a multi-class acoustic scene classification task.

Chapter 3

Sound Event Detection for Aircraft

Sound Event Detection (SED) is the temporal localization of an event (e.g. an aircraft fly-over) within a recording based on its acoustic signature. This is similar to Sound Event Classification (SEC), which yields a class from the predicted class probabilities of a given audio clip. A common method is to consider a sound 'detected' in a recording, when the classification of several subsequent clips of the recording has yielded its class.

This chapter explains the preliminaries of classification theory, then elaborates on commonly used classifiers and their application in the SED field. Section 3.1 provides a brief theoretical background for classification, applied to aircraft detection. Section 3.2 and Section 3.3 list common examples of generative and discriminative classifiers used for SED, respectively. Section 3.4 provides a detailed introduction of artificial neural networks (ANNs), which extract features from their hidden layers. Finally, Section 3.5 provides an overview of state-of-the-art classifiers for SED.

3.1 Supervised Classification in SED

A supervised classifier maps each input x_i to a class label y_j by means of a predictor $h(x_i|\boldsymbol{\theta})$, defined by the set of parameters $\boldsymbol{\theta}$. The objective of the classifier is to find the $\boldsymbol{\theta}$ that minimizes its error rate, which is approximated by a loss function L:

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} L(h(x_i|\theta), y_i). \tag{3.1}$$

When L is non-differentiable (such as with the 0-1 loss, where L is 0 if the label is predicted correctly, 1 otherwise), it must be replaced with a differentiable surrogate function $f(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta})$. This can be coupled with a regularization term $g(\boldsymbol{\theta})$, which adds a penalty for undesirable configurations such as high parameter weights or high deviation from a given prior distribution. The new loss function to be minimized is then:

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} f(x_i, y_i|\boldsymbol{\theta}) + g(\boldsymbol{\theta}). \tag{3.2}$$

The available dataset containing sample pairs of ($\boldsymbol{x}$, $\boldsymbol{y}$) is split into a mutually exclusive training set and test set, where the former is used to optimize θ, and the latter to evaluate the resulting $\boldsymbol{\theta}$. The training set is commonly further split up in an actual training set and a validation set. The validation set is evaluated during training, with the purpose to identify and prevent over-fitting during optimization of the model.

3.1.1 Classification for Aircraft Detection

For the aircraft detector, each evaluation sample must be classified as aircraft/non-aircraft. There are two approaches towards building such a classification model: binary classification or one-class classification.

A binary classifier is discriminatory, i.e. it learns to discriminate between classes based on the data available for both classes. Classification for classes y_1 and y_2 with prior probabilities $P(y_1)$ and $P(y_2)$ is done as follows:

$$\text{Classification}(x_i) = \begin{cases} y_1, & \text{if } p\left(x_i|y_1\right)P(y_1) > p\left(x|y_2\right)P(y_2) \\ y_2, & \text{otherwise} \end{cases} \tag{3.3}$$

When there is only sparse availability of class y_2, i.e. P(y_1) >> P(y_2), this will almost always result in p($x_i|y_1$) >> p($x_i|y_2$), leading to a biased classifier. A one-class classifier only learns the representation of a single class y, with the aim of recognizing instances of the target class based on a threshold α:

$$\text{Classification}(x_i) = \begin{cases} y, & \text{if } p\left(x|y\right) \geq \alpha \\ none, & \text{otherwise} \end{cases} \tag{3.4}$$

Since it does not rely on prior probabilities, a one-class classifier will not be biased when imbalance is present in the evaluation data. Such an approach is especially useful when only one of the classes in a selection of objects can be described with precision [50]. [51] observed that the performance of binary classifiers starts to deteriorate at an imbalance ratio of 1:2.5, although they denote that this ratio is dependent on the data and the classification problem, and not set in stone.

3.1.2 Classifier Evaluation

There are several metrics to evaluate a binary classifier, which are all applicable to a one-class classifier as well. The most obvious one would be the accuracy, which sums the amount of correctly classified samples over the total samples:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

Unfortunately, accuracy alone fails to account for class imbalances, and is also not an adequate metric when false positives (FP) are far less desirable than false negatives (FN) or vice versa. For this reason, classifier evaluation using precision, recall, and the F_1-score is usually preferred. Precision is the ratio of true positives (TP) over all predicted positives:

$$Precision = \frac{TP}{TP + FP}. \tag{3.6}$$

Recall is the ratio of true positives over all actual positives:

$$Recall = \frac{TP}{TP + FN}. \tag{3.7}$$

The F_1-score is the harmonic average of precision and recall:

$$F_1 = 2\frac{Precision * Recall}{Precision + Recall}. \tag{3.8}$$

Naturally, all the above metrics lie between 0 and 1, with a score of 1 indicating perfect performance. Performance is often visualized using the Receiver Operator Characteristic (ROC) curve, which plots the True Positive Rate (TPR or recall) against the False Positive Rate (FPR). The score of each ROC curve is given by the Area Under the Curve (AUC) which measures model separability, i.e. the probability of distinguishing between aircraft and non-aircraft.

While binary- and one-class classifiers can be distinguished based on their usage of training data, another distinction can be made as to how $p(y_j|x_i)$ is estimated. A generative classifier attempts to model the joint probability distribution $p(x_i, y_j)$ per class for each input, and then find $p(y_j|x_i)$ using Bayes' rule:

$$p(y_j|x_i) = \frac{p(x_i|y_j)P(y_j)}{P(x_i)} \tag{3.9}$$

On the contrary, a discriminatory classifier approximatees $p(y_j|x_i)$ by learning the differences between classes.

3.2 Generative Classifiers

Since the goal of a generative classifier is to model the feature space, the loss is related to the dissimilarity of the model distribution and the (unknown) data generating distribution. Without the inclusion of a prior, this is equivalent to Maximum Likelihood Estimation (MLE), i.e. maximizing the probability that the training data is generated by $P_{\boldsymbol{\theta}}$, the probability density w.r.t. $\boldsymbol{\theta}$. The objective function $f(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta})$ is then the negative log-likelihood function:

$$f(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}) = -\log P_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}). \tag{3.10}$$

3.2.1 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is an unsupervised generative classifier which assumes that the distribution which generates the training data can be approximated by a weighted mixture of K Gaussians, with probability density function

$$P_{\boldsymbol{\theta}}(x_i) = \sum_{k=1}^{K} \phi_k \frac{1}{\sqrt{2\pi|\Sigma_k|}} e^{-\frac{1}{2}(x_i-\mu_k)^T \Sigma_k^{-1}(x_i-\mu_k)}, \tag{3.11}$$

where the parameters $\boldsymbol{\theta}$ to be learned consist of the mixture coefficient ϕ_k (with each $\phi_k \geq 0$ and $\sum_{k=1}^{K} \phi_k = 1$), mean μ_k and positive definite covariance matrix Σ_k of each Gaussian. To reduce the amount of parameters to be learned, each Σ_k can be restricted to be diagonal, spherical ($\Sigma_k = \sigma_k I$) or even isotropic ($\Sigma_k = I$). The choice for K must be fixed before optimizing for θ, but is usually not known a priori. For classification, it is often treated as a hyper-parameter and optimized using cross-validation.

The GMM classifier introduced in Equation (3.11) does not rely on the labels ($\boldsymbol{y}$), but can be made supervised by fitting a separate GMM ($P_\theta(x|y)$) for each class. The predictor is then defined as

$$h(\boldsymbol{x}|\boldsymbol{\theta}) = \underset{\boldsymbol{y}}{\arg\max}\, P_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) = \underset{\boldsymbol{y}}{\arg\max}\, P_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}) P_{\boldsymbol{\theta}}(\boldsymbol{y}). \tag{3.12}$$

GMMs have often been combined with MFCC features, because the linear GMM benefits from the decorrelated coefficients. [28] used a feature vector consisting of the first 13 MFC coefficients in combination with a GMM classifier, trained with an OCC approach. Similar sounding objects, such as high-speed trains, were included in their evaluation set. They observed an accuracy of 95%, which deteriorated to 60% as SNR decreased.

Figure 3.1 displays the covariance matrices, with the diagonal restriction, of MFC coefficient 1 and 2 of a GMM trained on aircraft and non-aircraft data. The GMM was made supervised by assigning one Gaussian to each class. While 40 MFCCs were obtained from each audio recording, only the first 13 coefficients were used for the classification, in accordance with [28].

3.2.2 Hidden Markov Model

A limitation of the previously discussed GMMs is that they are not able to model the (temporal) dynamics of sound. For a general sequence of observations, the current observation at time t would depend on all prior observations. A Markov process is a sequence in which the current state $x[t]$ is dependent only on the previous state $x[t-1]$, and an initial state $x[0]$. The idea behind it is that the current observation $x[t]$ adequately summarizes all previous information. A hidden Markov model (HMM) is a Markov process where the (discrete) states which govern the dynamics are not directly observable, and the outcome $x[t]$ is only influenced by the current hidden state $z[t]$. An HMM is then defined as

$$P_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) = \prod_{t=1}^{T} P_{\boldsymbol{\theta}}(z[t]|z[t-1]) P_{\boldsymbol{\theta}}(x[t]|z[t]), \tag{3.13}$$

and consists of three main components. The initial state model $P_{\boldsymbol{\theta}}(z[1]|z[0])$ defines the probability of starting the sequence in each state. The transition model $P(z[t]|z[t+1])$ governs the dynamics of the hidden states. Assuming $z[t]$ can take on K different values, the transition model is defined by a KxK transition matrix $\boldsymbol{V}$. Finally, the emission model $P_{\boldsymbol{\theta}}(x[t]|z[t])$ governs how each observation is generated from each hidden state.

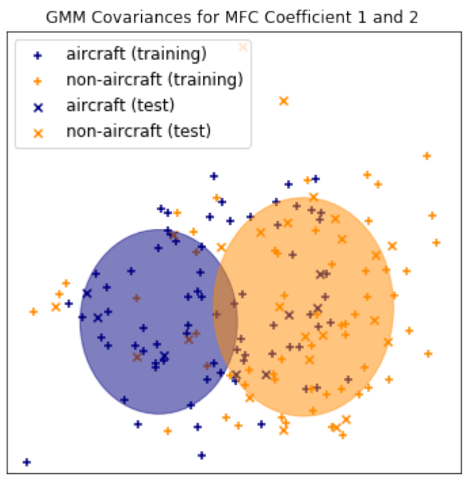

Figure 3.1: Covariance matrices belonging to MFC coefficient 1 and 2 of a GMM trained on aircraft and non-aircraft samples of the ESC-50 dataset [1].

3.3 Discriminative Classifiers

A discriminative classifier aims to predict label y for input x, but does not model the input space and joint distribution.

3.3.1 Support Vector Machine

The support vector machine (SVM) introduced by [52] is a linear, discriminative classifier mainly used for binary classification. Given a set of features $\boldsymbol{x}$ with corresponding class labels $\boldsymbol{y}$ indicated as +1 or -1, the objective is to construct a hyperplane which separates the inputs from each class with maximum margin. With a hyperplane defined as $\boldsymbol{W} \cdot \boldsymbol{x} - \boldsymbol{b} = 0$, the SVM operates on the margin hinge loss (a surrogate function for the 0-1 loss) plus a weight penalty governed by λ:

$$\min_{\boldsymbol{W},\boldsymbol{b}} \frac{\lambda}{2}||\boldsymbol{W}||^2 + \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{W} \cdot x_i - \boldsymbol{b})). \tag{3.14}$$

In Equation (3.14), the loss of an input-output pair is 0 if the input is on the correct side of the hyperplane, and proportional to the distance from the margin otherwise. The λ term balances accuracy and complexity.

The SVM as introduced above is limited to solving linear problems. To provide a non-linear solution, a kernel function $\phi(x_i, x_j)$ is required which maps the input data to a higher-dimensional space such that a hyperplane can be fitted through the new representation [53]. Examples of common kernels are the Gaussian kernel, polynomial kernel and sigmoid.

3.3.2 Logistic Regression

Logistic regression [54] is an alternative to the SVM for binary classification problems. It has the same parameterization as the SVM, but maps the scores obtained from the hyperplane defined by $\boldsymbol{W} \cdot x_i - \boldsymbol{b}$ to a 0-1 scale via the logistic sigmoid function. Instead of the hinge-loss, logistic regression uses the negative log-likelihood as a surrogate function for the 0-1 loss:

$$\min_{w,b} \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1-y_i}{2}\right)\left(w \cdot x_i - b + \log\left(1 + \mathrm{e}^{w \cdot x_i - b)}\right)\right). \tag{3.15}$$

Unlike the scores of the SVM, the scores of the logistic regressor contain a probabilistic interpretation, which may prove useful for confidence-related predictions or integration with larger models such as the transition model of an HMM [2]. Kernel methods can be applied to logistic regression to solve non-linear problems in an identical manner as the SVM.

3.4 Artificial Neural Networks

The artificial neural network (ANN) is a family of models inspired by, but not modelled after, the biological brain. A deep neural network (DNN) consists of an input layer, several hidden layers (artificial neural networks with none or few hidden layers are considered shallow), and an output layer. The trainable parameters of the network are the nodes ('neurons') of each layer, where the nodes of a layer are obtained from those of the previous layer via a non-linear mapping (activation function). For a DNN classifier, the final layer of the network is typically a linear discriminative classifier such as a logistic regression model or SVM. In essence, the purpose of the hidden layers is to extract the appropriate features from the input layer.

The cost function used for a DNN classifier is usually the negative log-likelihood function, also known as the cross-entropy cost. Due to the non-linear interaction between layers, this term does not have a closed form, and the network must instead be trained using an iterative approach such as stochastic gradient descent (SGD), where θ moves in the direction of the steepest negative gradient:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \left(\frac{1}{M'} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{M'} -\log p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) \right). \tag{3.16}$$

The gradient obtained from SGD serves as an estimate of the actual gradient, by randomly sampling a mini-batch of fixed size M' from the available training set of size M to obtain the next update for $\boldsymbol{\theta}$. The SGD algorithm terminates after a given amount of epochs, where one epoch indicates a complete pass through all samples in the training set. The advantage of SGD over batch gradient descent is that the computational time of the gradient is independent of the size of the training set.

The learning rate ϵ determines the sensitivity of $\boldsymbol{\theta}$ to a new update. Too large of a learning rate may increase the training error between updates, or even lead to a positive feedback loop which eventually results in numerical overflow. On the other hand, too small of a learning rate leads to slow training progress which may result in getting stuck at high training errors. While ϵ can be kept constant, a common choice is to reduce it linearly for the first few hundred epochs, and then keep it constant [3]. Tuning of the learning rate and other hyperparameters can be done via a random search, i.e. evaluating performance on the validation set for several different learning rates.

Several notable variations or improvements of SGD exist. SGD-momentum accelerates the learning of SGD by updating $\boldsymbol{\theta}$ according to a 'velocity', calculated from an exponentially decaying moving average of past gradients. The rate of decay of the influence of previous gradients is governed by another hyperparameter α, such that α and the velocity determine the momentum of $\boldsymbol{\theta}$ in parameter space. RMSProp [55] instead adapts the learning rates of the individual parameter weights, inversely proportional with the square root of the sum of an exponentionally decaying moving average of its squared values. As such, parameter weights with a large partial derivative get a larger decrease in their learning rate. Adam [56] is essentially a combination of RMSProp and momentum with added bias correction. Both RMSProp and Adam are considered the go-to for deep learning algorithms [3].

When the model parameters vastly outnumber the available training data, overfitting can occur. Bagging can prevent overfitting by creating several training subsets, sampled with replacement from the original training set, and training a separate model on each subset. The final model (ensemble) is then obtained by averaging all trained models. A less computationally expensive alternative to bagging is dropout [57]. Dropout applies a random binary mask over the non-output neurons in the network for each training example. This prevents neurons from co-adapting to the training data, thus making it more robust to overfitting [57]. The probability of dropping a neuron is regulated via a dropout term. Common values are 0.2 for input units and 0.5 for hidden units [3].

3.4.1 Multi-Layer Perceptron

A multi-layer perceptron (MLP) is a fully-connected network, i.e. every node in layer k is connected to every node in layer $k+1$. The information provided by the input layer simply propagates forward, through the hidden units, to the output layer, without any feedback connections. The layer-to-layer mapping of each unit is done via a non-linear transfer function ρ applied to an affine transformation of the unit in the previous layer:

$$z_i^{(k+1)} = \rho\left(\sum_{j=1}^{M_k} W_{i,j}^T z_i^k + b_i\right) \tag{3.17}$$

The weights $W_{i,j}$ and biases b_i in Equation (3.17) define the affine transformation of each unit and are usually referred to as weights for brevity. The parameter vector $\boldsymbol{\theta}$ thus consists of the weights of the units in each layer. Figure 3.2 illustrates the structure of a small MLP with two hidden layers.

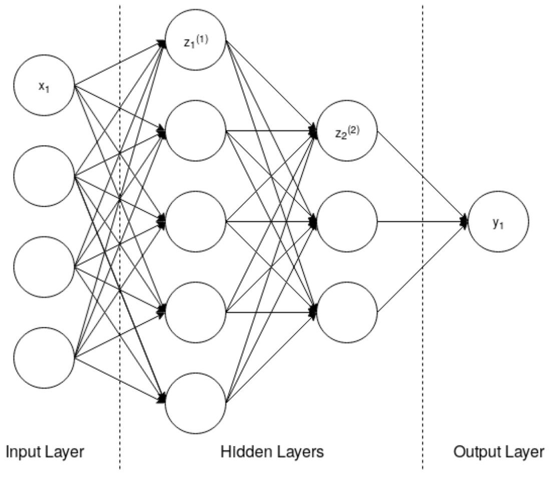

Figure 3.2: Schematic of a simple multilayer perceptron with 4 input units, 2 hidden layers and 1 output unit.

The preferred activation function for a feedforward network such as an MLP is the Rectified Linear Unit (ReLU) or one of its variants. Usage of sigmoidal activation functions such as the logistic sigmoid (σ) or the hyperbolic tangent (tanh) in a feedforward network is discouraged, since they have a large saturating region where the function is insensitive to changes in the gradient [3].

Computation of the gradient of the cost function of a feedforward network is done via the back-propagation algorithm. Starting at the output layer, it recursively applies the chain rule of calculus to compute the total gradient from the sub-gradients between layers. To avoid exploding gradients when initiating the algorithm, weights must be intialized to a small, non-zero value or random value.

3.4.2 Convolutional Neural Networks

A convolutional neural network (CNN), introduced by [58], is a feedforward network which uses the convolution operation, instead of the matrix multiplication used by MLPs, for at least one of its layers. A convolutional layer learns a set of filters, each comprised of one or more convolutional kernels. In audio processing, both 1D and 2D convolutions can be applied to spectrogram image representations. Although 2D convolution requires log-scaling between frequency bins in order to cover a constant frequency ratio per filter [2], research indicates that linear scaling remains feasible in practice [59]. An advantage of 2D convolution over 1D is that it enables the modelling of events which move in frequency. The first layers of a 2D CNN then learn local, transient tones, which can then be integrated over the frequency range by subsequent layers [2].

Due to the sharing of parameters, a convolutional layer is equivariant to translation; when a time-frequency structure in the input is shifted in time (and/or frequency, in case of 2D convolution), the output of the layer will reflect this. After applying the activation function to the input convoluted with each filter, a set of feature maps is obtained. A convolutional layer may also be followed by a pooling layer, which reduces the dimensionality of the layer. Such a pooling function outputs a single summary statistic (e.g. the maximum or average) of the region it is applied to. Max pooling also introduces invariance to small translations in the input across the pooling dimension [3].

Figure 3.3 and Figure 3.4 illustrate a one- and two-dimensional CNN respectively (indicated by the shaded receptive field of the filters) with a 94x252 spectrogram image as input. All filters use 'valid' convolution, i.e. the convolution is not padded with zeroes.

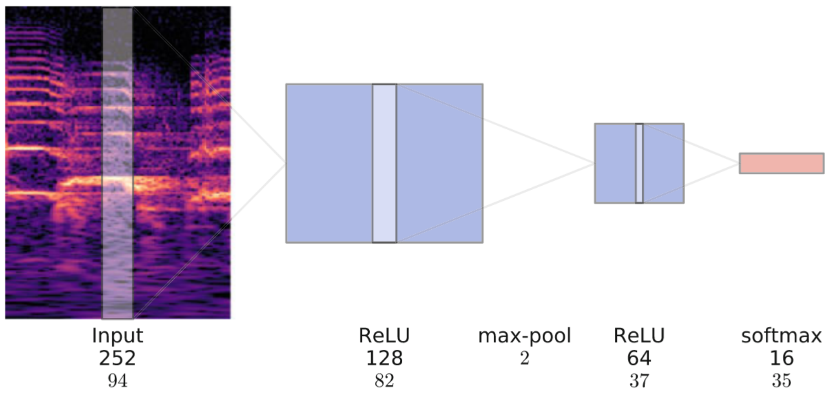

Figure 3.3: Schematic of a shallow 1-D CNN with convolution in the time dimension [2].

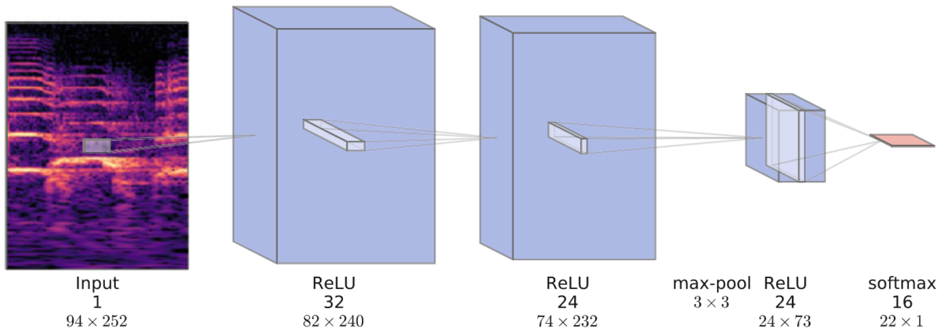

Figure 3.4: Schematic of a shallow 2-D CNN with convolution in the time and frequency dimension [2].

[59] note that for tasks which require audio recognition in the presence of noise, images perform best as input features. [45] also denote that CNNs fed with spectrogram images significantly outperform all other feedforward architecture Furthermore, they deduce that neural networks have a trade-off between noise-free and noisy performance. The most common input feature to a CNN is the log-scaled Mel-spectrogram image or a related variant. [60] observed that raw waveforms did not reach the performance of spectrogram images.

3.4.3 Recurrent Neural Networks

Unlike MLPs and CNNs, a Recurrent Neural Network (RNN) is able to model long-term temporal dependencies found in sequential data by sharing parameters across several time steps. A recurrent layer encodes temporal dependencies via hidden units $h[t]$, which depend on the layer input $x[t]$ and the hidden units at the previous time step $h[t-1]$:

$$h[t] = \rho(\boldsymbol{U}^T x[t] + \boldsymbol{W}^T h[t-1] + \boldsymbol{b}), \tag{3.18}$$

where $\boldsymbol{U}$ denote the input-to-hidden weights, $\boldsymbol{W}$ denote the hidden-to-hidden weights and $\boldsymbol{b}$ is a bias term. The activation function ρ is commonly the tanh function [3]. The output $o[t]$ of the recurrent layer is derived from the hidden units $h[t]$ via a matrix transformation with hidden-to-output weights $\boldsymbol{V}$ and bias $\boldsymbol{c}$:

$$o[t] = \boldsymbol{V}^T h[t] + \boldsymbol{c}. \tag{3.19}$$

The gradient of an RNN is computed with the back-propagation through time (BPTT) algorithm. BPTT is essentially standard back-propagation, with the additional constraint that the gradient flows backwards in time until the starting point of the sequence is reached. A drawback of BPTT is that is considerably slower than back-propagation, because of the sequential nature of RNNs. Another disadvantage intrinsic to RNNs is that they are sensitive to vanishing or exploding gradients, since the gradient is multiplied by the hidden-to-hidden weight matrix $\boldsymbol{W}$ every time step. Exploding gradients make learning unstable, but can be mitigated somewhat by gradient clipping. Vanishing gradients make the direction in which to move in unclear.

Several adaptations to the recurrent layer can be made. Bi-directional RNNs are used for problems where an input $x[t]$ may depend on past and future values. This is encoded via a separate hidden state vector $g[t]$ which propagates information backwards in time similarly to Equation (3.18). The output $o[t]$ of the layer is then the weighted sum of the forwards- and backwards-propagating state vectors. In general audio processing tasks such as speech detection or event recognition, bi-directional RNNs generally outperform their causal counterparts [2]. An example schematic of a bi-directional RNN consisting of a single recurrent layer is illustrated in Figure 3.5

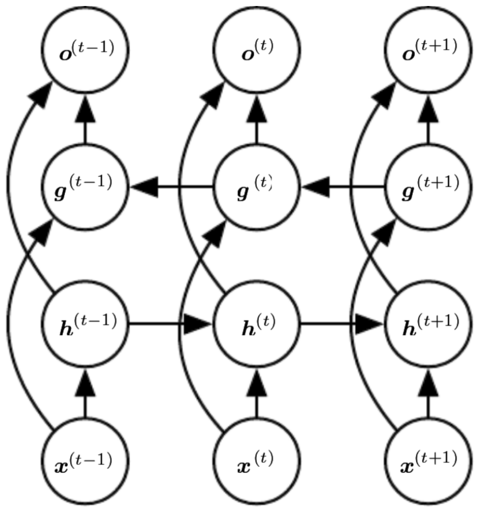

Figure 3.5: Schematic of a bi-directional recurrent layer with hidden units h and g [3].

The tanh activation function in a recurrent layer can also be replaced by Long Short-Term Memory (LSTM) cells which ensure that the gradient vanishes nor explodes by introducing a self-looping memory unit with gated weights [61]. Because the weight of the self-loop is gated, the time scale of the layer is dynamic, i.e. it may vary per input sequence. Each LSTM cell consists of a memory unit $c[t]$ and hidden state $h[t]$, which are governed by the input gate $i[t]$, forget gate $f[t]$ and output gate $o[t]$:

$$i[t] = \sigma\left(u_i^T x[t] + w_i^T h[t-1] + b_i\right), \tag{3.20}$$

$$f[t] = \sigma\left(u_f^T x[t] + w_f^T h[t-1] + b_f\right), \tag{3.21}$$

$$o[t] = \sigma\left(u_o^T x[t] + w_o^T h[t-1] + b_o\right), \tag{3.22}$$

$$\tilde{c}[t] = \tanh\left(u_c^T x[t] + w_c^T h[t-1] + b_c\right), \tag{3.23}$$

$$c[t] = f[t]c[t-1] + i[t]\tilde{c}[t], \tag{3.24}$$

$$h[t] = \tanh\left(c[t]\right) o[t]. \tag{3.25}$$

The forget gate $f[t]$ controls which information from the previous hidden state $h[t-1]$ is passed to the new cell state $c[t]$. The influence of the proposed new cell state $\tilde{c}[t]$ on $c[t]$ is controlled by the input gate $i[t]$. Finally, the output (i.e. the hidden state) of the LSTM is controlled by the output gate $o[t]$ which selects the parts of the cell state that will be outputted. A schematic of a recurrent layer consisting of LSTM cells is illustrated in Figure 3.6.

[1]Image adapted from `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

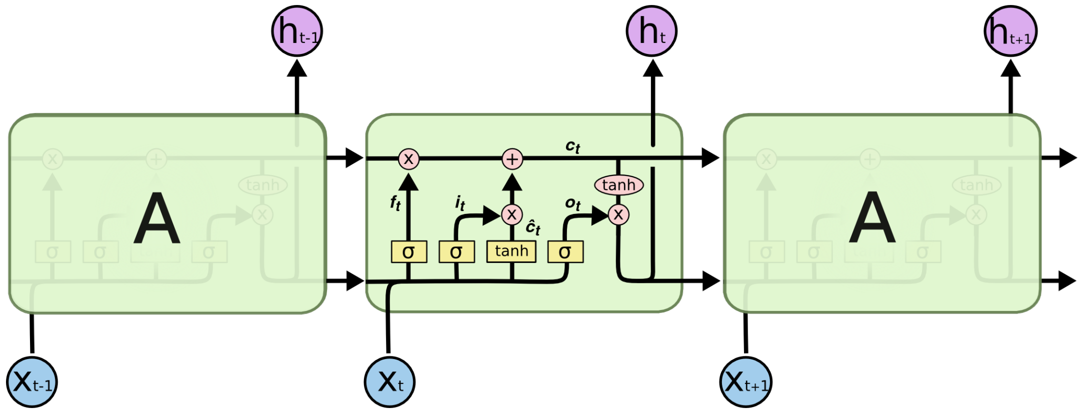

Figure 3.6: Schematic of an LSTM layer.[1]

A bi-directional LSTM (B-LSTM) network generally outperforms any other type of recurrent architecture (including regular LSTMs). [62] report state-of-the-art performance for polyphonic SED using a B-LSTM network with log Mel-band features, although a direct comparison with CNNs is not made. [63] use a Convolutional Recurrent Neural Network (CRNN), which combine the invariance in the frequency domain captured by CNN and the integration of information from previous timestep for the RNN.

3.5 Classifier Comparison

Based on the literature covered in this chapter, it has become evident that discriminative classifiers based on a DNN outperform previously established classifiers such as the GMM and HMM. To distinguish between the performance of the types of DNNs (MLP, CNN, RNN, CRNN) covered in this chapter, Table 3.1 shows an overview of classifier performance and feature selection on audio-related research topics. Most papers use dropout regularization in FC layers, as well as Adam optimization.

[4], [64], [5] and [65] all evaluated on UrbanSound8K [66], a dataset with over 8,000 four-second recordings of distinct environmental sound sources (e.g. dog bark, gun shot, siren) divided into ten classes. Aside from down-sampling and standardisation of the audio signal, [64] did not pre-process the input of their network. Instead, the input was reduced via a large convolutional and max pooling strides. They opted for a very deep, fully convolutional network with global average pooling as they hypothesised that when most of the learning occurs in the convolutional layers, a deep enough convolutional network would require no fully connected layers. While the resulting 18-layer deep network led to satisfactory performance, it did not out-perform the log Mel-SIF features.

[4] and [65] split the non-silent parts of the recordings into segments of approximately 950 ms with 50% overlap between them, with the log Mel-spectrogram as well as their deltas provided as input (2x60x41). On the other hand, [5] randomly extract 128 frames from the log Mel-spectrogram of the first three seconds of each clip. [5] and [65] also augment the data via time stretching, pitch shifting and adding background noise.

These network architectures each consist of two to three convolutional layers followed by two fully connected layers. [4] uses filters wide in frequency, to learn patterns over the whole frequency range. [5] use a small receptive field to learn small, localized patterns which can be fused in subsequent layers to detect larger time-frequency signatures. [65] use dilated kernels, which are padded with zeroes in the center, in combination with Leaky ReLu. This leads to features which are more clearly separable than those of traditional kernels, and as such improves classification accuracy.

The robust sound event classification problem as researched by [59], [45], [67] and [68] is evaluated on the RWCP dataset, which consists of 50 classes with 50 files of training data per class. Noise is added from the NOISEX-92 database, resulting in mixtures with SNRs of 20dB, 10dB and 0dB. While higher performance was achieved on the multi-condition set, only performance on mismatched condition (i.e. train without noise,

Topic	Feature Shape (CxFxT)	Frame (Overlap)	Classifier	Best Performance
ESC [4]	log mel-SIF + deltas (2x60x41)	46 (23) ms	CNN (10)	73.7% acc.
ESC [64]	raw waveform (f_s=8 kHz)	n/a	Fully CNN (10)	71.8 % acc.
ESC [5]	log mel-SIF (1x128x128)	23 (0) ms	CNN (10)	79.0 % acc.
ESC [65]	log mel-SIF + deltas(2x60x-)	-	CNN (10)	81.9 % acc.
Robust SED [59]	log mel-SIF (1x24x30)	-	DBN (50)	92.9 % mean acc.
Robust SED [45]	energy-based SIF (1x40x52)	64 (60) ms	CNN (50)	94.0 % mean acc.
Robust SED [67]	energy-based SIF (1x52x-)	100 (90) ms	CNN (50)	98.6 % mean acc.
Robust SED [68]	quantized SIF (3x513x93)	64 (63) ms	CNN (50)	98.6 % mean acc.
Polyphonic SED [32]	log mel-band energy (1x40x1)	50 (25) ms	MLP (61)	61.7 % acc.
Polyphonic SED [62]	log mel-SIF (1x40x[10, 25, 100])	50 (25) ms	RNN (61)	65.5 F1
Polyphonic SED [63]	log mel-SIF (1x40x73)	40 (20) ms	CRNN (61)	68.3 F1
Large-scale SED [70]	log mel-SIF (1x96x64)	25 (15) ms	CNN (3,000)	0.93 AUC

Table 3.1: Overview of classification/detection tasks and the used feature, input shape, type of classifier and accuracy.

evaluate on noise) has been reported in the table, because this is more in line with the aircraft detection scenario.

[59] constructed a 2-layer neural network, with each hidden layer constructed from a pre-trained Restricted Boltzmann Machine (RBM) pair, forming a Deep Belief Network (DBN). Under mismatched conditions, best performance is achieved via energy-scaled voting weights, as high-energy regions offer more discriminative capability than low-energy regions. [45] use highly overlapped STFT windows to capture instantaneous information. From each recording, 18 SIFs are obtained from the 6-frame context of the three frames with the highest time-domain energy. A fairly standard CNN with 2 convolutional layers and 1 fully-connected layers is used. [67] learn sets of filters with varying shape simultaniously. They use 1-max pooling and 1D (temporal) convolution since it allows the detection of a specific feature which may occur at any time in a signal. [68] perform RGB-quantization on their spectrogram image input, with a shape of (3x513x93), where the time-domain length was obtained from downsampling the full-time spectrum. Their network was pre-trained on the ImageNet dataset, consisting of 5 convolutional layers and 2 fully-connected layers.

[32], [62] and [63] evaluated on a database of real-life recordings categorized into 61 different event classes, with overlap between events [69]. Highest performance was achieved by [63] with log Mel-SIF input, using 4 convolutional layers, 3 recurrent layers and 1 fully-connected layer. The convolutional layers serve as feature extractors, the recurrent layers provide context by integrate these features over time, providing context, and the fully-connected layer produces class activity probabilities.

Chapter 4

Ego-Noise Analysis

Ego-sound suppression of an MAV concerns the suppression or filtering of the noise generated by its motors and airframe. Due to the limited dimensions of an MAV, the acoustic sensor(s) must be placed near the noise sources out of necessity. This can lead to a signal-to-noise ratio as low as -20 dB [71], which limits SED applicability severely.

The outline of this chapter is as follows. Section 4.1 provides a brief overview of MAV noise sources. In Section 4.2 and Section 4.3, self-contained and dependent methods for MAV ego-noise suppression are elaborated upon. Finally, Section 4.4 discusses the Denoising Auto-Encoder (DAE) and its purpose in ego-noise suppression.

4.1 MAV Noise Sources

The sound signal $x[n]$ recorded by the microphone is a combination of the target sound $s[n]$ and ego-noise $v[n]$:

$$x[n] = s[n] + v[n] \tag{4.1}$$

The signal $v[n]$ emitted by a multi-rotor UAV can be considered a mixture of narrow-band harmonic noise and broadband noise [72]. The harmonic component consists of the mechanical noise generated by the rotating motors, with its fundamental frequencies proportional to its rotational speed, while the broadband component is comprised of noise generated by the propeller blades cutting through the air [73]. As such, [73] propose the following model for the ego-noise $v[n]$ of a p-rotor MAV:

$$v[n] = \sum_{p=1}^{P} v_p[n] + v_f[n] \tag{4.2}$$

Each motor generates a directional noise source $\boldsymbol{v}_p$, while the superimposition of the noise of all propellers is modelled as a directionless diffuse noise source $\boldsymbol{v}_f$. Noise generated by the rest of the airframe of the MAV is neglected. Naturally, the intensity of the noise rises as the motor rotational speed rises. As a consequence, the noise cannot be assumed to remain stationary during flight.

Ego-noise suppression methods can be divided into self-contained and dependent methods. Self-contained methods use only information available from the microphone(s), and typically require multiple microphones to suppress the noise of a multi-rotor UAV. Dependent methods required information from other sensors, such as the motor speed, but only require a single microphone.

4.2 Self-contained Approaches

Self-contained approaches suppress MAV ego noise by designing a spatial filter $\boldsymbol{w}[l,n]$ which localizes and extracts the target sound $s[n]$ from the set of recorded sounds $\boldsymbol{x}[l,n]$:

$$s[n] = \boldsymbol{w}^{\mathrm{H}}[]l,n]\boldsymbol{x}[]l,n], \tag{4.3}$$

with superscript H denoting the Hermitian transpose [71].

Because the noise of a p-rotor MAV is a mixture of *p+1* components (Equation (4.2)), at least *p+2* microphones are required to adequately suppress the ego-noise [73]. The position of the microphone array relative to the MAV must also remain fixed, to ensure that the acoustic mixture remains stationary.

4.2.1 Beam Forming

In signal processing, a beamformer (BF) is a processor which acts as a spatial filter by outputting a linear combination of the signals obtained from an array of sensors [74]. In the audio processing domain, the sensing object is naturally a microphone which perceives a time series of audio waves. A distinction can be made between fixed beamformers (also known as delay-and-sum beamformers) and adaptive beamformers.

Delay-and-Sum Beam Forming

A delay-and-sum beamformer (DSBF) requires a priori knowledge about the spatial layout of the microphone array and sound source direction. Taken from [71], the sound from this specified direction is then amplified by delaying and summing the signals from the microphone array:

$$s_{\mathrm{BF}}\left[k,l,\theta_d\right] = \frac{1}{M}\sum_{m=1}^{M} x\left[k,l\right] \mathrm{e}^{j2\pi f_k \tau(1,m,\theta_d)}. \tag{4.4}$$

The time constant τ indicates the relative transmission delay of the source sound to two individual microphones:

$$\tau(m_1,m_2,\theta_d) = \frac{\|\boldsymbol{r}_{m_2} - \boldsymbol{r}_{\theta_d}\| - \|\boldsymbol{r}_{m_1} - \boldsymbol{r}_{\theta_d}\|}{c}. \tag{4.5}$$

[75] have implemented DSBF to locate a speech sound source while the MAV is hovering, in combination with an array of 16 microphones in an octagonal configuration with a diameter of $2.3m$. They concluded that DSBF provided satisfactory localization in 60% of the test cases.

Adaptive Beam Forming

An adaptive beamformer (ABF) generally outperforms a DSBF [73] and does not require the location of the target sound and acoustic sensors. Instead, only the correlation matrices of the target sound *or* the noise is required. The correlation matrix $\boldsymbol{R}_{ss}(k,l)$ of the target sound $s[k,l]$ is computed by

$$\boldsymbol{R}_{ss}(k,l) = \mathrm{E}\{\boldsymbol{s}(k,l)\boldsymbol{s}^{\mathrm{H}}(k,l)\} = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{s}(k,l)\boldsymbol{s}^{\mathrm{H}}(k,l). \tag{4.6}$$

Correlation matrices $\boldsymbol{R}_{vv}(k,l)$ and $\boldsymbol{R}_{xx}(k,l)$ for the noise sources and sensor output respectively are defined in an identical manner. Under the assumption that the target sound and noise signals are generated independently, $\boldsymbol{R}_{xx}(k,l)$ can also be expressed as the sum of $\boldsymbol{R}_{ss}(k,l)$ and $\boldsymbol{R}_{vv}(k,l)$. When these correlation matrices are known, a filter w can be obtained from e.g. a generalized eigen-vector decomposition of $\boldsymbol{R}_{xx}$ and $\boldsymbol{R}_{vv}$, with w corresponding with the generalized eigenvector of the largest eigenvalue [71].

Unfortunately, the correlation matrices of s or v are difficult to obtain since s and v themselves are unknown. An alternative is to estimate $\boldsymbol{R}_{vv}$ from frames where no target sounds are present, i.e. where $v[k,l] = x[k,l]$, but there exists no feasible way to detect these frames for an in-flignt MAV with non-stationary noise and low SNR.

4.2.2 Blind Source Separation

Blind Source Separation (BSS) aims to separate all individual sources from the mixed signal obtained from the $p+2$ microphones, by performing an Independent Component Analysis (ICA) followed by permutation alignment [71]. The ICA estimates a demixing matrix which transforms $x[k,l]$ into a set of maximally independent components. According to the Central Limit Theorem, a sum of non-Gaussian signals is more Gaussian than its individual components. One way to interpret ICA is to find a demixing matrix which maximizes the non-Gaussianity of each individual signal. This can be accomplished via e.g. the InfoMax algorithm [76], under the assumption that all signals are independently generated. The permutation ambiguity can be solved based on the inter-frequency dependency of separated signals, such that a frequency-independent global reference is obtained for each source [77].

A limitation of the BSS approach is that it requires a stationary mixing network for a given interval, i.e. the position of noise- and target source must remain fixed relative to the microphone array. While it has led to promising results for a moving MAV recording a static speaker [73], it is not clear how it will work in an outdoor environment.

4.2.3 Time-Frequency Processing

Time-frequency (T-F) processing methods exploit the time-frequency sparsity of audio signals, by estimating the Direction Of Arrival (DOA) of the target sound at each time-frequency bin and then combining these results to reduce noise [71]. Like DSBF, the T-F process does require an angle of arrival of the target sound. The time-frequency bins belonging to the target sound are then detected by measuring how close each bin is to the direction of the target sound. From here, a target correlation matrix is computed, such that an adaptive beamformer can be formulated. A drawback of T-F processing is that its performance deteriorates when the DOA of the target sound is close to the DOA of one of the noise sources. The T-F method slightly outperforms BSS for signals with low SNR, while both vastly outperform delay-and-sum as well as adaptive beamformers [71].

[78] extend their T-F method to track moving sound sources with a hovering MAV. by dividing the audio streams into temporal windows, and estimating the target location using a time-frequency filter. This filter consists of three steps: spatial filtering, spatial confidence estimation and peak tracking.

4.3 Dependent Approaches

Dependent approaches use additional sensory information to construct an adaptive filter or to predict the ego-noise signature of the robot. MAV applications typically require the rotational speed of the rotors is necessary [79]. Because of the limited research into dependent approaches for MAVs, this section also highlights ego-noise estimation approaches for ground-based robots.

4.3.1 Order-Analysis

[79] use an order-analysis approach for a hovering MAV, resulting in a spectral filter obtained by representing the microphone signal in the revolution-order domain instead of the time-frequency domain. They make use of the assumption that ego noise is predominantly generated by the motors at harmonic orders of the fundamental frequency. The noise-free signal is then obtained by subtracting the energy at the harmonic orders from the total energy within the spectrum. A drawback of this method is that it does not discriminate between target sound and unwanted noise; any target sound present at the harmonic orders will not be observable in the noise-free signal. This may especially be a problem for a manoeuvring MAV with motors operating at different rotational speeds.

4.3.2 Template Methods

Template-based methods utilize a database of ego-noise templates, where each template contains the ego-noise signature generated from a set of feature vectors. [80] apply template-based online learning to a ground-based robot. They make use of the nearest neighbor algorithm to match the observed feature vector with the closest feature vector in the database, and retrieve the corresponding template. If no external noise is present, and the estimated template is not similar to the closest template from the database, the estimated template and corresponding feature vector are added to the database.

4.4 Denoising Autoencoders

An autoencoder is a type of feedforward network which uses an encoder-decoder scheme in attempt to output a reconstruction of an input $\boldsymbol{x}$. The autoencoder is usually restricted to be undercomplete: by making the hidden layer(s) smaller than the input layer, the encoder is forced to compress the most salient features of the input. The loss function of an autoencoder is usually a function that penalizes the reconstruction of the input for being dissimilar to the original, such as the MSE.

A denoising autoencoder (DAE) aims to reconstruct $\boldsymbol{x}$ from a copy of $\boldsymbol{x}$ corrupted by noise. This way, the autoencoder is forced to implicitly learn the distribution of the input. Recently, the DAE has had some success for the denoising/enhancement of speech signals. [81] used a DAE consisting of a single hidden layer, in combination with the 40-band Mel frequency power spectrum. Evaluating on car- and factory noise at SNRs of +10, +5 and 0dB, the DAE outperformed existing algorithms for all six test cases. It should be noted, however, that these noise datasets are significantly more stationary than MAV ego-noise.

Chapter 5

Literature Synthesis

This chapter concludes the literature study for the preliminary phase of this thesis. The research topic of the thesis is the recognition of aircraft based on their acoustic signature, from an in-flight MAV. The literature study has been divided into three main areas, which have been analyzed in detail. Firstly, the properties of aircraft sound were analyzed, and many audio representations suitable for detection were covered. Secondly, an overview of existing methods for general sound event detection (SED) methods suitable for aircraft detection was presented. Finally, the feasibility of existing methods regarding the suppression of MAV ego-noise are discussed. The main findings of each topic are synthesized in this chapter.

5.1 Properties and Representation of Aircraft Sound

Aircraft sound as perceived on the ground consists primarily of harmonic sounds generated by its rotating engines, as well as broadband noise from the turbulent flow across the airframe [19]. Furthermore, an aircraft passing by is characterized by a Doppler shift for any (near-)stationary observer.

When classifying a sound signal, it is often necessary to extract relevant features from it and discard irrelevant information. Engineered features can be divided into purely time-, purely frequency- or time-frequency features. The application of purely-time frequency features, such as the short-time energy (STE) and the zero-crossing rate (ZCR), is limited to simple cases such as silence presence/absence detection [30]. The log Mel-band energy is a powerful frequency-based feature suitable for more complex tasks such as polyphonic SED [32] and multi-channel SED [33]. The Mel Frequency Cepstrum Coefficients (MFCCs), operating in the cepstral domain, have been the standard for audio classification tasks for decades, but have recently been outperformed by time-frequency features such as the spectrogram image feature (SIF). Inspired by recent advances in computer vision, these features use a spectrogram representation of the signal to obtain characteristics regarding the shape and evolution of its time-frequency contents [2].

5.2 Sound Event Detection for Aircraft

Aircraft presence/absence detection can be considered a binary detection task, and can be accomplished by piece-wise classification of the incoming audio signal. Classifiers can be divided into generative classifiers, which model the feature space, and discriminative classifiers, which learn the difference between classes. Generative classifiers include the Gaussian Mixture Model (GMM) and the Hidden Markov Model (HMM). In the past, these classifiers have been frequently combined with MFCCs for audio classification [28]. Discriminative classifiers such as the Support Vector Machine (SVM) and logistic regression generally outperform generative classifiers.

Due to advances in computer processing power in recent years, Artificial Neural Networks (ANNs) have emerged as the main choice for audio analysis tasks such as automatic speech recognition (ASR), environmental sound classification (ESC) and robust SED. State-of-the-art performance has been achieved by convolutional neural networks (CNNs) in combination with log-amplitude Mel-frequency SIFs. As such, this combination will be chosen for further analysis during the remainder of the thesis.

5.3 Ego-Noise Analysis

Classification of aircraft sound by an in-flight MAV is impeded by its ego-noise. This noise can be considered a mixture of narrow-band harmonic noise, generated by the rotating engines, and broadband noise, generated by the propeller blades cutting through the air [73]. Self-contained ego-noise filtering approaches sucn as beam forming (BF) and blind source separation (BSS) use a microphone array to design a spatial filter which localizes and extracts the target sound. However, these methods do not account for a dynamic environment.

Dependent approaches make use of additional information provided by sensors on the MAV. Research regarding these approaches has been limited. [79] used an order-analysis approach to filter out all incoming sound at harmonic frequencies of the fundamental frequency of the motors. [80] made use of a template-based approach for estimating the ego-noise of a ground-based robot. A different type of approach is to use a denoising autoencoder, which has been applied successfully for denoising speech signals contaminated with noise [81].

The remainder of the thesis will focus on two test cases for ego-noise removal and aircraft classification. The first test case filters the ego-noise via a regression approach, and then detects aircraft using the denoised audio signal. The second test case will filter ego-noise and detect aircraft simultaneously.

III

Preliminary Analysis

Chapter 6

Aircraft Classification

This chapter considers a preliminary analysis of acoustic-based aircraft classification. The goal of this analysis is to compare the performance of various feature representations and input dimensions coupled with CNN architecture found in literature. The method of obtaining the data required for analysis is outlined in Section 6.1. In Section 6.2, the feature extraction process is explained in detail. Finally, the evaluated architectures and results of the analysis are given in Section 6.3.

6.1 Data Acquisition

During flight, it is important that the MAV does not register nearby road vehicles as aircraft due to their engine noise. Likewise, wind noise during flight must also not be mistaken for aircraft sound. For this reason, a classifier that distinguishes between aircraft and non-aircraft sound is necessary. The ESC-50 dataset [1] is used for training and evaluation of the classifier, as it is one of the few publicly available datasets containing environmental recordings of helicopters. The dataset consists of 50 classes which can be arranged into 5 major categories. Each class itself contains 40 five-second recordings. An overview of all classes per category is given in Table 6.1. Naturally, the classes 'helicopter' and 'airplane' are considered as 'aircraft'. To keep a balanced dataset, only two other classes will be designated as non-aircraft. Out of the ones remaining, the classes 'wind' and 'engine' appear to be the two most prominent external sound sources during MAV flight and are denoted as the 'non-aircraft' class. The dataset is then split into a training, validation and test set with {0.64, 0.16, 0.20} ratios between the respective sets. Each of the classes is evenly represented in each set, resulting in {25, 7, 8} recordings per class in the training, validation and test set, respectively.

The 46 remaining classes are not considered for training/evaluation of the classifier during this preliminary analysis. A disadvantage of this approach is that the remainder of the dataset is rather small: only 160 recordings, equivalent to 800 seconds of audio, remain. AudioSet [82] is a significantly larger dataset, consisting

Animals	Nature	Human	Interior	Exterior/Urban
Dog	Rain	Crying baby	Door knock	**Helicopter**
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door creaks	Car horn
Frog	Chirping	Coughing	Can opening	***Engine***
Cat	Water drops	Footsteps	Washing machine	Train
Hen	***Wind***	Laughing	Vacuum cleaner	Church bells
Insects	Pouring water	Brushing teeth	Clock alarm	**Airplane**
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking	Glass breaking	Hand saw

Table 6.1: Overview of the classes in the ESC-50 dataset. Classes considered as 'aircraft' are typeset in bold, while classes considered 'non-aircraft' are typeset in bold italics. All other classes are not considered for training/evaluation. Table adapted from `https://github.com/karolpiczak/ESC-50`

of over 2 million 10-second audio clips. The set uses multi-class labeling, i.e. a clip may contain more than 1 label. Although the set offers approximately 12,000 clips of aircraft sound ('aircraft', 'helicopter'), speech-dominated events such as a conversations in an airplane cockpit or a crowd jeering during an airplane fly-over are also included in this labeling. Even when filtering all clips containing any kind of 'speech' label, clips where the aircraft is simply background noise are still present. For this reason, the small but accurate fraction of the ESC-50 dataset is preferred over AudioSet.

6.2 Feature Extraction

After acquiring the data, an input feature must be extracted from the raw audio. The Mel-Frequency Cepstral Coefficients (MFCCs) were considered the go-to feature for sound-related tasks, particulary speech recognition, prior to the advent of deep learning in the 2010s. Gaussian Mixture Models (GMMs) especially benefited from the decorrelation of the filterbank coefficients that the DCT provided. Given the ability of Convolutional Neural Networks (CNNs) to efficiently learn local structures within a 2D input, image-like features based on the spectrogram of a waveform have outperformed manually engineered features such as the MFCCs. Even for environmental sound classification tasks, the Mel-frequency scale remains a popular alternative to linear scaling. Four feature variations will be evaluated during this analysis: the MFCCs, the (linear) spectrogram, the Mel-spectrogram (scaling based on human perception of pitch) and the constant-Q spectrogram (scaling based on the Western musical scale).

Unlike the Mel-scale, which is linear until approximately 700 Hz, the constant-Q scale is purely logarithmic. This requires a minimum frequency to be chosen, which is set at the 'C1' tone (32.7 Hz). The scale also assumes 12 steps per octave. An overview of the constant-Q tones and corresponding frequencies, as well as the resulting frequency bins and minimum required sample rate out of the ranges {24,000; 32,000; 44,100} Hz are shown in Table 6.2.

Frequency bins	0	12	24	**36**	48	**60**	72	**84**	96	**108**
Max. tone	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Max. frequency (Hz)	32.7	65.4	130.8	261.6	523.2	1,046.5	2,093.0	4,186.0	8,372.0	16,744.0
Min. sample frequency (Hz)	-	-	-	24,000	-	24,000	-	24,000	-	44,100

Table 6.2: Relationship between the number of frequency bins used for the constant-Q transform, the frequency of the highest tone, and the minimum necessary frequency out of the ranges {24,000, 32,000, 44,100} Hz.

All spectra are extracted from the audio using the *librosa* python library[83]. The complete feature extraction process for each recording is almost identical for each feature, and consists of the following steps:

1. Trim the silent fragments in the recording.
2. Extract the Short Time Fourier Transform, using a hop size of 512 frames and a window length of 1,024 frames (the constant-Q transform uses a variable window length), with appropriate frequency scaling (linear, Mel or constant-Q) and appropriate number of frequency bins and sample rate. For MFCCs, only the first 13 coefficients are used, computed from the Mel filterbank.
3. Convert the time-frequency representation to the decibel scale.
4. Split the recording into segments with the appropriate number of frames, with 50% overlap between segments. The frames are offset such that they are centered w.r.t. the recording, e.g. a five-second recording sampled at 44.1 kHz (containing 431 frames) with a segment length of 60 is split using frame indices {6-65, 36-95, ..., 366-425}.
5. Rescale each segment to a [0, 1] range. Scaling or normalization generally improves the learning capability and numerical stability of the model.
6. Use a first-order Savitzky–Golay filter to compute the smoothed time derivative (delta) of the feature, with 'mirror' padding at the edges. This provides dynamic information to go along with the static spectra.

Example features of each class (airplane, helicopter, engine, wind) are displayed in Figure 6.1 to Figure 6.4. Each feature was generated using a sample rate of 44.1 kHz, 108 frequency bins (13 coefficients for MFCCs) and 84 frames.

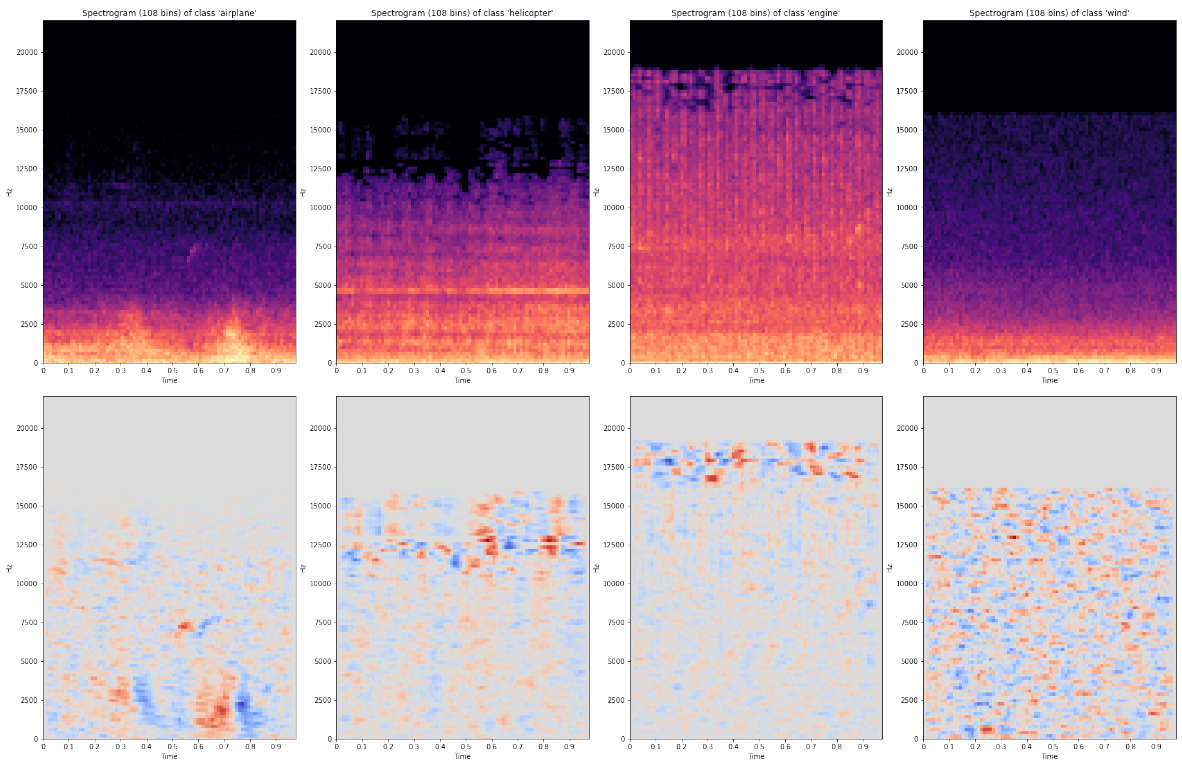

Figure 6.1: Example spectra (top) and deltas (bottom) of the classes 'airplane', 'helicopter', 'engine' and 'wind'.

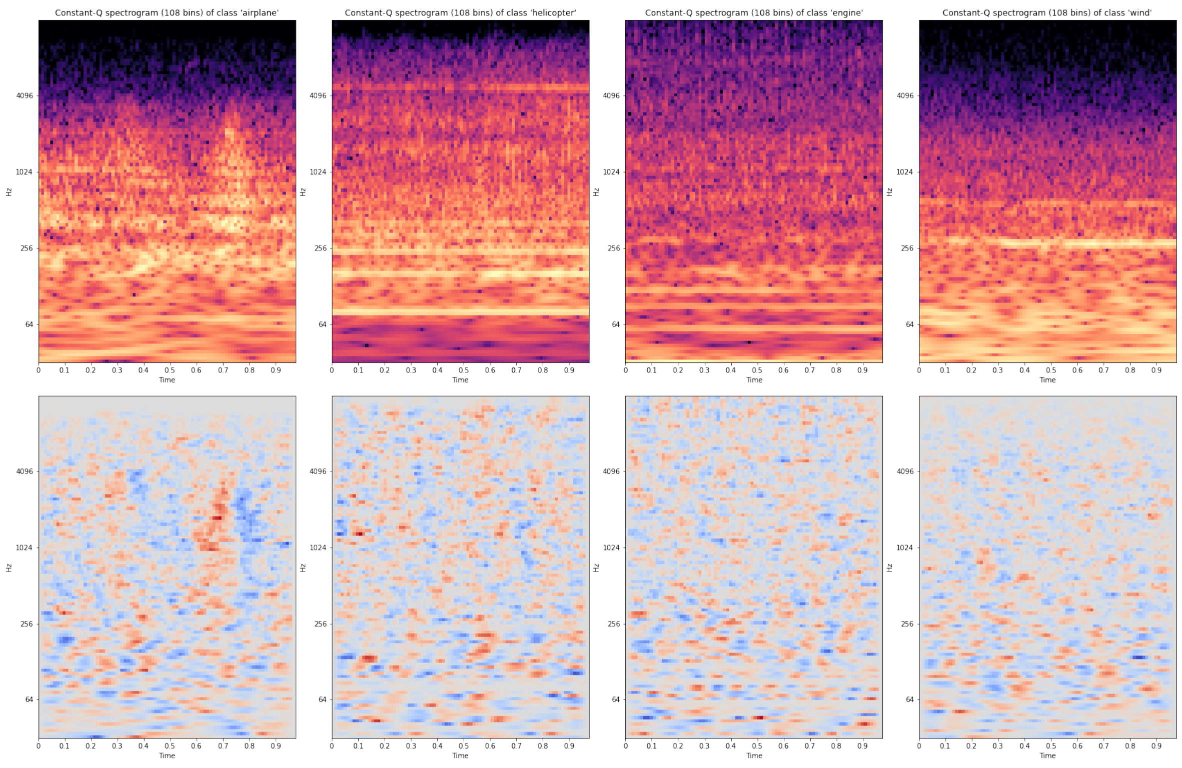

Figure 6.2: Example constant-Q spectra (top) and deltas (bottom) of the classes 'airplane', 'helicopter', 'engine' and 'wind'.

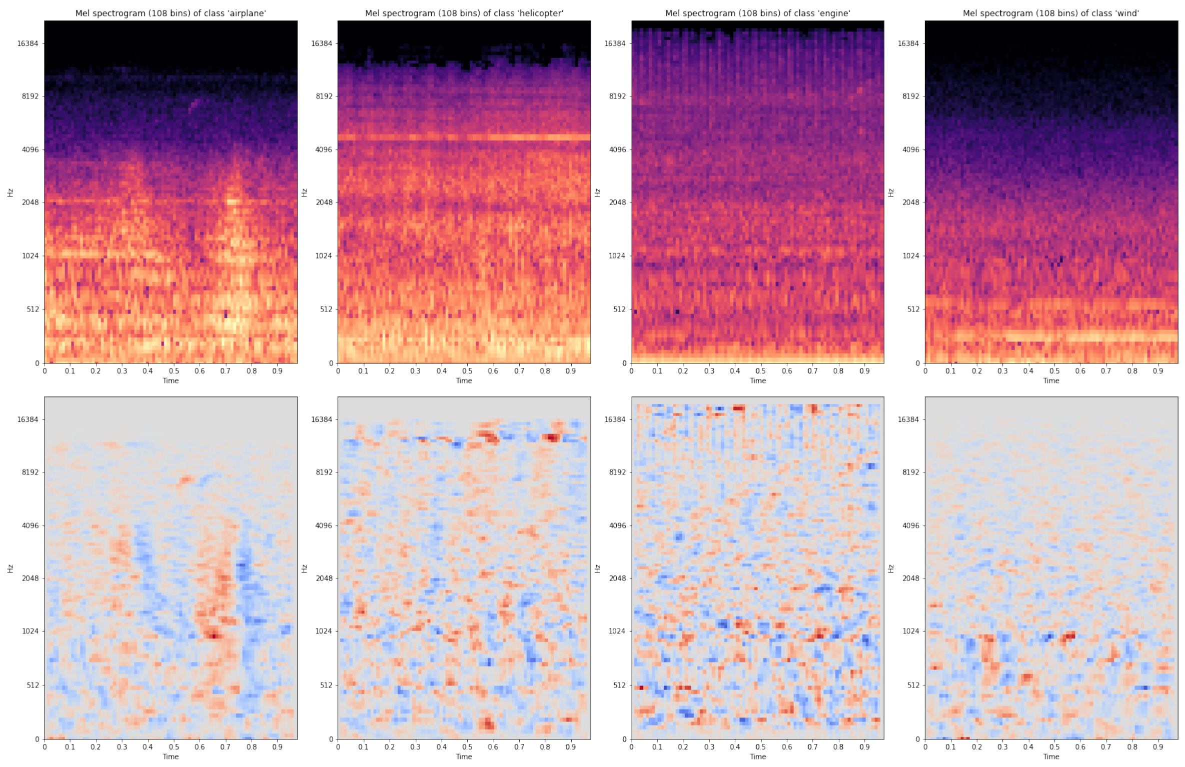

Figure 6.3: Example Mel spectra (top) and deltas (bottom) of the classes 'airplane', 'helicopter', 'engine' and 'wind'.

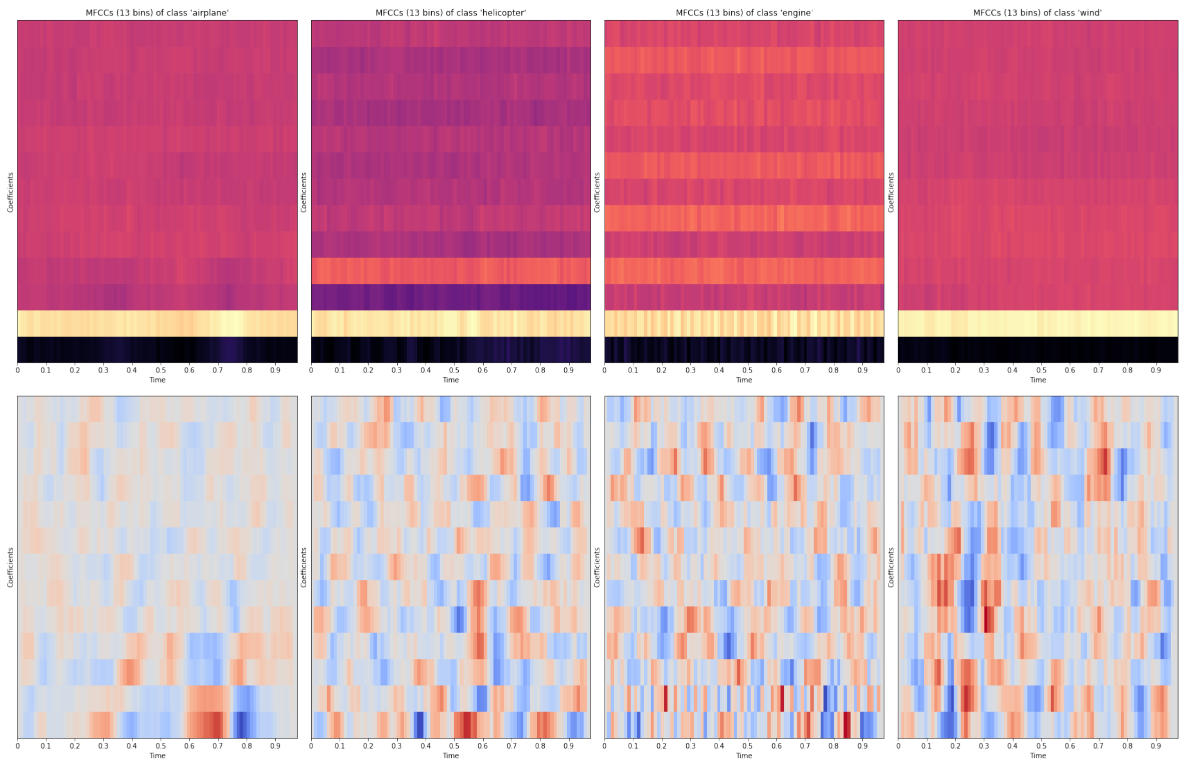

Figure 6.4: Example MFCCs (top) and deltas (bottom) of the classes 'airplane', 'helicopter', 'engine' and 'wind'.

6.3 Preliminary Analysis

6.3.1 Experiment Setup

Various settings for the analysis will be explored, for a total of 42 combinations per feature. The first setting to be varied is the sample frequency. Because the majority of the energy of aircraft sound appears to be contained at frequencies below 12kHz, the sample frequency is varied at 24.0, 32.0 and the default 44.1 kHz. Second is the number of frequency bins, the 'height' of each input to the network, varied at 36, 60, 84 and 108 bins. Because the constant-Q scale contains twelve steps per octave, these are all integer multiples of 12. A height of 108 requires a sample frequency of at least 32.5 kHz (Table 6.2) and is thus only done for the default sample rate. For MFCCs, only the first 13 coefficients are used, computed from the Mel filterbank. Third is the 'width' of each input, which is determined by the frames per split segment. Each test case contains one square input (i.e. frequency bins = frames per segment) and a fixed size at 60. An additional test with 108 frequency bins and 84 frames is done to compensate for the lack in frequency variation. Finally, each test is repeated with and without the delta-channel as the second channel input.

Two CNN architecture types will be explored for each set of features. The first type is based on the work of [4], who also created the ESC-50 dataset. Characteristic of this network is the tall filters found in the first convolutional layer, which span almost the entire frequency range. The second network is based on the work of [5], which is a more conventional CNN with square filters in all convolutional layers. Both networks are relatively simple, consisting of 2-3 convolutional layers and 1-2 fully-connected layers. As is convential with CNNs, both networks use ReLu connections for every hidden layer. Because the dataset used is considerably smaller, the number of parameters has been reduced greatly by reducing the width of each layer for both models. Table 6.3 and Table 6.4 shows the modifications made to the networks of [4] and [5], respectively.

	Piczak	Modified
Input	(2, 60, 41)	various
Conv1	(80, 57, 6)	(16, H-3, 7)
Mp1	(4, 3) kernel, (1, 3) stride	*identical*
Conv2	(80, 1, 3)	(16, 1, 3)
Mp2	(1, 3)	*identical*
Fc1	(240, 5000)	(various, 64)
Fc2	(5000, 5000)	(64, 32)
Out	(5000, 50)	(32, 1)
Dropout	Conv1, Fc1, Fc2	Fc1, Fc2

Table 6.3: Comparison of the architecture of [4] (left) and the modifications made to it (right).

	Salamon	Modified
Input	(1, 128, 128)	various
Conv1	(24, 5, 5)	(16, 5, 5)
Mp1	(2, 4)	(2, 2)
Conv2	(48, 5, 5)	(16, 5, 5)
Mp2	(2, 4)	(2, 2)
Conv3	(48, 5, 5)	(16, 5, 5)
Fc1	(10000, 64)	(various, 64)
Out	(64, 50)	(64, 1)
Dropout	Conv3, Fc1	Fc1

Table 6.4: Comparison of the architecture of [5] (left) and the modifications made to it (right).

Because this is a binary classification problem, the loss function of choice is the binary cross-entropy (BCE):

$$L_{BCE} = \frac{1}{N}\sum_{i=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right], \tag{6.1}$$

where y_n indicates the target output (0/1) and $\hat{y}_n$ the predicted output. A batch size of 256 is used during training (i.e. N=256 in Equation (6.1)), as it increases computational speed (due to parallelization) without negatively affecting the generalization performance of the network. The model is implemented using Pytorch [84] and trained on a single GeForce GTX 1050Ti GPU via CUDA [85]. The AdamW algorithm is used to train the model, with a learning rate of 0.0001 and weight decay of 0.01. Training is done for a maximum of 500 epochs, with early stopping after the validation loss does not improve after 30 epochs. The random seed, which determines the batch shuffling and initial weights of the network, is kept fixed and is reset before the training of each set, in an attempt to reduce the influence of randomness during training.

6.3.2 Results

The 20 best configurations (out of 328) are summarized in Table 6.5, showing the accuracy, model architecture and parameter settings. Table 6.6 and Table 6.7 show the best performing feature for the modified Piczak

Accuracy	Model	Feature	Sample rate	Channels	Height	Width
87.9%	Salamon	Mel spectrogram	44,100	2	108	84
87.8%	Salamon	Mel spectrogram	44,100	1	84	60
87.5%	Salamon	Mel spectrogram	44,100	1	108	84
87.1%	Salamon	Mel spectrogram	44,100	2	84	60
86.7%	Salamon	Mel spectrogram	32,000	2	84	84
86.4%	Salamon	Mel spectrogram	44,100	2	36	36
86.3%	Salamon	Mel spectrogram	44,100	1	108	60
86.3%	Salamon	Mel spectrogram	44,100	1	36	36
86.3%	Salamon	Spectrogram	44,100	2	108	108
86.1%	Salamon	Mel spectrogram	44,100	1	36	60
86.0%	Salamon	Mel spectrogram	24,000	1	84	84
85.7%	Salamon	Mel spectrogram	24,000	1	60	36
85.6%	Salamon	Spectrogram	44,100	2	108	60
85.4%	Piczak	Mel spectrogram	24,000	2	60	36
85.1%	Salamon	Mel spectrogram	32,000	1	84	84
84.9%	Salamon	Spectrogram	44,100	1	108	84
84.6%	Salamon	Mel spectrogram	44,100	1	108	108
84.6%	Salamon	Mel spectrogram	44,100	2	36	60
84.3%	Salamon	Spectrogram	44,100	1	84	60
84.3%	Salamon	Spectrogram	44,100	1	60	60

Table 6.5: Results of prelim classification

Accuracy	Feature
85.4%	Mel spectrogram
84.1%	Spectrogram
82.3%	Constant-Q spectrogram
82.3%	MFCCs

Table 6.6: Highest classification accuracy per feature for the modified Piczak architecture.

Accuracy	Feature
87.9%	Mel spectrogram
86.3%	Spectrogram
81.5%	Constant-Q spectrogram
75.2%	MFCCs

Table 6.7: Highest classification accuracy per feature for the modified Salamon architecture.

and Salamon architecture, respectively. From these results, several observations can be made. It is clear that the network with square filters, derived from [5], outperforms the one from [4]. The Mel spectra also consistently outperform all other features. It is also apparent that the constant-Q spectra perform considerably worse than the other spectra, despite having the highest resolution at lower frequencies, were vehicle sound is expected to be dominant. It seems that the high frequency components are necessary to differentiate them. Tall input features also appear to outperform square ones, with detailed input (108, 84) appearing to have a slight edge over coarse inputs (36, 60). Finally, the delta-channel does not appear to influence the results significantly (87.9% and 87.1% with compared to 87.8% and 87.5% without deltas). The Receiver Operator Characteristic (ROC) curve is displayed in Figure 6.5.

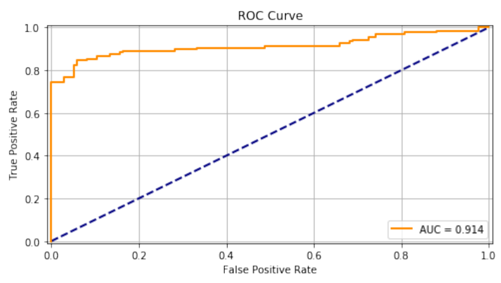

Figure 6.5: ROC-curve of the classifier. Area under the curve = 0.914.

Chapter 7

Ego-Noise Prediction

The objective of the preliminary ego-noise analysis is to assess whether a meaningful relationship between ego-noise and MAV states (e.g. rotor rotational speed, body angular velocities). Section 7.1 explains the steps taken to generate and prepare the dataset for learning and evaluation. Section 7.2 provides the motivation and outcomes of the experiments conducted during this phase of the research.

7.1 Data Acquisition and Pre-processing

A Parrot Bebop2 drone was provided by the faculty to collect the training- and evaluation data. A Raspberry Pi 3 (RPi), fitted with a USB microphone, was mounted on (and powered by) the MAV to record audio data. The measured states consist of the rotor rpm (4), the stabilization commands by the autopilot (4), attitude angles (3) and body angular rates (3), all of which were logged internally on the MAV. Four manual flights, lasting approximately 30 seconds each, were performed for the data collection, inside an indoor area of 8 by 8 meter.

Paparazzi UAV, running on a nearby laptop, is used to control the drone. Data transmission from MAV to ground station is done via UDP, while transmission from RPi to ground station is done via TCP. As soon as all motors are turned on, the MAV signals the ground station and starts transmitting state data at a 100 Hz frequency. In turn, the ground station signals the RPi to start recording (and transmitting) audio. The microphone records at a frequency of 48 kHz, while the data is sent in chunks of 512 samples. When the motors are turned off, the MAV signals the laptop again, which in turn signals the RPi to stop recording audio.

Several adjustments must be made to the raw measurements before the ego-noise analysis. Similar to the audio data obtained from the ESC-50 dataset (Section 6.1), the recorded noise is converted to a suitable time-frequency representation. In this case, the audio is converted to the best performing feature representation from the aircraft classification task: a Mel spectrogram sampled at 44.1 kHz, containing 108 frequency bins, with a hop size and window length of 512 and 1024 samples, respectively. Each state vector is augmented with the time-derivative (delta) of the rotor rpm and stabilization commands to provide dynamic information. All recorded states are then scaled so that they fall within a reasonable interval.

The Mel spectrogram of the flight, along with the scaled rotor speed, stabilization commands and attitude rates are displayed in Figure 7.1. In this recording, the MAV is on the ground, with motors on, for 2.5 seconds. It then takes off and reaches a stable position after another 1.5 seconds. It then performs mostly horizontal flight for 15 seconds, with a sharp yaw motion at t=13, and ends the flight with a rough landing.

It is important that the data from each individual source is synchronized: each frame of the Mel-spectrogram must be matched with a corresponding state vector. With 10 ms between successive state vectors and approximately 11.6 ms between frames, each frame is simply matched with the state vector closest in time. State vectors without a matching frame are discarded. The synchronization is illustrated in Figure 7.2 for the first five spectrogram frames. In this case, the spectra at t_{mic} = {0.0, 11.6, 23.2, 34.8, 46.4} ms are matched with the state vectors at t_{mav} = {0.0, 10.0, 20.0, 30.0, 50.0}, respectively. The state vector at t_{mav} = 40 ms is discarded.

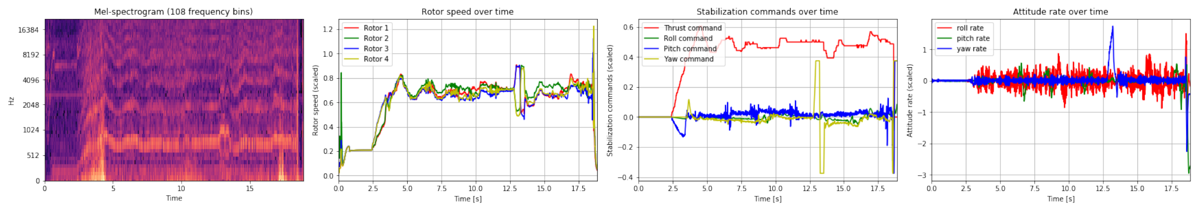

Figure 7.1: One of the manual MAV flights.

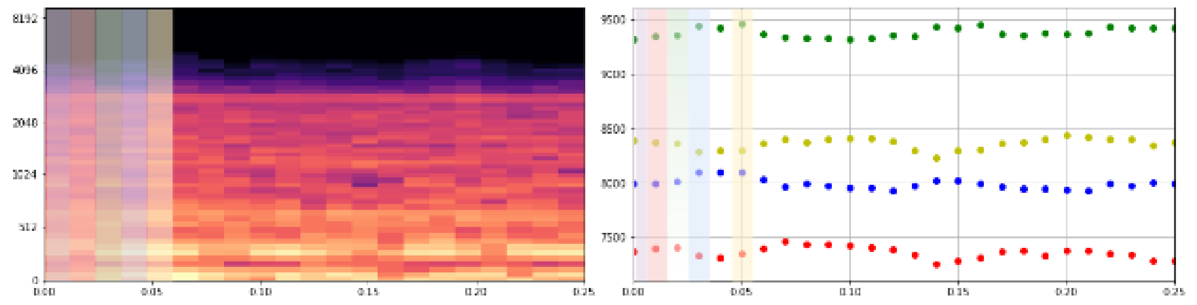

Figure 7.2: Synchronization of the data.

The complete dataset, consisting of the synchronized state-spectrogram pairs of the four recordings, is split up into a training, validation, and test dataset. Two recordings (5,057 datapoints) have been used for training, one (2,849 datapoints) is used for validation and one (1,627 datapoints) is used for testing purposes.

7.2 Experiments

A simple feedforward network, in the form of a Multi-Layer Perceptron (MLP), is used as the model for the ego-noise predictor. The optimal architecture is determined by randomly selecting the number of layers (2-4) and number of units per layer (40-200) for a total of 30 trials. For each randomly selected architecture, eight different state configurations are examined, varying in complexity. The attitude angles are ignored for this analysis.

The chosen loss function is the mean-squared error (MSE), which is standard for a regression analysis:

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{7.1}$$

where y_i is the target output and $\hat{y}_i$ the predicted output. Training is done using the established Adam algorithm, with a learning rate of 0.001 and weight decay of 0.01 (default parameters). A batch size of 256 is used for training, which is stopped after 30 epochs. Again, ReLu activations are used for all hidden layers. Table 7.1 shows the ten best performing network-state combinations (i.e. lowest MSE loss) out of the 240 configurations tested. The indices in the 'States' column denote the used states: rotor RPM (1), stabilization commands (2), rotational rates (3), rotor RPM-delta (4) and stabilization commands-delta (5).

There appears to be little consistency between network architecture and performance, likely due to the mini-batch gradient descent-induced randomness and/or weight initialization during training. The rotor RPM and stabilization commands appear to be the most important states, considering their occurrence in Table 7.1.

To illustrate the difference in performance between configurations, the models with the best performance (MSE=4.976e-3) and worst performance (MSE=6.102e-3, not tabulated in Table 7.1) are illustrated in Figure 7.3 and Figure 7.4, respectively. In each figure, the original spectrum (top), predicted spectrum (middle) and residual spectrum (bottom) are displayed. From the two figures, it becomes clear that the model on the right is unable to follow the changing harmonic frequencies of the MAV during flight, resulting in higher discrepancies. The highest discrepancies for the model on the left occur during take-off, the yawing around 13 seconds, and landing. The inability to adapt to these rapid maneuvers suggests that the availability of acceleration and/or velocity flight data could be beneficial.

MSE loss (e-3)	Structure	States
4.976	19x120, 120x100, 100x140, 140x108	1, 2, 3, 4, 5
5.021	4x60, 60x120, 120x100, 100x108	1
5.044	19x180, 180x160, 160x108	1, 2, 3, 4, 5
5.059	7x140, 140x80, 80x120, 120x108	2, 3
5.070	4x60, 60x160, 160x140, 140x108	2
5.078	4x120, 120x180, 180x108	1
5.093	19x120, 120x140, 140x120, 120x108	1, 2, 3, 4, 5
5.095	11x80, 80x140, 140x108	1, 2, 3
5.097	8x80, 80x108	1, 2
5.111	11x120, 120x180, 180x108	1, 2, 3

Table 7.1: The ten best (i.e. lowest MSE loss) combinations of network structure and MAV states.

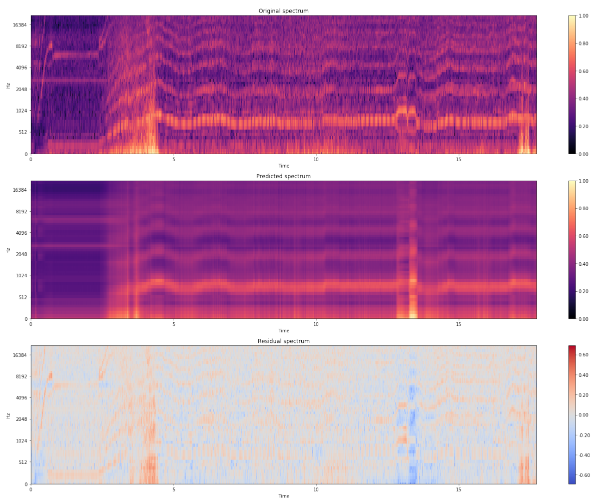

Figure 7.3: Original (top), predicted (middle) and residual (bottom) Mel spectrogram generated by the best performing model (MSE loss = 4.976e-3).

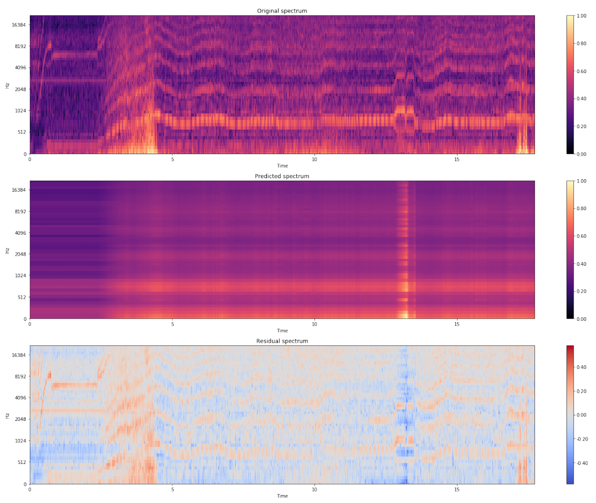

Figure 7.4: Original (top), predicted (middle) and residual (bottom) Mel spectrogram generated by the worst performing model (MSE loss = 6.102e-3).

Chapter 8

Discussion of Preliminary Results

In the preliminary analyses presented in Chapter 6 and Chapter 7, investigations into aircraft detection via classification and ego-noise prediction using MAV flight data were conducted. This chapter summarizes and discusses the findings of the investigations, which form the basis for the experiments conducted in the scientific paper in part I.

8.1 Classification

The preliminary analysis concerning aircraft classification consisted of finding the best combination of feature representation and model architecture. The data for aircraft classification consisted of 40 recordings of airplane, helicopter, engine and wind sound extracted from the ESC-50 dataset [1]. All recordings were converted to a time-frequency representation via the Short-Time Fourier Transform, providing a more suitable representation for classification.

Four different time-frequency features that were either commonly used in the past or showed promising results in recent research regarding sound event classification were investigated during the analysis: the (regular) spectrogram, the constant-Q spectrogram, the Mel spectrogram and the Mel Frequency Cepstral Coefficients (MFCCs). Each of the features was tested in combination with two Convolutional Neural Networks (CNNs) adapted from prior research. One network, inspired by [4], contains tall filters spanning most of the frequency range. The second is a variation of [5], which makes use of square filters. Furthermore, audio sample rate and input dimensions (width, height) were varied during the analysis.

The best performing combination consisted of a 108 by 84 Mel spectrogram, sampled at 44.1 kHz, achieving 87.9% accuracy on the test set. While this is an adequate result for the preliminary analysis, improvements are still necessary. Furthermore, the final model must also be robust to residual ego-noise, caused by the discrepancy between actual and predicted ego-noise.

8.2 Ego-Noise

The objective of the preliminary ego-noise analysis was to assess whether ego-noise could be satisfactorily predicted using MAV flight data such as rotor rotational speed, stabilization commands and angular velocities. Audio data was obtained from a microphone located near the Bebop2 MAV, flight data was logged by the MAV itself. To remain consistent with the aircraft classification, the audio was transformed to the optimal time-frequency representation described in the previous section. The gathered MAV states (input) and Mel spectra (output) were then synchronized to construct the dataset used for ego-noise prediction.

The chosen network configuration was a Multi-Layer Perceptron (MLP). The best architecture (number of layers, units per layer) and state vector combination was found by conducting a random grid search through the network depth and layer width for each combination of state vectors (motor speed, stabilization commands, angular velocity, motor speed differential and stabilization command differential).

The best model consisted of 3 hidden layers with 100 to 140 units each and made use of the complete state vector. However, no clear relationship between the architecture of the best, near-best and worst configurations could be found; it could simply be the result of randomness during training. While the best model was able to follow the wave-like variations in dominant frequencies during flight, major discrepancies were still present during rapid movements such as take-off. This may indicate that positional data (acceleration, velocity) is required. Another shortcoming of the models used during this analysis is the lack of temporal integration between successive data; the noise generated by a rapid acceleration may still be present several frames later. This introduces a need for learning sequential data through e.g. a Recurrent Neural Network (RNN), which must be investigated further.

Bibliography

[1] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018, ACM Press, 2015.

[2] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*. Springer, 2018.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[4] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2015.

[5] 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, 2017.

[6] G. Wild, J. Murray, and G. Baxter, "Exploring civil drone accidents and incidents to help prevent potential air disasters," *Aerospace*, vol. 3, no. 3, p. 22, 2016.

[7] M. G. Wu, A. C. Cone, S. Lee, C. Chen, M. W. Edwards, and D. P. Jack, "Well clear trade study for unmanned aircraft system detect and avoid with non-cooperative aircraft," in *2018 Aviation Technology, Integration, and Operations Conference*, p. 2876, 2018.

[8] L. R. Salazar, R. Sabatini, S. Ramasamy, and A. Gardi, "A novel system for non-cooperative uav sense-and-avoid," in *Proceedings of european navigation conference*, 2013.

[9] S. Ramasamy, R. Sabatini, and A. Gardi, "Avionics sensor fusion for small size unmanned aircraft sense-and-avoid," in *2014 IEEE Metrology for Aerospace (MetroAeroSpace)*, pp. 271–276, May 2014.

[10] X. Yu and Y. Zhang, "Sense and avoid technologies with applications to unmanned aircraft systems: Review and prospects," *Progress in Aerospace Sciences*, vol. 74, pp. 152–166, 2015.

[11] D. M. Marshall, R. K. Barnhart, S. B. Hottman, E. Shappee, and M. T. Most, *Introduction to unmanned aircraft systems*. Crc Press, 2016.

[12] A. Moses, M. J. Rutherford, and K. P. Valavanis, "Radar-based detection and identification for miniature air vehicles," in *2011 IEEE International Conference on Control Applications (CCA)*, pp. 933–940, Sep. 2011.

[13] G. Ludeno, I. Catapano, G. Gennarelli, F. Soldovieri, A. R. Vetrella, A. Renga, and G. Fasano, "A micro-uav-borne system for radar imaging: A feasibility study," in *2017 9th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*, pp. 1–4, June 2017.

[14] T. G. McGee, R. Sengupta, and K. Hedrick, "Obstacle detection for small autonomous aircraft using sky segmentation," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 4679–4684, April 2005.

[15] T. Zsedrovits, A. Zarandy, B. Vanek, T. Peni, J. Bokor, and T. Roska, "Visual detection and implementation aspects of a uav see and avoid system," in *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, pp. 472–475, IEEE, 2011.

[16] A. Rozantsev, V. Lepetit, and P. Fua, “Detecting flying objects using a single moving camera,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 879–892, May 2017.

[17] S. Hwang, J. Lee, H. Shin, S. Cho, and D. Shim, “Aircraft detection using deep convolutional neural network in small unmanned aircraft systems,” in *2018 AIAA Information Systems-AIAA Infotech Aerospace*, 01 2018.

[18] E. Tijs, G. de Croon, J. Wind, B. Remes, C. De Wagter, H. de Bree, and R. Ruijsink, “Hear-and-avoid for micro air vehicles,” in *Proceedings of the International Micro Air Vehicle Conference and Competitions (IMAV), Braunschweig, Germany*, vol. 69, 2010.

[19] G. J. J. Ruijgrok, *Elements of Aviation Acoustics.* VSSD, 2007.

[20] T. F. Quatieri, *Discrete-time speech signal processing: principles and practice.* Pearson Education India, 2006.

[21] A. Pico, G. Schillaci, V. V. Hafner, and B. Lara, “How do i sound like? forward models for robot ego-noise prediction,” in *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pp. 246–251, Sep. 2016.

[22] G. Schillaci, C.-N. Ritter, V. V. Hafner, and B. Lara, “Body representations for robot ego-noise modelling and prediction. towards the development of a sense of agency in artificial agents,” in *Proceedings of the Artificial Life Conference 2016 13*, pp. 390–397, MIT Press, 2016.

[23] G. Heinzel and A. Rüdiger, “Spectrum and spectral density estimation by the discrete fourier transform (dft), including a comprehensive list of window functions and some new flat-top windows,” *Max Plank Inst*, vol. 12, 01 2002.

[24] D. Griffin and J. Lim, “Signal estimation from modified short-time fourier transform,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[25] J. C. Brown, “Calculation of a constant q spectral transform,” *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[26] A. Rakotomamonjy and G. Gasso, “Histogram of gradients of time–frequency representations for audio scene classification,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, pp. 142–153, Jan 2015.

[27] T. Lidy and A. Schindler, “CQT-based convolutional neural networks for audio scene classification,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, pp. 60–64, September 2016.

[28] C. Asensio, M. Ruiz, and M. Recuero, “Real-time aircraft noise likeness detector,” *Applied Acoustics*, vol. 71, no. 6, pp. 539–545, 2010.

[29] S. S. Stevens, J. Volkmann, and E. B. Newman, “A scale for the measurement of the psychological magnitude pitch,” *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[30] R. Bachu, S. Kopparthi, B. Adapa, and B. Barkana, “Voiced/unvoiced decision for speech signals based on zero-crossing rate and energy,” in *Advanced Techniques in Computing Sciences and Software Engineering* (K. Elleithy, ed.), (Dordrecht), pp. 279–282, Springer Netherlands, 2010.

[31] C. Panagiotakis and G. Tziritas, “A speech/music discriminator based on rms and zero-crossings,” *IEEE Transactions on multimedia*, vol. 7, no. 1, pp. 155–166, 2005.

[32] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, “Polyphonic sound event detection using multi label deep neural networks,” in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, July 2015.

[33] S. Adavanne, G. Parascandolo, P. Pertila, T. Heittola, and T. Virtanen, “Sound event detection in multichannel audio using spatial and harmonic features,” in *Scenes and Events 2016 Workshop (DCASE2016)*, p. 6, 2016.

[34] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa, “Computational auditory scene recognition,” in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. II–1941–II–1944, May 2002.

[35] S. O. Sadjadi and J. H. L. Hansen, “Unsupervised speech activity detection using voicing measures and perceptual spectral flux,” *IEEE Signal Processing Letters*, vol. 20, pp. 197–200, March 2013.

[36] M. McKinney and J. Breebaart, “Features for audio and music classification,” in *ISMIR*, 2003.

[37] T. Giannakopoulos, D. Kosmopoulos, A. Aristidou, and S. Theodoridis, “Violence content classification using audio features,” in *Advances in Artificial Intelligence* (G. Antoniou, G. Potamias, C. Spyropoulos, and D. Plexousakis, eds.), (Berlin, Heidelberg), pp. 502–507, Springer Berlin Heidelberg, 2006.

[38] W. B. Kleijn and K. K. Paliwal, *Speech coding and synthesis.* Elsevier Science Inc., 1995.

[39] D. G. Bhalke, C. B. R. Rao, and D. S. Bormane, “Automatic musical instrument classification using fractional fourier transform based- mfcc features and counter propagation neural network,” *Journal of Intelligent Information Systems*, vol. 46, pp. 425–446, Jun 2016.

[40] S. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, pp. 357–366, August 1980.

[41] B. Milner and X. Shao, “Speech reconstruction from mel-frequency cepstral coefficients using a source-filter model,” in *Seventh International Conference on Spoken Language Processing*, 2002.

[42] M. A. Hossan, S. Memon, and M. A. Gregory, “A novel approach for mfcc feature extraction,” in *2010 4th International Conference on Signal Processing and Communication Systems*, pp. 1–5, Dec 2010.

[43] S. Chu, S. Narayanan, and C. . J. Kuo, “Environmental sound recognition with time–frequency audio features,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, pp. 1142–1158, Aug 2009.

[44] J. Dennis, H. D. Tran, and H. Li, “Spectrogram image feature for sound event classification in mismatched conditions,” *IEEE Signal Processing Letters*, vol. 18, pp. 130–133, Feb 2011.

[45] H. Zhang, I. McLoughlin, and Y. Song, “Robust sound event recognition using convolutional neural networks,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 559–563, April 2015.

[46] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, vol. 1, pp. 886–893, IEEE, 2005.

[47] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, no. 6755, p. 788, 1999.

[48] P. D. O’grady and B. A. Pearlmutter, “Convolutive non-negative matrix factorisation with a sparseness constraint,” in *2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, pp. 427–432, IEEE, 2006.

[49] V. Bisot, R. Serizel, S. Essid, and G. Richard, “Acoustic scene classification with matrix factorization for unsupervised feature learning,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6445–6449, March 2016.

[50] D. M. J. Tax, *One-class classification: Concept learning in the absence of counter-examples.* PhD thesis, Technische Universiteit Delft, 2001.

[51] C. Bellinger, S. Sharma, and N. Japkowicz, “One-class versus binary classification: Which and when?,” in *2012 11th International Conference on Machine Learning and Applications*, vol. 2, pp. 102–106, Dec 2012.

[52] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[53] J. Shawe-Taylor, N. Cristianini, *et al.*, *Kernel methods for pattern analysis.* Cambridge university press, 2004.

[54] D. R. Cox, “The regression analysis of binary sequences,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[55] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[56] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations*, 12 2014.

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[58] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[59] I. McLoughlin, H. Zhang, Z. Xie, Y. Song, and W. Xiao, “Robust sound event classification using deep neural networks,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, pp. 540–552, March 2015.

[60] S. Dieleman and B. Schrauwen, “End-to-end learning for music audio,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6964–6968, IEEE, 2014.

[61] F. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” *Neural computation*, vol. 12, pp. 2451–71, 10 2000.

[62] G. Parascandolo, H. Huttunen, and T. Virtanen, “Recurrent neural networks for polyphonic sound event detection in real life recordings,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6440–6444, March 2016.

[63] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional recurrent neural networks for polyphonic sound event detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 1291–1303, June 2017.

[64] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, “Very deep convolutional neural networks for raw waveforms,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 421–425, March 2017.

[65] X. Zhang, Y. Zou, and W. Shi, “Dilated convolution neural network with leakyrelu for environmental sound classification,” in *2017 22nd International Conference on Digital Signal Processing (DSP)*, pp. 1–5, IEEE, 2017.

[66] J. Salamon, C. Jacoby, and J. P. Bello, “A dataset and taxonomy for urban sound research,” in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 1041–1044, 2014.

[67] H. Phan, L. Hertel, M. Maass, and A. Mertins, “Robust audio event recognition with 1-max pooling convolutional neural networks,” in *17th Annual Conference of the International Speech Communication Association (INTERSPEECH 2016)*, (San Francisco, USA), pp. 3653–3657, ISCA, September 2016.

[68] I. Ozer, Z. Ozer, and O. Findik, “Noise robust sound event classification with convolutional neural network,” *Neurocomputing*, vol. 272, pp. 505–512, 2018.

[69] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, “Acoustic event detection in real life recordings,” in *2010 18th European Signal Processing Conference*, pp. 1267–1271, IEEE, 2010.

[70] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, “Cnn architectures for large-scale audio classification,” in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 131–135, IEEE, 2017.

[71] L. Wang and A. Cavallaro, “Microphone-array ego-noise reduction algorithms for auditory micro aerial vehicles,” *IEEE Sensors Journal*, vol. 17, no. 8, pp. 2447–2455, 2017.

[72] G. Sinibaldi and L. Marino, “Experimental analysis on the noise of propellers for small uav,” *Applied Acoustics*, vol. 74, no. 1, pp. 79 – 88, 2013.

[73] L. Wang and A. Cavallaro, “Ear in the sky: Ego-noise reduction for auditory micro aerial vehicles,” in *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 152–158, Aug 2016.

[74] B. D. Van Veen and K. M. Buckley, “Beamforming: A versatile approach to spatial filtering,” *IEEE assp magazine*, vol. 5, no. 2, pp. 4–24, 1988.

[75] T. Ishiki and M. Kumon, “Design model of microphone arrays for multirotor helicopters,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6143–6148, Sep. 2015.

[76] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent component analysis*, vol. 46. John Wiley & Sons, 2004.

[77] L. Wang, “Multi-band multi-centroid clustering based permutation alignment for frequency-domain blind speech separation,” *Digital Signal Processing*, vol. 31, pp. 79 – 92, 2014.

[78] L. Wang, R. Sanchez-Matilla, and A. Cavallaro, “Tracking a moving sound source from a multi-rotor drone,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2511–2516, Oct 2018.

[79] P. Marmaroli, X. Falourd, and H. Lissek, “A uav motor denoising technique to improve localization of surrounding noisy aircrafts: proof of concept for anti-collision systems,” in *Acoustics 2012*, 2012.

[80] G. Ince, K. Nakadai, and K. Nakamura, “Online learning for template-based multi-channel ego noise estimation,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3282–3287, IEEE, 2012.

[81] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, “Speech enhancement based on deep denoising autoencoder.,” in *Interspeech*, pp. 436–440, 2013.

[82] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780, IEEE, 2017.

[83] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in python,” 2015.

[84] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[85] NVIDIA, P. Vingelmann, and F. H. Fitzek, “Cuda, release: 10.2.89,” 2020.