



TECHNISCHE  
UNIVERSITÄT  
WIEN



Institut für  
Computertechnik  
Institute of  
Computer Technology

A MASTER THESIS ON

# Recurrent Audio Forecasting for Active Noise Cancellation

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF

**Diplom-Ingenieur**

(Equivalent to Master of Science)

in

Embedded Systems (066 504)

by

**Moritz Steinhauser**

01622498

**Supervisor(s):**

Univ.Prof. Dipl.-Ing. Dr.techn. Axel Jantsch

Projektass. Dipl.-Ing. Matthias Bittner, MSc

Vienna, Austria

February 2026



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar  
The approved original version of this thesis is available in print at TU Wien Bibliothek.

# Abstract

Active Noise Cancellation (ANC) suppresses unwanted sound by generating an anti-noise signal that destructively interferes with the disturbance. It is widely used in consumer applications such as headphones and automotive systems. For effective cancellation, the anti-noise must be generated with minimal latency, which imposes strict real-time constraints. This motivates predictive approaches in which the anti-noise is computed based on a short-term prediction of the acoustic signal. However, accurate forecasting under tight timing constraints introduces significant challenges in terms of computational efficiency, particularly for embedded systems with limited hardware resources. Conventional ANC approaches primarily rely on adaptive filters operating in feedback and feedforward configurations, while recent research has explored complex deep learning models. While State-Space Models (SSMs) have been successfully applied in sequence modeling, their use for predictive audio forecasting in ANC systems has not yet been systematically explored. They combine a linear recurrent structure with a complex-valued diagonal parameterization, enabling parallel training and efficient discretization. This design allows constant-memory inference independent of sequence length and supports causal real-time operation on embedded hardware. In this work, we investigate the use of SSMs for audio forecasting to enable real-time anti-noise generation. We show that SSMs can accurately forecast audio signals and achieve low Mean Absolute Error (MAE) and Mean Squared Error (MSE) for the SpeechCommands and ESC-50 datasets. Our analysis further demonstrates strong cross-dataset generalization and shows that the proposed models can operate at different sampling rates without retraining, enabling flexible deployment across a wide range of systems. However, we also identify limitations in prediction accuracy for signals with energy distributed across the entire available bandwidth and practical constraints regarding computational efficiency in strictly resource-limited scenarios.

# Kurzfassung

Aktive Geräuschunterdrückung (Active Noise Cancellation) reduziert unerwünschte Geräusche, indem ein Gegensignal erzeugt wird, das mit dem Störsignal destruktiv interferiert. Derartige Systeme sind weit verbreitet, beispielsweise in Kopfhörern oder in der Fahrzeugakustik. Für eine effektive Unterdrückung muss das Gegensignal mit minimaler Latenz erzeugt werden, wodurch strenge Echtzeitanforderungen an das System gestellt werden. Dies motiviert vorausschauende Ansätze, bei denen das Gegensignal auf Basis einer kurzzeitigen Vorhersage des akustischen Signals berechnet wird. Eine präzise Vorhersage unter diesen zeitlichen Randbedingungen erfordert jedoch eine hohe rechnerische Effizienz, insbesondere bei eingebetteten Systemen mit begrenzten Hardware-Ressourcen. Konventionelle Ansätze zur Geräuschunterdrückung basieren überwiegend auf adaptiven Filtern in Rückkopplungs- und Vorsteuerungsstrukturen. Neuere Arbeiten untersuchen komplexe Modelle des maschinellen Lernens. Zustandsraummodelle wurden zwar bereits erfolgreich zur Sequenzmodellierung eingesetzt, jedoch bislang nicht systematisch für die Vorhersage von Audiosignalen im Kontext der aktiven Geräuschunterdrückung untersucht. Sie kombinieren eine lineare rekurrente Struktur mit einer komplexwertigen diagonalen Parametrierung, wodurch paralleles Training und eine effiziente Diskretisierung ermöglicht werden. Diese Architektur führt zu einem konstanten Speicherbedarf unabhängig von der Sequenzlänge und erlaubt eine kausale Echtzeitverarbeitung auf eingebetteter Hardware. In dieser Arbeit werden Zustandsraummodelle zur Vorhersage von Audiosignalen untersucht, um die Echtzeitberechnung von Gegensignalen zu ermöglichen. Es wird gezeigt, dass Zustandsraummodelle Audiosignale präzise vorhersagen können und niedrige Fehlermetriken für die SpeechCommands und ESC-50 Datensätze erreichen. Die Analyse zeigt zudem eine gute Generalisierungsfähigkeit über verschiedene Datensätze hinweg und belegt, dass die vorgeschlagenen Modelle bei unterschiedlichen Abstraten ohne erneutes Training eingesetzt werden können, was eine flexible Nutzung in einer Vielzahl von Systemen ermöglicht. Gleichzeitig werden Grenzen hinsichtlich der Vorhersagegenauigkeit bei Signalen, deren Energie über das gesamte Frequenzspektrum verteilt ist, sowie Einschränkungen bezüglich der rechnerischen Effizienz unter stark ressourcenbeschränkten Bedingungen aufgezeigt.

## Erklärung

*Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.*

*Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.*

## Copyright Statement

I, Moritz Steinhauser, hereby declare that this thesis is my own original work and, to the best of my knowledge and belief, it does not:

- Breach copyright or other intellectual property rights of a third party.
- Contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
- Contain material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.
- Contain substantial portions of third party copyright material, including but not limited to charts, diagrams, graphs, photographs or maps, or in instances where it does, I have obtained permission to use such material and allow it to be made accessible worldwide via the Internet.

Signature: \_\_\_\_\_

Vienna, Austria, February 2026

Moritz Steinhauser



# Contents

<b>Abstract</b>	<b>iii</b>
<b>Kurzfassung</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Theoretical Background</b>	<b>3</b>
2.1 Fundamentals of Active Noise Cancellation . . . . .	3
2.2 Fundamentals of Digital Filters . . . . .	7
2.3 Forecasting and Autoregressive Modeling . . . . .	9
<b>3 State of the Art</b>	<b>11</b>
3.1 Active Noise Cancellation . . . . .	11
3.2 Raw Audio Generation . . . . .	13
3.3 Deep State-Space Models . . . . .	14
<b>4 Methodology</b>	<b>17</b>
4.1 Datasets . . . . .	18
4.2 Models . . . . .	19
4.3 Evaluation . . . . .	19
4.4 One-Step-Ahead Prediction . . . . .	23
4.5 Zero-Phase Filtering . . . . .	25
<b>5 Experiments</b>	<b>27</b>
5.1 Baseline . . . . .	27
5.2 Sampling Rate Experiments . . . . .	32
5.3 Parallel Approach . . . . .	35
5.4 Zero-Phase Filtering . . . . .	37

5.5 Real-Time Capability . . . . .	40
<b>6 Conclusion</b>	<b>43</b>
<b>Bibliography</b>	<b>45</b>

# List of Tables

4.1	The training parameters used throughout this thesis. . . . .	17
4.2	The table shows an overview of the models and their parameters, which are used in this thesis. . . . .	19
5.1	MAE and MSE for four models trained on a one-step-ahead prediction task. Evaluation is conducted on the SpeechCommands and ESC-50 datasets. Each model is trained on both datasets to assess generalization. . . . .	29
5.2	MAE and MSE for two models trained at 16 kHz and 44.1 kHz. Evaluation is performed on the ESC-50 and SpeechCommands datasets at a sampling rate of 44.1 kHz. The resampling of the datasets is done with the torchaudio library. . . . .	34
5.3	MAE and MSE losses for the parallel model on the SpeechCommands and ESC-50 datasets.	37
5.4	The number of parameters, MACs per sample, and state memory requirements for the three model configurations. A sample in this case corresponds to one datapoint in the audio signal. . . . .	41



# List of Figures

2.1	The figure shows two sinusoids with varying relative phase, depicted in blue and green. The red signal represents their sum and highlights the importance of phase when signals are superimposed. The plot illustrates the transition from constructive to destructive interference. . . . .	4
2.2	The graphic shows a basic feedback loop for ANC systems. The primary and secondary paths are denoted as $P(z)$ and $S(z)$ respectively. The error microphone and loudspeaker are both inside the earcup. The ANC block implements the algorithm that is used to obtain the anti-noise signal. . . . .	5
2.3	A feedforward loop for ANC systems is depicted. The reference microphone is placed outside, while the error microphone and the loudspeaker are placed inside the earcup. $P(z)$ and $S(z)$ model the primary and secondary path of the system. The ANC block implements the specific algorithm to calculate the anti-noise signal from the reference and the error signal. . . . .	5
2.4	The figure illustrates a feedforward ANC system that employs the FxLMS algorithm. $P(z)$ and $S(z)$ represent the primary and secondary acoustic paths, respectively, while $\hat{S}(z)$ denotes the estimated secondary path. $W(z)$ is an adaptive filter that is continuously updated according to the LMS adaptation rule. . . . .	6
4.1	The training and inference stages are depicted. During training the models take sequences of raw audio as input and produce sequences at the output. Backpropagation is used to update the model parameters. During inference the models produce outputs sample by sample. The ideal transfer function has a magnitude of 0 dB and a linear phase response corresponding to a forecast of one sample. . . . .	18
4.2	The architecture of the multi-layer models is illustrated. Each layer implements a S-Edge layer. The last layer does not apply a nonlinearity. The same architecture is used for both the medium and large model, differing only in the number of layers. . . . .	20

4.3	Empirical Bode plot alongside a time-domain representation of input and output signals for a specific frequency. While the transfer function is calculated in the frequency domain using the DFT, the time-domain representation allows an approximate estimation of magnitude and phase. . . . .	22
4.4	The figure shows the calculation of the ground truth signals for the prediction task on the left, which represents a time shift, and the filtering task on the right, which in this case is a low-pass filtered signal with a cut-off frequency of 2 kHz. The original signals are depicted in blue, the ground truth signals in purple. . . . .	23
4.5	The figure illustrates a parallel approach for the one-step-ahead prediction task. The input signal is split into four frequency bands using zero-phase filtering. Four prediction models are trained on the respective bands. The sum of the outputs of the models represents the final output signal, which is the prediction of the overall system. . . . .	24
4.6	Magnitude and phase responses of the ideal zero-phase filters. The Python function <code>filtfilt</code> is used to generate the curves by applying the filter in forward and backward direction. Each filter has a passband width of 2 kHz. . . . .	26
5.1	Magnitude and phase responses of the three models trained for one-step-ahead prediction on the SpeechCommands dataset. Additionally, the ideal magnitude and phase responses are depicted. The M and L models achieve the desired behavior with slight deviations from the ideal responses, while the S model, lacking nonlinearities, fails to model the complex dynamics of the task. . . . .	28
5.2	Predicted and target waveforms for two time-series segments of the word <i>right</i> , including the corresponding error signals. The left plot resembles a low frequency section of the signal, the right plot a high frequency section. . . . .	30
5.3	Frequency spectra of target signals and model outputs for two words from the SpeechCommands dataset, including the spectra of the corresponding error signals. . . . .	31
5.4	This figure depicts the magnitude and phase responses of two models trained at sampling rates of 16 kHz and 44.1 kHz. . . . .	33
5.5	The normalized frequency spectra of an audio sample from the ESC-50 dataset is shown for two different sampling rates, 16 kHz and 44.1 kHz. The data is resampled with the <code>torchaudio</code> library. . . . .	33
5.6	The ESR across frequencies for two different models for the SpeechCommands dataset is shown. The 16 kHz model is evaluated at both frequencies, while the 44.1 kHz model is only evaluated at 44.1 kHz. . . . .	34

5.7	Magnitude and phase responses of the parallel model. Each curve corresponds to one input–output pair trained on a specific frequency band. Outside the respective bands the models have learned high gain and nonlinear phase responses. . . . .	36
5.8	The ESR across frequencies, evaluated on the SpeechCommands dataset, is shown for both the baseline and the parallel models. The parallel model, although not real-time capable, achieves the lowest overall error. . . . .	36
5.9	Bode plots of the four outputs of the zero-phase filtering model. The magnitude responses show clear lowpass, bandpass and highpass behavior, while the phase responses deviate from ideal zero-phase filtering. . . . .	38
5.10	THD ratio and absolute values for the four outputs of the zero-phase filtering model. The THD increases with frequency and in some cases exceeds the main frequency component outside the trained bands. . . . .	39
5.11	Average attenuation on the SpeechCommands dataset for the four outputs of the zero-phase filtering model. . . . .	40



# Acronyms

**ANC** Active Noise Cancellation. iii, xi, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 17, 25, 27, 29, 40, 41, 43, 44

**AR** Autoregressive. 9

**ARMA** Autoregressive Moving Average. 9

**BPTT** Backpropagation Through Time. 14

**CNN** Convolutional Neural Network. 12, 13

**CRN** Convolutional Recurrent Network. 12

**DDPG** Deep Deterministic Policy Gradient. 12

**DSP** Digital Signal Processor. 12

**ESR** Error-to-Signal Ratio. xii, xiii, 34, 35, 36

**FIR** Finite Impulse Response. 7, 8

**FPGA** Field Programmable Gate Array. 12, 13

**FxLMS** Filtered-x Least Mean Square. xi, 1, 4, 6, 11, 12, 13

**IIR** Infinite Impulse Response. 7, 8, 25

**LMS** Least Mean Square. xi, 4, 6

**LTl** Linear Time-Invariant. 8, 9

**MA** Moving Average. 9

**MAC** Multiply-Accumulate. ix, 40, 41

**MAE** Mean Absolute Error. iii, ix, 13, 29, 34, 35, 37

**MIMO** Multiple-Input, Multiple-Output. 8, 14, 15

**MLP** Multilayer Perceptron. 13

**MSE** Mean Squared Error. iii, ix, 29, 34, 35, 37

**NLL** Negative Log-Likelihood. 15

- NMSE** Normalized Mean Square Error. 12
- RL** Reinforcement Learning. 12
- RMS** Root Mean Square. 22
- RNN** Recurrent Neural Network. 9, 13, 14
- SISO** Single-Input, Single-Output. 14, 15
- SOC** System-On-Chip. 40, 41
- SSM** State-Space Model. iii, 2, 8, 9, 11, 14, 15, 16, 17, 20, 25, 27, 32, 37, 43
- THD** Total Harmonic Distortion. xiii, 20, 21, 22, 37, 39
- THF** Tangential Hyperbolic Function. 12
- TTS** Text-To-Speech. 14
- ZOH** Zero-Order Hold. 8, 15

# Chapter 1

## Introduction

Active Noise Cancellation (ANC), often also referred to as Active Noise Control, has gained significant attention in recent decades. The fundamental idea of actively reducing unwanted sound is first introduced by P. Lueg in 1936, laying the theoretical foundation for modern ANC systems [1]. Since then, substantial technological progress has been made, particularly due to advances in digital signal processing and embedded electronics.

Today, ANC is most prominently found in consumer products such as headphones, but it is also widely used in applications like automotive noise reduction and aircraft cabin noise control. Especially in headphones, ANC systems must operate under strict constraints. These include low power consumption, limited computational resources, and very small allowable delays to ensure real-time operation. As a result, the design of effective ANC algorithms for such embedded systems poses a challenging task.

Digital filtering techniques form the core of most ANC implementations. Fixed filtering methods have the advantage of low computational complexity but lack adaptability to changing noise environments. Adaptive filtering methods, on the other hand, can adjust their parameters in real-time to better cancel varying noise. The Filtered-x Least Mean Square (FxLMS) algorithm is a widely used adaptive filtering technique. While these methods are well understood and computationally efficient, their performance is limited in more complex acoustic environments. In particular, cancelling higher-frequency noise remains difficult due to the reactive behavior of the filters. Furthermore, rapidly changing noise environments can degrade the performance of adaptive filters, as they may not converge quickly enough to track the changes [2].

More recently, artificial intelligence and machine learning methods gain increasing attention in ANC research. These approaches offer the potential to model complex and time-varying acoustic environments more accurately and to create proactive systems that show predictive behavior. However,

their applicability in real-world ANC systems is often constrained by the available computational power and energy budget, especially in battery-powered devices like headphones.

To address these challenges, this thesis explores the use of autoregressive modeling with recurrent State-Space Models (SSMs), as they were used in [3] and [4], for real-time audio sample prediction. The goal is to develop a forecasting model that predicts future audio samples based on past observations, enabling proactive noise cancellation. By accurately forecasting upcoming noise, the ANC system can generate anti-noise signals in advance, potentially improving cancellation performance, especially for higher-frequency noise components.

Therefore, this work seeks to address the following research questions:

1. Can autoregressive modeling with recurrent SSMs accurately predict future audio samples?
2. Is it feasible to use a prediction-based model for ANC in headphones, particularly in terms of implementation within hardware and real-time constraints?

The first research question aims to assess whether autoregressive SSMs can efficiently predict audio patterns. In particular, the following sub-questions are investigated:

- How sensitive is the prediction accuracy to different types of audio, such as environmental sounds and speech?
- How effectively does the model handle high-frequency content, which is typically more challenging to cancel in ANC systems?
- What are the effects of model hyperparameters (e.g., state size, number of layers) on the forecasting accuracy?
- Can the prediction model adapt to changes in noise characteristics over time, such as impulsive or rapidly shifting noise?

The second research question focuses on the practical application of this prediction model in an ANC system. This aspect is analyzed in a theoretical manner by considering the model size, computational requirements, and latency constraints. As a consequence, this work can be seen as a first step towards integrating advanced forecasting models into real-world ANC systems.

The remainder of this thesis is structured as follows: Chapter 2 provides theoretical background on ANC systems, digital filters, forecasting and autoregressive modeling techniques. Chapter 3 covers related work on ANC, raw audio generation and deep SSMs. Chapter 4 describes the methodology used to develop and evaluate the proposed forecasting models. Chapter 5 presents the experimental results and analysis of the forecasting models. Chapter 6 concludes the thesis by summarizing the findings and outlining directions for future work.

## Chapter 2

# Theoretical Background

The performance of ANC is highly dependent on the physical system and the algorithms used. Therefore, the fundamentals of ANC systems and common algorithms are described in the first section. Since digital filters are widely used to implement ANC controllers, the second section covers the basics of digital filters and their representations. Finally, the third section introduces forecasting and autoregressive modeling, which are essential concepts for the methods developed in this thesis.

### 2.1 Fundamentals of Active Noise Cancellation

The main objective of ANC is to eliminate or at least attenuate undesired noise. Applications for ANC include headphones, automotive systems, industrial machinery and airplane cabins. The underlying mechanism used to achieve this effect is known as destructive interference. It is first demonstrated in [1] that sound waves can be cancelled by generating an identical waveform with inverted phase. This inverted signal is referred to as anti-noise. Figure 2.1 illustrates this phenomenon.

ANC is only effective if the anti-noise can be produced before the original sound wave reaches the point where the noise is intended to be cancelled. Consequently, the system must be able to anticipate the incoming sound wave.

#### 2.1.1 Systems

In ANC systems, two primary control strategies are usually employed: feedback and feedforward systems. In the following, these two approaches are described and visualized based on a headphone application.

Feedback ANC systems, as displayed in Figure 2.2, use a single microphone positioned inside the earcup to detect and react to the residual noise in real time. These systems operate without relying on any external reference signal and instead focus solely on the noise present within the earcup. By

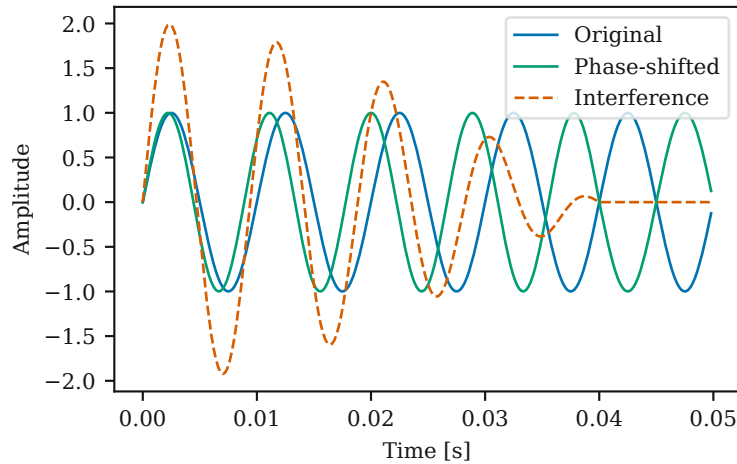


Figure 2.1: The figure shows two sinusoids with varying relative phase, depicted in blue and green. The red signal represents their sum and highlights the importance of phase when signals are superimposed. The plot illustrates the transition from constructive to destructive interference.

detecting the internal noise, the feedback system generates an anti-noise signal to cancel it. However, feedback systems face several challenges. Stability issues can arise, particularly when dealing with higher-frequency noise, which can lead to inconsistencies in noise reduction. Additionally, feedback systems are limited by the *waterbed effect*, a phenomenon that makes it difficult to achieve uniform noise cancellation across all frequencies. Essentially, reducing noise in one frequency band may cause an increase in noise in another, limiting the systems' overall effectiveness [2].

In contrast, feedforward ANC systems, as depicted in Figure 2.3, utilize an external reference microphone positioned outside the earcup to capture noise before it reaches the ear. Inside the earcup, an error microphone measures the residual noise, and the controller processes both the reference and error signals to produce the anti-noise output. Feedforward systems are generally more effective at cancelling noise across a broader frequency range. However, they can struggle with rapidly changing or high-frequency noise, as these noise types are harder to track and cancel effectively [2, 5].

### 2.1.2 Algorithms

Most ANC systems rely on feedforward control strategies, since feedback systems are subject to stability limitations caused by delays in the acoustic feedback path [2,6]. The most common algorithm used is the FxLMS algorithm. It is based on the Least Mean Square (LMS) algorithm, which adapts filter coefficients to minimize a given error signal. In the context of ANC, this error is the residual noise that reaches the listener's ear. The goal of the algorithm is to minimize this error signal by adapting the filter weights  $w$  according to the following update rule:

$$w_{k+1} = w_k + \mu x_k e_k, \quad (2.1)$$

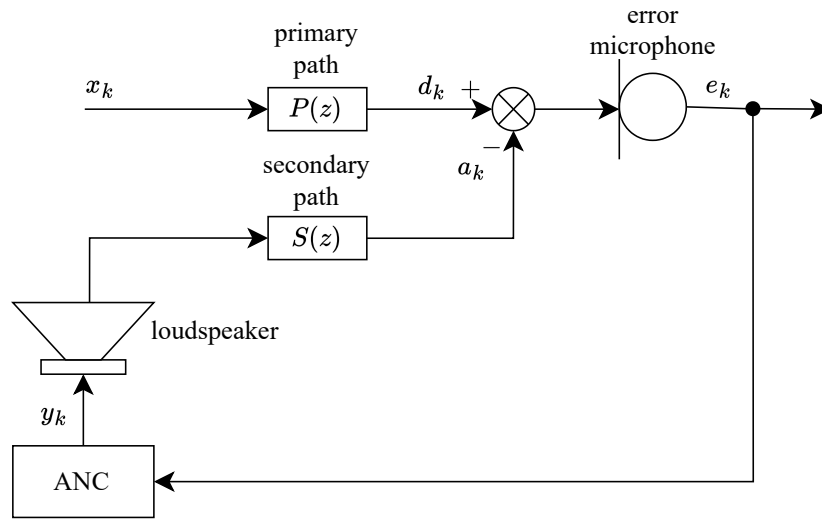


Figure 2.2: The graphic shows a basic feedback loop for ANC systems. The primary and secondary paths are denoted as  $P(z)$  and  $S(z)$  respectively. The error microphone and loudspeaker are both inside the earcup. The ANC block implements the algorithm that is used to obtain the anti-noise signal.

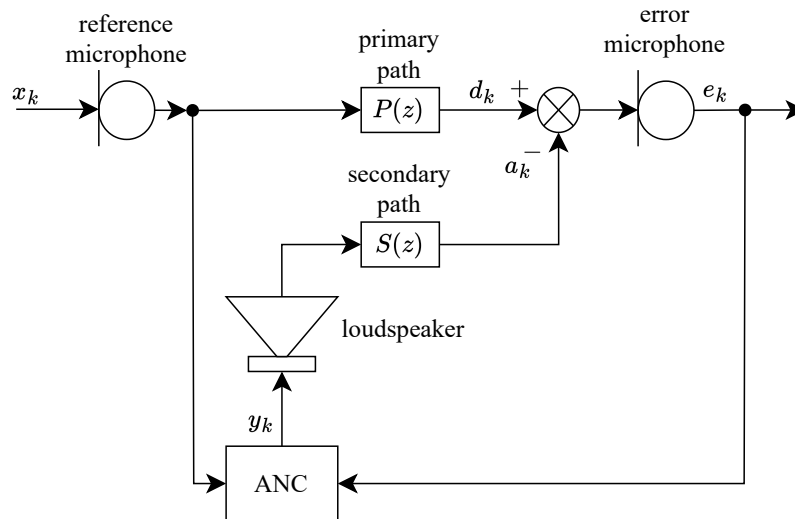


Figure 2.3: A feedforward loop for ANC systems is depicted. The reference microphone is placed outside, while the error microphone and the loudspeaker are placed inside the earcup.  $P(z)$  and  $S(z)$  model the primary and secondary path of the system. The ANC block implements the specific algorithm to calculate the anti-noise signal from the reference and the error signal.

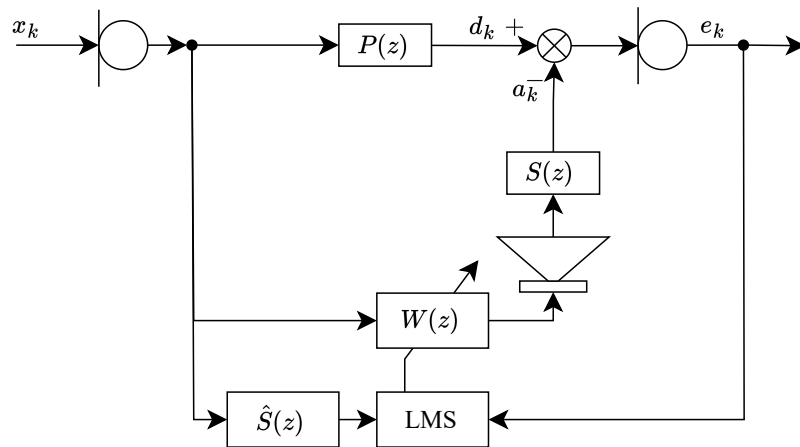


Figure 2.4: The figure illustrates a feedforward ANC system that employs the FxLMS algorithm.  $P(z)$  and  $S(z)$  represent the primary and secondary acoustic paths, respectively, while  $\hat{S}(z)$  denotes the estimated secondary path.  $W(z)$  is an adaptive filter that is continuously updated according to the LMS adaptation rule.

where  $\mu$  is the learning rate,  $x_k$  is the reference signal, and  $e_k$  is the error signal at time step  $k$ .

When using LMS in a feedforward ANC system (as illustrated in Figure 2.4), the adaptive filter must also compensate for the secondary path, which is the acoustic path from the speaker to the error microphone. In practice, this would require the filter to learn the inverse of the secondary path  $S^{-1}(z)$ , which is often very difficult or even impossible to achieve in real time.

To overcome this problem, the FxLMS algorithm introduces an additional filter that estimates the secondary path. This estimated filter is applied to the reference signal before it is used in the LMS update. As a result, the influence of the secondary path is effectively removed from the adaptation process, leading to improved stability and faster convergence. The LMS update rule is then modified as follows:

$$w_{k+1} = w_k + \mu x'_k e_k, \quad (2.2)$$

where  $x'_k = \hat{s}_k * x_k$  is the filtered reference signal, and  $\hat{s}_k$  represents the estimated impulse response of the secondary path [2, 6–9].

### 2.1.3 Challenges

With such systems and algorithms, several challenges arise. From a physical perspective, variations in the primary and secondary paths can occur due to factors such as headphone movement or differences in ear geometry. These changes particularly affect time-invariant approaches, where a fixed filter is

used, and can also introduce a risk of instability in feedback systems. Moreover, the direction of arrival of the noise source influences the performance of ANC systems and must therefore be considered during the design process.

Latencies in the primary and secondary paths play a crucial role as well. Ideally, the primary path should exhibit a higher latency, while the secondary path should have a lower one. This allows the system to react to the incoming noise in time, ensuring that destructive interference occurs precisely at the target location.

In general, the algorithm must operate at high processing speed, as the system needs to respond to noise as quickly as possible. If the anti-noise cannot be computed within the required time frame, the attenuation performance degrades rapidly. Furthermore, computational complexity and power consumption are critical design factors, particularly for portable applications such as headphones [2, 9].

## 2.2 Fundamentals of Digital Filters

In practice, the controller in an ANC system is usually implemented as a digital filter. Two classes of discrete-time filters can be distinguished: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. The following subsections summarize their key properties and relevance for ANC applications.

### 2.2.1 FIR Filter

FIR filters are constructed from systems that do not include recursion or feedback, and are therefore referred to as feedforward systems. They are described by

$$y_k = \sum_{i=0}^N \tilde{b}_i u_{k-i}, \quad (2.3)$$

where  $y_k$  is the output and  $u_k$  is the input at discrete time  $k$ .  $\tilde{b}_i$  are the filter coefficients. As the name suggests, FIR filters have a finite impulse response and are always stable, which is a major advantage. Their simple structure makes them easy to design and implement. However, achieving a desired filter behavior often requires a high filter order, meaning many filter coefficients are needed [10, 11].

### 2.2.2 IIR Filter

IIR filters on the other hand do not have a finite impulse response and are therefore not necessarily stable. They include recursive elements, meaning that the output depends not only on the current and previous inputs but also on previous outputs. The general difference equation for an IIR filter is given

by

$$y_k = \sum_{i=0}^N b_i u_{k-i} - \sum_{j=1}^M a_j y_{k-j}, \quad (2.4)$$

where  $y_k$  is the output,  $u_k$  is the input,  $b_i$  and  $a_j$  are the filter coefficients,  $N$  is the feedforward order, and  $M$  is the feedback order. The presence of feedback (the  $a_j$  terms) allows IIR filters to achieve a desired frequency response with fewer coefficients compared to FIR filters, making them computationally efficient. However, the recursive nature can lead to instability if the filter coefficients are not chosen carefully. Additionally, IIR filters generally do not have linear phase, which can introduce phase distortion [10, 11].

### 2.2.3 State-Space Models

Both FIR and IIR filters can also be represented in state-space form, which provides a more general and compact description of Linear Time-Invariant (LTI) systems. The state-space formulation expresses the same dynamics as the difference equations 2.3 and 2.4 through internal states that evolve over time. This representation is mathematically equivalent but is often preferred in control theory and modern signal processing because it facilitates system analysis, stability assessment, and implementation of Multiple-Input, Multiple-Output (MIMO) systems.

A continuous-time linear SSM can be described by the following equations:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad (2.5)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \quad (2.6)$$

where  $\mathbf{A}$  denotes the state matrix,  $\mathbf{B}$  represents the input matrix,  $\mathbf{C}$  is the output matrix, and  $\mathbf{D}$  indicates the feedthrough matrix. Such systems are frequently used in control theory with the objective to model physical systems. The system defines a sequence-to-sequence transformation, which maps the input  $\mathbf{u}(t)$  onto the output  $\mathbf{y}(t)$  based on the hidden state  $\mathbf{x}(t)$ .

When discretized, for example with the Zero-Order Hold (ZOH) method, the system can be represented by the following equations:

$$\mathbf{x}_k = \mathbf{A}_d \mathbf{x}_{k-1} + \mathbf{B}_d \mathbf{u}_k, \quad (2.7)$$

$$\mathbf{y}_k = \mathbf{C}_d \mathbf{x}_k + \mathbf{D}_d \mathbf{u}_k. \quad (2.8)$$

In these equations, the matrices  $\mathbf{A}_d$  and  $\mathbf{B}_d$  are functions of the original matrices  $\mathbf{A}$  and  $\mathbf{B}$  and the

sampling time  $T_s$ :

$$\mathbf{A}_d = \exp(\mathbf{A}T_s), \quad \mathbf{B}_d = \mathbf{A}^{-1}(\mathbf{A}_d - \mathbf{I})\mathbf{B}, \quad \mathbf{C}_d = \mathbf{C}, \quad \mathbf{D}_d = \mathbf{D}. \quad (2.9)$$

To analyze the input and output behavior of such systems, the transfer function can be calculated. The following equations represent the analytical calculation for the continuous and discrete version:

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}, \quad \mathbf{G}(z) = \mathbf{C}_d(z\mathbf{I} - \mathbf{A}_d)^{-1}\mathbf{B}_d + \mathbf{D}_d, \quad (2.10)$$

with  $s = j\omega$  and  $z = e^{sT_s}$ . The complex unit is denoted by  $j$  whereas  $\omega$  represents the angular frequency.

The transfer functions can be used to analyze the frequency response of the system, which is one of the most important characteristics of LTI systems [12–16].

## 2.3 Forecasting and Autoregressive Modeling

Forecasting or prediction describes a process of estimating future values in a series of values based on a set of input values. Throughout this work, time series are assumed to be sampled at equidistant intervals. The process is thereby not restricted to a certain structure or condition. For example it could predict the value  $x_{k+l}$  at discrete time  $k$ , with a lead time  $l$  based on arbitrary inputs, like the past values and also external inputs.

Autoregressive modeling denotes a special case of forecasting in which future values are generated sample by sample, using only past samples of the same signal. This induces a causal structure,

$$\hat{x}_k = f(x_{k-1}, x_{k-2}, \dots),$$

such that no future information is available and the model's output at each step becomes the input for the next. Classical statistical approaches formalize this principle through models such as Autoregressive (AR), Moving Average (MA), and Autoregressive Moving Average (ARMA) models, where the signal is treated as a stochastic process and the next sample is expressed as a linear combination of past values and optionally noise terms.

In machine learning, the term autoregressive is used in a broader but conceptually related sense: models are executed one step at a time, feeding the previously generated output back into the model to obtain the next prediction. This pattern is shared across sequence models such as Recurrent Neural Networks (RNNs), Transformers, and modern SSMs. While these models are not necessarily based on stochastic assumptions, they implement the same computational principle.

For applications such as ANC, this causal online execution is essential since the system must produce predictions with minimal delay and cannot rely on future samples. Autoregressive formulations therefore provide a natural bridge between traditional signal processing and modern neural forecasting architectures used in this work [17, 18].

## Chapter 3

# State of the Art

ANC in small consumer devices, such as in-ear headphones, faces several practical challenges, such as limited available processing time, nonlinear acoustic paths, and changing environment. Classical approaches like the FxLMS algorithm are still widely used, but they reach their limits in these situations.

At the same time, deep learning has made strong progress in modeling raw audio signals. Autoregressive models can learn temporal structures in a causal manner, making them suitable candidates for forecasting future audio samples. This idea of predicting incoming noise, rather than only reacting to it, creates a potential advantage for ANC systems, especially under tight latency constraints.

This chapter gives an overview of related work in three areas relevant to this thesis: traditional and learning-based ANC methods, raw audio generation, which provides the conceptual basis for the forecasting idea, and deep SSMs, which offer efficient and hardware-friendly architectures for sequence modeling.

### 3.1 Active Noise Cancellation

The algorithm most commonly used for ANC is the FxLMS algorithm, which is usually applied in feed-forward or hybrid ANC systems. This adaptive algorithm updates its filter coefficients in real-time, adjusting to changes in the acoustic environment. Despite its effectiveness, the FxLMS algorithm has notable limitations. First of all, the standard algorithm is linear. Therefore it cannot model nonlinearities that are present in the audio signals. Nonlinearities can be introduced through loudspeakers or microphones for example. Another issue is its slow convergence, particularly when dealing with complex signals that exhibit a high spread of eigenvalues, which can hinder rapid adaptation. Another significant drawback is the reliance on accurate secondary path estimation. If the secondary path is not correctly estimated or changes during operation, the controller may fail to effectively cancel the noise, reducing the overall performance of the system [2, 6].

To address the limitations of the FxLMS algorithm, several alternative strategies have been developed, including the use of nonlinear systems, online secondary path estimation, and deep learning-based approaches. In particular, deep learning has gained attention for its ability to model complex, nonlinear relationships and adapt to dynamic environments. In the following, some notable deep learning approaches for ANC are presented [2, 19].

In [5], a deep Convolutional Recurrent Network (CRN) is developed to directly generate the anti-noise signal from the reference signal. The model is trained using the ideal anti-noise signal, shifted by a few samples, allowing it to predict upcoming anti-noise samples based on the reference signal. Evaluation is conducted using an experimental setup in a room measuring  $3\text{m} \times 4\text{m} \times 2\text{m}$ . The performance of the approach is assessed using the Normalized Mean Square Error (NMSE) metric and power spectral analysis, and compared to the standard FxLMS algorithm and its nonlinear extension, the Tangential Hyperbolic Function (THF) - FxLMS. The CRN model demonstrates superior noise attenuation across a broader frequency range and handles nonlinearities in the secondary path, particularly those introduced by loudspeakers, more effectively. Despite these promising results, the main limitation of this approach lies in its high computational complexity, which restricts its feasibility for real-time applications in embedded devices like headphones.

Reinforcement Learning (RL) demonstrates promising results in [19]. In this study, the secondary path is not explicitly modeled and a simple feedback loop is used. A Deep Deterministic Policy Gradient (DDPG) agent is trained to minimize the error signal, allowing the system to adapt to changes in the secondary path within just a few seconds. The RL algorithm is executed on a high-performance CPU, while the acoustic control is implemented on a Digital Signal Processor (DSP). Evaluation is conducted both through simulation and in an experiment within a duct ANC system. The experimental setup involves one loudspeaker simulating the primary noise source and two alternately used loudspeakers for cancelling the noise and an error microphone at the end of the tube. While the FxLMS algorithm still outperforms the RL-based system in terms of noise attenuation under a fixed secondary path, the simplicity and adaptability of this approach to environmental changes are key advantages. Notably, the system is able to adapt to switching between the two loudspeakers within a few seconds and thereby adapting to the specific secondary paths and their nonlinearities. However, as with the previous method, the main limitation of this approach is its high computational complexity. Furthermore, the work focuses only on narrowband noise, while applicability for broadband noise suppression is not addressed.

In [20], a 10-layer dilated Convolutional Neural Network (CNN) is implemented on a Field Programmable Gate Array (FPGA) to solve the causality constraint and to enhance the prediction of future noise and thereby noise attenuation. Compared to FxLMS, the CNN model observes a substantially

larger receptive field, enabling more accurate estimation of upcoming noise samples. The network is trained offline using 10-second noise segments that combines ambient noise and high-frequency sine waves, with the Mean Absolute Error (MAE) of the residual noise serving as the loss function. While inference is performed in real-time on the FPGA, training is conducted on a computer. The proposed system achieves a maximum power reduction of 24 dB, outperforming the conventional FxLMS algorithm (15 dB), and a noise attenuation bandwidth of 2000 Hz compared to 1500 Hz for FxLMS. However, since the model is not adaptive to changing acoustic conditions, its performance may degrade in dynamic environments. Furthermore, the system is tightly coupled to the specific FPGA hardware implementation and cannot be easily transferred to other platforms.

In [21], DNoiseNet, a feedback-based approach for active noise control, is proposed. The controller employs a neural network architecture that combines causal and dilated convolutions, nonlinearities, RNNs, and fully connected layers. While the convolutional layers extract representative features from the raw audio signals, the RNN component is responsible for predicting the anti-noise signal. To compensate for secondary path effects, an additional Multilayer Perceptron (MLP) is trained. DNoiseNet operates in a sample-by-sample manner, allowing real-time generation of the anti-noise signal. The model is evaluated on noise recordings from construction sites, vehicles, and airplane cockpits and demonstrates superior performance compared to all other ANC methods examined in the study, including a traditional FxLMS algorithm and several deep learning-based approaches. These results underline the potential of deep learning in ANC applications and highlight the advantages of incorporating neural feedback-loop architectures. However, as the focus of this work is on room acoustic noise, where device performance is less constrained, the applicability of this approach to small consumer devices with strict latency and computational limitations remains uncertain.

## 3.2 Raw Audio Generation

Similar to the discussed deep learning approaches, where systems learn to predict the anti-noise signal, this thesis will implement an audio forecasting system. The next section will explore the state of the art in raw audio generation, which provides the conceptual foundation for the forecasting approach used in this work.

WaveNet is one of the pioneering approaches to target neural autoregressive generation of raw audio signals [22]. The model uses CNNs to estimate the probability distribution of the next sample in an audio sequence. To ensure that the model only depends on past values, causal convolutions are employed. This design enforces the necessary temporal order in audio generation. While training can be parallelized, inference remains sequential, as each new sample depends on previously generated

samples. WaveNet is evaluated on three tasks: speech generation, Text-To-Speech (TTS), and music generation. It demonstrates an impressive ability to model speech from 109 different speakers, capturing not only the linguistic content but also other acoustic details such as recording quality, breathing, and mouth movements. Although the model sometimes produces non-existent words, the generated speech sounds human-like. However, when applied to audio generation, one of the primary limitations is the model's receptive field, which is not large enough to maintain global coherence over extended sequences. Expanding the receptive field without significantly increasing computational cost remains a central difficulty.

In contrast to WaveNet, RNNs are employed in SampleRNN to model long-term dependencies in audio signals [23]. RNNs are well-suited for capturing such dependencies, but modeling raw audio remains particularly challenging due to the need to handle both short-term correlations between neighboring samples and long-term dependencies over thousands of samples. SampleRNN addresses this issue by using multiple layers that operate at different temporal resolutions, allowing the model to capture both fine-grained and broader patterns in the audio sequence. To reduce the high computational cost associated with training RNNs, truncated Backpropagation Through Time (BPTT) is used. This technique involves splitting long training sequences into shorter subsequences, which speeds up the training process while still allowing the model to learn long-term dependencies. SampleRNN is evaluated on three datasets covering speech, music, and onomatopoeia, and it is compared to the previously discussed WaveNet approach. Human evaluation reveals a preference for SampleRNN over WaveNet, indicating its superior ability to generate natural-sounding audio. The main limitation of this approach is its lack of computational efficiency, as the sequential nature of RNNs makes it difficult to parallelize training.

### 3.3 Deep State-Space Models

Recently, SSMs have gained significant attention in the context of deep learning for tasks that require modeling long-range dependencies. One of the cornerstone works is presented in [12] and introduces the S4 layer, which is a system of Single-Input, Single-Output (SISO) SSMs combined to create a MIMO system. It is trained in the frequency domain and the state matrix is initialized as a diagonal plus low-rank matrix. In [13] the S4D layer is shown, which is based on a diagonal state matrix and simpler and more efficient calculations.

In [18] a framework called SaShiMi is developed. SaShiMi leverages S4 layers, which can be computed using convolutions for efficient training, and recurrent operations for fast inference. Similar to SampleRNN, SaShiMi employs a multi-scale architecture, processing audio signals at different temporal

resolutions. The model operates in a tiered structure: the first tier samples audio at the original frequency, while lower tiers sample at progressively lower frequencies and then upsample their outputs. This approach allows the model to capture both fine-grained and long-range temporal dependencies in audio signals. The model is evaluated on three datasets, covering both music and speech, with comparisons to WaveNet and SampleRNN. The Negative Log-Likelihood (NLL) is used as a primary metric to quantitatively assess performance across models. SaShiMi outperforms both WaveNet and SampleRNN in all tasks. For speech generation, additional quantitative metrics are also employed to assess the models' performance, further highlighting SaShiMi's advantages.

The S5 layer, introduced in [14], implements a diagonal MIMO SSM instead of multiple SISO systems. This design enables the use of parallel scans to efficiently compute the linear recurrence in the time domain. The S5 layer operates exclusively in the time domain and therefore does not require explicit kernel computations in the frequency domain.

The S-Edge layer is developed in [3]. The initial work focuses on audio classification but the S-Edge layer has also proven valuable for other tasks like in [4] and [24], where it is used for raw piano audio synthesis. The S-Edge layer is based on the S5 layer but implements a few notable improvements, which are dynamic input and output shapes, input and output biases and a custom C++17 implementation.

S-Edge is modeled as a complex-valued MIMO system with input bias  $\tilde{\mathbf{b}}$  and output bias  $\tilde{\mathbf{c}}$  and the following state and output equations:

$$\frac{d\tilde{\mathbf{x}}(t)}{dt} = \tilde{\mathbf{\Lambda}}\tilde{\mathbf{x}}(t) + \tilde{\mathbf{B}}\mathbf{u}(t) + \tilde{\mathbf{b}}, \quad (3.1)$$

$$\mathbf{y}(t) = \Re(\tilde{\mathbf{C}}\tilde{\mathbf{x}}(t) + \tilde{\mathbf{c}}). \quad (3.2)$$

The input matrix  $\tilde{\mathbf{B}}$ , the output matrix  $\tilde{\mathbf{C}}$  as well as the biases and the state  $\tilde{\mathbf{x}}$  are complex-valued. The state matrix  $\tilde{\mathbf{\Lambda}} = \text{diag}(\tilde{\boldsymbol{\lambda}})$  is in diagonal form with the complex eigenvalues  $\tilde{\boldsymbol{\lambda}}$ . The input  $\mathbf{u}(t)$  and the output  $\mathbf{y}(t)$  are real-valued. Using ZOH the following discretized system can be derived:

$$\tilde{\mathbf{x}}_k = \tilde{\mathbf{\Lambda}}_d \tilde{\mathbf{x}}_{k-1} + \tilde{\mathbf{B}}_d \mathbf{u}_k + \tilde{\mathbf{b}}_d, \quad (3.3)$$

$$\mathbf{y}_k = \Re(\tilde{\mathbf{C}}_d \tilde{\mathbf{x}}_k + \tilde{\mathbf{c}}_d). \quad (3.4)$$

Here,  $\tilde{\mathbf{\Lambda}}_d = \text{diag}(\tilde{\boldsymbol{\lambda}}_d)$  with  $\tilde{\boldsymbol{\lambda}}_d = e^{\boldsymbol{\Delta} \circ \tilde{\boldsymbol{\lambda}} T_s}$ .  $\boldsymbol{\Delta}$  is a learnable parameter that scales the eigenvalues and has the same dimension as the hidden state.  $T_s$  is a scalar value that represents the sampling rate of the discrete system. The discrete input matrix is given as  $\tilde{\mathbf{B}}_d = \text{diag}(\tilde{\boldsymbol{\lambda}}^{-1} \circ (\tilde{\boldsymbol{\lambda}}_d - 1)) \tilde{\mathbf{B}}$ . The output matrix and the biases remain unchanged during discretization, i.e.,  $\tilde{\mathbf{C}}_d = \tilde{\mathbf{C}}$ ,  $\tilde{\mathbf{b}}_d = \tilde{\mathbf{b}}$  and  $\tilde{\mathbf{c}}_d = \tilde{\mathbf{c}}$ .

The final output of the S-Edge layer is:  $\text{Output} = \text{Skip}(\mathbf{u}_k) + \text{LeakyReLU}(\mathbf{y}_k)$ . The learnable

parameters of the S-Edge layer include  $\tilde{\lambda}$ ,  $\tilde{\mathbf{B}}$  and  $\tilde{\mathbf{b}}$ ,  $\tilde{\mathbf{C}}$  and  $\tilde{\mathbf{c}}$ ,  $\Delta$  and the skip connection. The skip connection is implemented as a fully connected layer that maps the input to the output dimension. While showing promising results on audio classification, Bittner and Schnöll et al. also provide valuable insights from control theory to successfully understand and analyze SSMs [3].

# Chapter 4

## Methodology

This chapter describes the methodological framework used to investigate the suitability of deep SSMs for one-step-ahead prediction of raw audio waveforms in the context of ANC. Building on the theoretical background and the analysis of related work presented in the previous chapters, the focus lies on the practical realization, training, and evaluation of the proposed models.

The central objective is to assess whether recurrent state-space-based architectures are capable of accurately predicting future audio samples in a causal and sample-wise manner, while still allowing for efficient training on long audio sequences. Therefore, models based on the S-Edge layer are trained and evaluated on different types of audio data, including speech and environmental sounds, which exhibit distinct spectral and temporal characteristics.

A key aspect of the proposed approach is the distinction between training and inference. During training, the models operate on complete audio sequences and are optimized using parallel scan techniques to enable efficient backpropagation. During inference, however, the models are deployed in an autoregressive configuration, processing one audio sample at a time, which reflects the constraints of real-time ANC applications. Figure 4.1 illustrates this separation between training and inference, as well as the idealized input–output behavior of a one-step-ahead predictor.

Table 4.1 lists the common training parameters used for the experiments in this thesis. These parameters are chosen based on preliminary experiments to ensure stable and efficient training across different datasets and model architectures.

Table 4.1: The training parameters used throughout this thesis.

sampling rate	16 kHz
batch size	32
optimizer	AdamW
learning rate	$1e-3$
learning rate scheduler	ReduceLROnPlateau

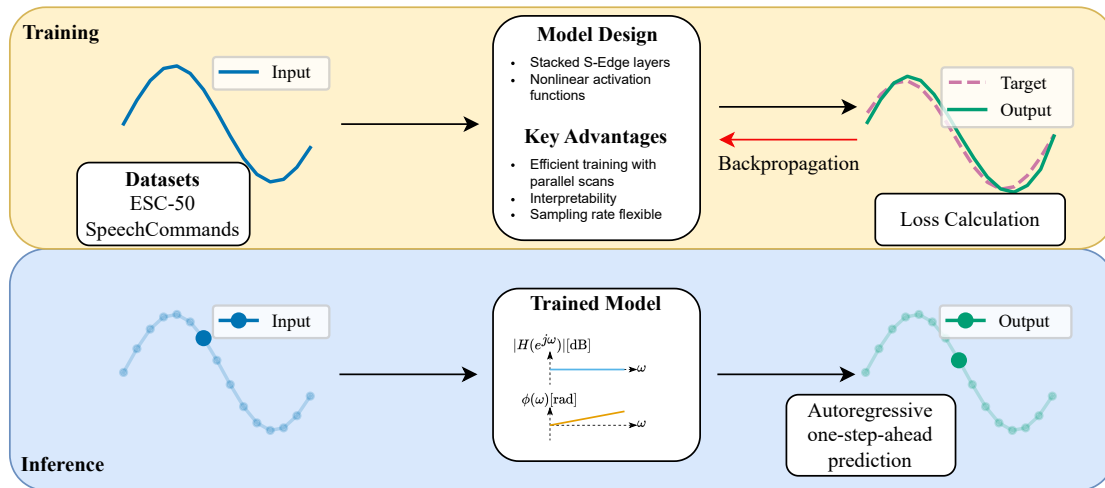


Figure 4.1: The training and inference stages are depicted. During training the models take sequences of raw audio as input and produce sequences at the output. Backpropagation is used to update the model parameters. During inference the models produce outputs sample by sample. The ideal transfer function has a magnitude of 0 dB and a linear phase response corresponding to a forecast of one sample.

The following sections describe the datasets, followed by a detailed explanation of the model architectures and the evaluation metrics. The methodology is concluded with a description of the specific tasks addressed in this thesis: one-step-ahead prediction and zero-phase filtering.

## 4.1 Datasets

Two different datasets are used to train and validate the models. The ESC-50 dataset covers environmental sounds while the SpeechCommands dataset contains spoken words. Each dataset is prepared in advance to fulfill certain criteria. They are resampled to a common sampling rate, and mean and standard deviation are calculated across the dataset. Furthermore, the datasets are split into train, test, and validation parts. The training part makes up 70% of the whole dataset, while the test and validation sets account for 15% each. To artificially increase the dataset size and improve model convergence, the audio samples are split into overlapping subsamples.

### 4.1.1 ESC-50 Dataset

The ESC-50 dataset contains environmental sounds of five different categories. These are animal sounds, natural soundscapes and water sounds, human sounds, interior/domestic sounds, and exterior/urban noises. In each of those categories, ten subclasses are defined, each containing 40 5 s audio clips, sampled at 44.1 kHz. In total the set contains roughly 2.8 hrs of recordings [25].

### 4.1.2 Speech Commands Dataset

The Speech Commands dataset covers human speech. It contains utterances of 35 words, spoken by 2,618 speakers. The data is stored as 1 s (or shorter) clips, sampled at 16 kHz. In total the set contains almost 30 hrs of raw audio. In contrast to the ESC-50 dataset, it does not contain any background sounds and is therefore well suited to train models specifically on speech [26].

## 4.2 Models

Mainly three models are used throughout this thesis: a linear model consisting of only one S-Edge layer, as well as two nonlinear models with three and nine layers. The nonlinear models apply a nonlinearity after each layer except the last one. The input to the models is raw audio and can be provided either as a whole sequence or sample-wise. The output that is produced is also interpreted as raw audio. The models can run in parallel mode during training, receiving raw audio sequences and producing raw audio sequences, or in autoregressive mode, receiving raw audio samples and producing only raw audio samples. The autoregressive mode is needed for real-time applications. The parallel mode, with parallel scans in the background, is needed for efficient training of the models.

Table 4.2 summarizes the used models. If not mentioned otherwise, these parameters apply to all experiments throughout this thesis.

Table 4.2: The table shows an overview of the models and their parameters, which are used in this thesis.

Model	Layers	State Size	Intermediate Size	Nonlinearity	Parameters
S	1	64	-	None	0.5k
M	3	64	32	LeakyReLU	18.6k
L	9	64	32	LeakyReLU	75.8k

Figure 4.2 illustrates the architecture of the models. For the two nonlinear models, LeakyReLU is applied as nonlinearity after each layer except the last one. The state size is set to 64 for all models. The intermediate size between the layers is set to 32 for the nonlinear models. The linear model does not have an intermediate size since it consists of only one layer. The input and output dimension is set to 1 for the one-step-ahead prediction task.

## 4.3 Evaluation

The trained models are evaluated using several metrics and analysis methods. In addition to the loss functions used during training, metrics based on the model parameters are considered. These include the eigenvalues of the state matrix and the corresponding transfer functions. For nonlinear models,



$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi \frac{k}{N} n}, \quad k = 0, \dots, N-1, \quad (4.1)$$

where  $N$  denotes the number of samples in the signal  $x_n$ . From the frequency-domain representations  $X_k$  and  $Y_k$ , the magnitude and phase of the transfer function

$$H_k = \frac{Y_k}{X_k}$$

can be calculated as

$$|H_k| = \sqrt{\Re\{H_k\}^2 + \Im\{H_k\}^2}, \quad (4.2)$$

and

$$\angle H_k = \text{atan2}(\Im\{H_k\}, \Re\{H_k\}). \quad (4.3)$$

Figure 4.3 shows an example of such an empirical transfer function, along with a graphical representation of the underlying calculation principle. Although the actual computation is performed in the frequency domain, time-domain plots can also provide an approximate estimation of the magnitude and phase by comparing amplitudes and time delays between the input and output signals:

$$|H_k| \approx \frac{A_o}{A_i}, \quad (4.4)$$

and

$$\angle H_k \approx -2\pi f_k \Delta T, \quad (4.5)$$

where  $A_i$  and  $A_o$  denote the maximum amplitudes of the input and output signals, respectively, and  $\Delta T$  represents the time delay between them. These relations are approximations that hold for pure sinusoidal signals and neglect noise and additional frequency components. Nevertheless, they provide a useful intuition for understanding the input–output characteristics of the model.

It has to be noted that this empirical method only considers the linear aspects of the model. Non-linear artifacts, such as harmonic distortion, will not be represented in the Bode plot. Therefore, the empirical Bode plot serves as an approximation of the models' linear behavior around the frequency of the input signal.

### 4.3.2 Total Harmonic Distortion

The THD metric is used to quantify the nonlinear distortion. This is a measure for the power of harmonic frequencies in relation to the fundamental frequency and is calculated as follows:

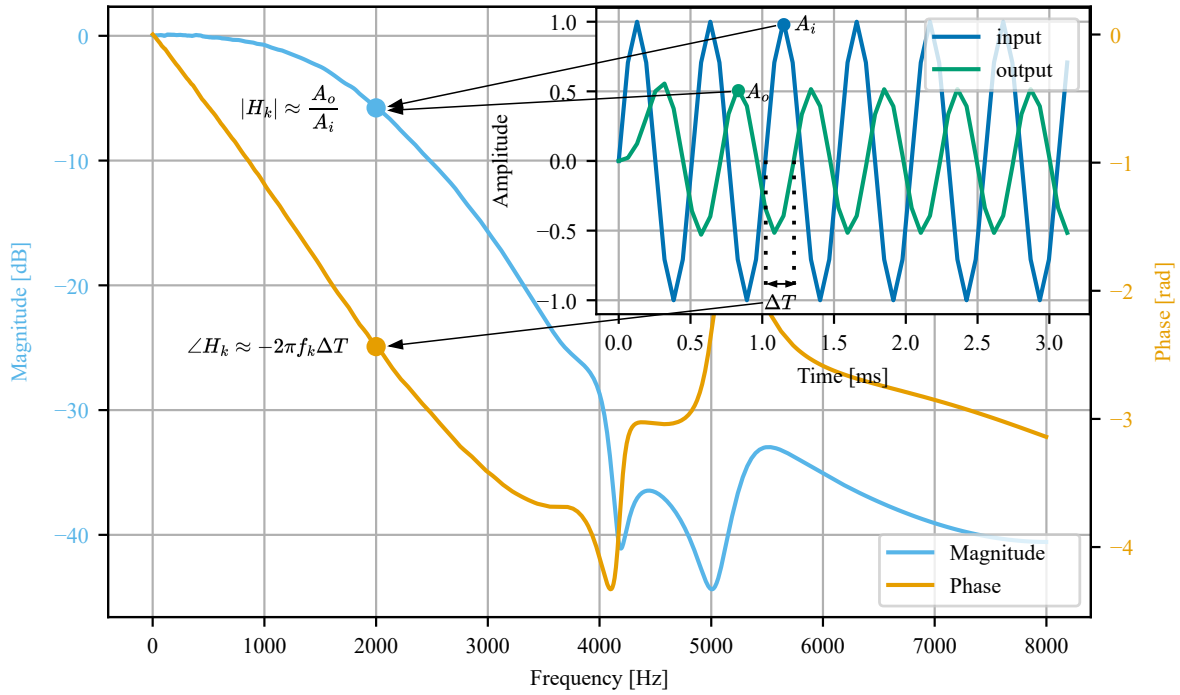


Figure 4.3: Empirical Bode plot alongside a time-domain representation of input and output signals for a specific frequency. While the transfer function is calculated in the frequency domain using the DFT, the time-domain representation allows an approximate estimation of magnitude and phase.

$$\text{THD} = \sqrt{\sum_{k=1}^K A_{k,rms}^2}, \quad (4.6)$$

with  $A_{k,rms}$  being the Root Mean Square (RMS) amplitude of the  $k$ -th harmonic.  $K$  is the number of harmonics considered. The amplitudes are obtained from the frequency spectrum of the signal. To also account for aliasing effects, the harmonic frequencies are folded back into the Nyquist range when they exceed it:

$$f_k = \begin{cases} f_s - f_k, & \text{if } f_k > \frac{f_s}{2} \\ f_k, & \text{otherwise.} \end{cases}$$

$f_s$  is the sampling frequency. Another interesting metric is the THD normalized to the fundamental frequency. This is calculated as

$$\text{THD}_{\text{ratio}} = \frac{\text{THD}}{A_{0,rms}}, \quad (4.7)$$

with  $A_{0,rms}$  being the RMS amplitude of the fundamental frequency. This metric gives a better understanding of the distortion in relation to the main signal component. The higher the THD ratio, the more distortion is present in the signal.

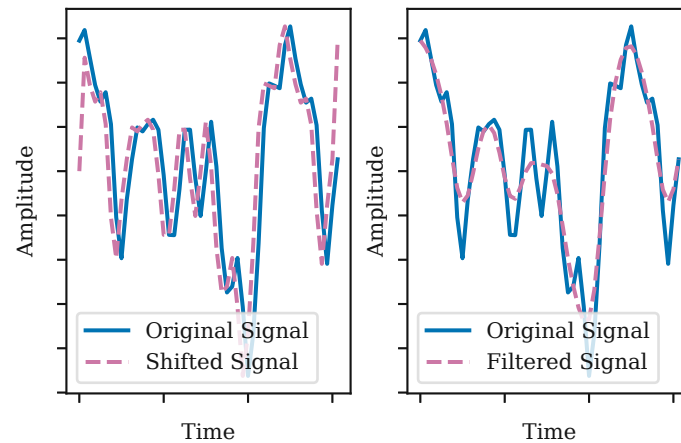


Figure 4.4: The figure shows the calculation of the ground truth signals for the prediction task on the left, which represents a time shift, and the filtering task on the right, which in this case is a low-pass filtered signal with a cut-off frequency of 2 kHz. The original signals are depicted in blue, the ground truth signals in purple.

## 4.4 One-Step-Ahead Prediction

The baseline approach for this thesis is to train a model with an input and output dimension of one for one-step-ahead prediction. This means that for each input sample, the model has to predict the next sample in the sequence. The following sections describe the ground truth calculation and a parallel approach that is tested to improve the prediction accuracy.

### 4.4.1 Ground Truth

To train the neural network models for one-step-ahead prediction of raw audio waveforms, the ground truth signals are calculated by using a simple temporal shift. For input value  $x_k$  at discrete time  $k$ , the corresponding target value is  $x_{k+1}$ . Figure 4.4 illustrates the ground truth for the prediction task on the left side.

### 4.4.2 Parallel Approach

Figure 4.5 illustrates a parallel approach for the one-step-ahead prediction, where the input signal is split into four frequency bands using zero-phase filtering. Each band is then processed by a dedicated model that is trained only on the respective frequency band. The final output signal is calculated as the sum of the outputs of the four models. The key point of this approach is the zero-phase filtering of the input signal, which ensures that no phase distortions are introduced. That means that the input signal can be perfectly reconstructed from the four filtered signals by summing them up. From the sampling rate of 16 kHz, each band spans over 2 kHz. The benefit of this approach is that each model only sees a limited frequency range at its input and can therefore specialize on that range. In Figure 4.5 this is

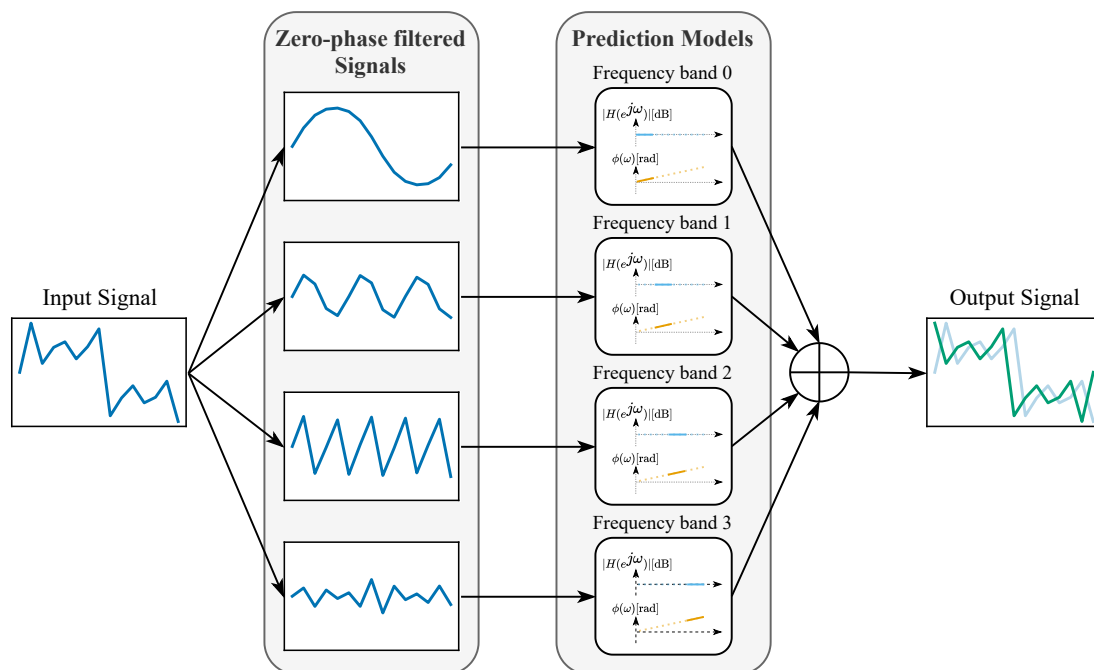


Figure 4.5: The figure illustrates a parallel approach for the one-step-ahead prediction task. The input signal is split into four frequency bands using zero-phase filtering. Four prediction models are trained on the respective bands. The sum of the outputs of the models represents the final output signal, which is the prediction of the overall system.

illustrated by the ideal transfer functions of each model. Each model should have a magnitude of 0 dB in its pass band. The phase response of each model should ideally be a positive linear slope, corresponding to a prediction by one sample period. The regions outside the pass band are not relevant, since the input signal has no energy in those areas if filtered properly. In the following section, zero-phase filtering is described in more detail.

## 4.5 Zero-Phase Filtering

To achieve zero-phase filtering, a non-causal operation is required. This means that future samples of the signal have to be taken into account when calculating the current output sample. Therefore, zero-phase filtering cannot be implemented in real-time applications directly. The implementation in this thesis uses the `filtfilt` function from the `scipy` library. The underlying filters are chosen as Butterworth IIR filters of fourth order. This method applies the filter in forward and backward direction to eliminate phase distortions. The resulting signals have zero phase shift for all frequencies but are limited to the respective frequency band. The right side of Figure 4.4 illustrates the input and output signals for a low-pass filter that is applied using the `filtfilt` function.

### 4.5.1 Training

Since the `filtfilt` function is non-causal, it cannot be used in real-time applications. Therefore, the goal is to train SSMs to approximate the zero-phase filtering behavior. The models are trained on input-output pairs, where the input is the raw audio signal and the output is the zero-phase filtered signal. The models have an output dimension of four, corresponding to the four frequency bands. Apart from that, the model architecture is the same as described in Section 4.2. During training, each output is penalized by its respective loss function. The total loss, which is used for backpropagation, is calculated as the sum of the four individual losses. The goal is that the model learns to approximate the ideal transfer function for zero-phase filtering.

### 4.5.2 Ideal Transfer Function

Figure 4.6 shows the magnitude and phase responses of the ideal zero-phase filters. The responses are obtained by comparing the inputs and outputs of the `filtfilt` function, which applies the filter in forward and backward direction. Each filter has a passband of 2 kHz. Within this range, the magnitude is approximately 1, while it drops to 0 in the stopband.

The phase plot reflects the desired zero-phase characteristic relevant for the ANC application. Small deviations at very low and very high frequencies can be observed, but they are negligible because the

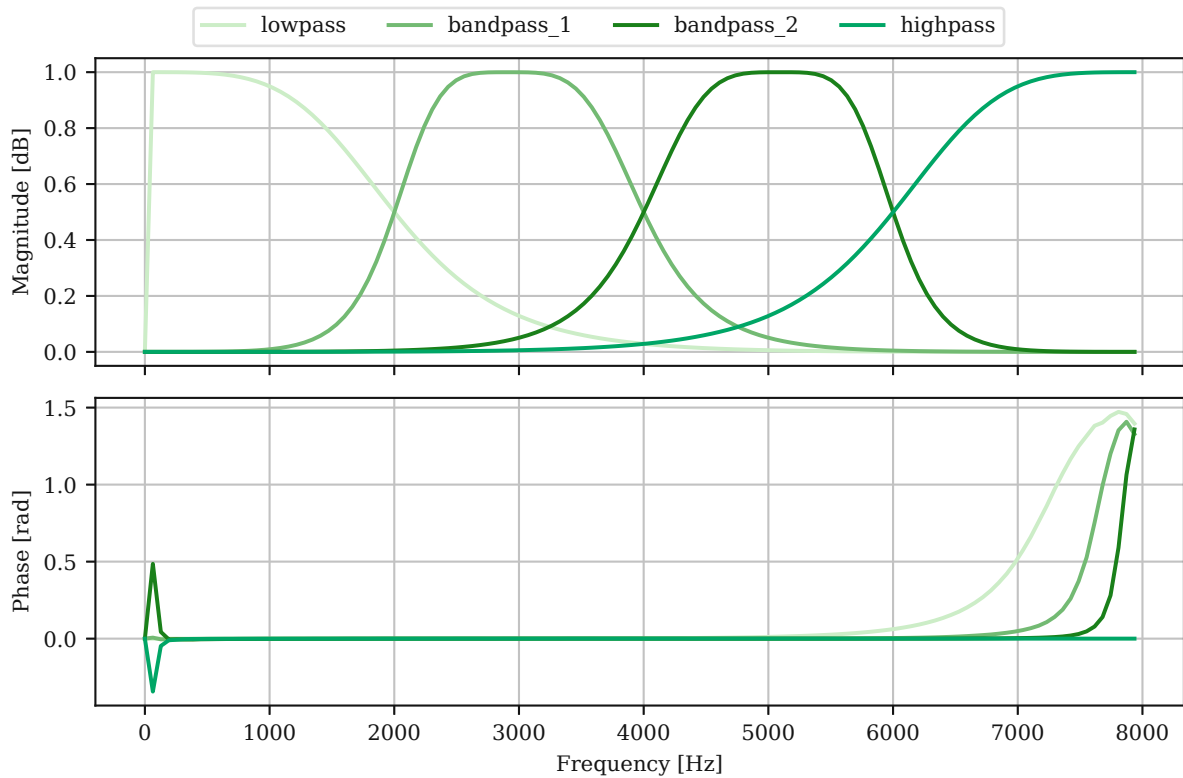


Figure 4.6: Magnitude and phase responses of the ideal zero-phase filters. The Python function `filtfilt` is used to generate the curves by applying the filter in forward and backward direction. Each filter has a passband width of 2 kHz.

magnitude in these regions is close to zero. As a result, the phase has no practical effect on the output.

## Chapter 5

# Experiments

This chapter describes the experiments conducted to evaluate SSMs on one-step-ahead prediction and zero-phase filtering using the datasets and methods described in the previous chapter. The main goal is to understand how the models behave on real audio data and whether they exhibit properties that are needed for ANC systems, such as a stable frequency response and accurate prediction or filtering across different signal types. The focus of this work is on model design and evaluation in simulation. Therefore, no hardware implementation is performed. Real-time suitability is analyzed theoretically based on model complexity and expected hardware constraints.

The experiments start with a comparison of different model sizes using a fixed baseline setup for the one-step-ahead prediction task in Section 5.1. This analysis aims to understand how model size and depth affect prediction accuracy. Next, the influence of different sampling rates during training and inference is investigated in Section 5.2. This is followed by a parallel approach in Section 5.3 in which the input signal is split into separate frequency bands and processed by multiple S-Edge layers. Although this approach is not real-time capable, it serves to evaluate whether decomposing the task into simpler subproblems can improve accuracy. The chapter then presents the results of the zero-phase filtering experiments in Section 5.4. Finally, the real-time aspects are discussed in Section 5.5 by relating the computational and memory requirements of the models to the constraints of typical headphone hardware.

### 5.1 Baseline

The three models presented in Table 4.2 are trained on the one-step-ahead prediction task. To evaluate the dataset generalization capabilities of the models, each is trained on both the SpeechCommands and ESC-50 datasets.

Figure 5.1 shows the resulting Bode diagrams obtained using the empirical transfer function ap-

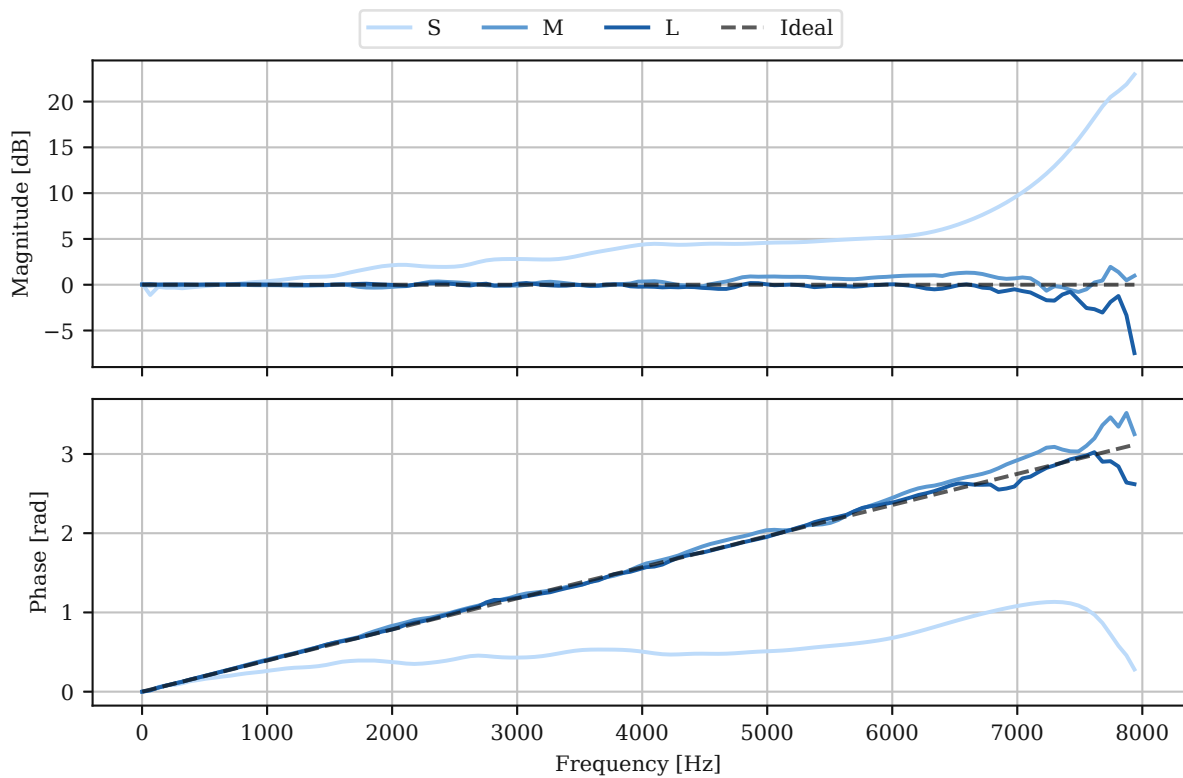


Figure 5.1: Magnitude and phase responses of the three models trained for one-step-ahead prediction on the SpeechCommands dataset. Additionally, the ideal magnitude and phase responses are depicted. The M and L models achieve the desired behavior with slight deviations from the ideal responses, while the S model, lacking nonlinearities, fails to model the complex dynamics of the task.

Table 5.1: MAE and MSE for four models trained on a one-step-ahead prediction task. Evaluation is conducted on the SpeechCommands and ESC-50 datasets. Each model is trained on both datasets to assess generalization.

Model	Trained On	ESC-50		SpeechCommands	
		MAE	MSE	MAE	MSE
S	ESC-50	2.60e-02	5.50e-03	9.35e-03	9.71e-04
M		1.58e-02	1.81e-03	4.09e-03	2.07e-04
L		1.42e-02	1.52e-03	3.28e-03	1.45e-04
S	SpeechCommands	3.18e-02	1.36e-02	7.87e-03	1.00e-03
M		1.73e-02	2.30e-03	3.25e-03	1.55e-04
L		1.51e-02	1.76e-03	2.62e-03	1.07e-04

proach. The ideal model would behave as an all-pass filter with a magnitude of 0 dB across all frequencies. The smallest model (S) deviates substantially from this ideal response, exhibiting high gain at higher frequencies which leads to excessive output amplitudes. In contrast, the M and L models closely match the ideal magnitude response, with only minor deviations at high frequencies. The deeper architectures and the use of nonlinearities between layers enable these models to capture the desired predictive behavior.

The ideal phase response of a one-step-ahead prediction model corresponds to a one-sample time advance. In the frequency domain, this corresponds to a linear phase response with a positive slope, ranging from 0 rad at 0 Hz to  $\pi$  rad at the Nyquist frequency. Similar to the magnitude response, this desired behavior is observed in the M and L models, while the S model fails to approximate the linear slope and consequently cannot predict the next audio sample accurately.

Table 5.1 summarizes the results for both datasets, reporting the MAE and Mean Squared Error (MSE) calculated on the respective test sets. Overall, larger models achieves better performance. The losses on the ESC-50 dataset are approximately five to ten times higher than those on SpeechCommands, which can be attributed to the higher complexity and the smaller size of the ESC-50 dataset since it contains a wider range of high-frequency content. Furthermore, the table shows that the models generalize well across datasets as performance degradation when evaluated on a different dataset is relatively small. For both datasets, the degradation is in the order of  $1e-3$ . This indicates that the models primarily learn to capture the frequency content of the input signal rather than memorizing specific waveform patterns. This property is particularly important for ANC applications as the model must generalize across a wide range of noise types.

Figure 5.2 shows two segments of the waveform corresponding to the word *right*. The plots include the target (ground truth), model output, and the resulting error signals. The left plot depicts a lower-frequency segment representing the beginning of the word, where the prediction error is small. The model accurately reproduces the waveform and its phase, with only minor amplitude deviations.

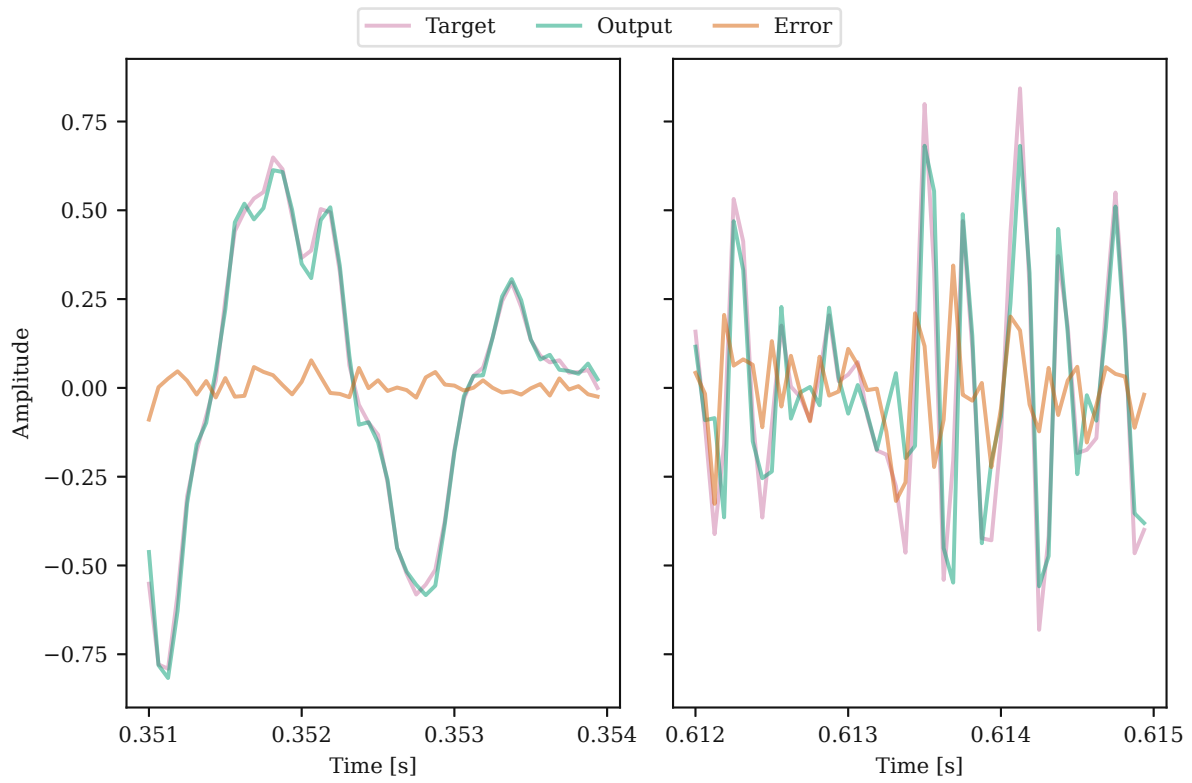


Figure 5.2: Predicted and target waveforms for two time-series segments of the word *right*, including the corresponding error signals. The left plot resembles a low frequency section of the signal, the right plot a high frequency section.

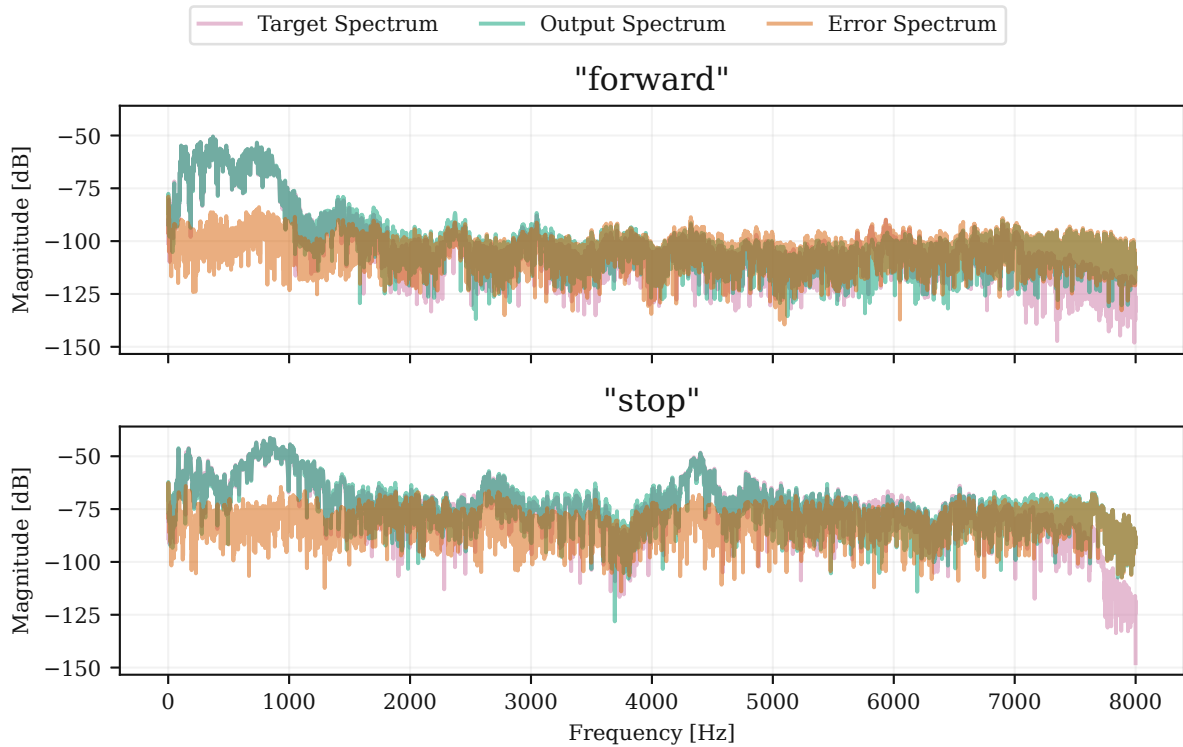


Figure 5.3: Frequency spectra of target signals and model outputs for two words from the SpeechCommands dataset, including the spectra of the corresponding error signals.

The right plot, showing a higher-frequency segment corresponding to the letter *t*, illustrates a larger prediction error. While the phase is still estimated reasonably well, the predicted amplitudes are consistently underestimated. This correlates to the Bode diagram in Figure 5.1, where the magnitude and phase responses deviate more from the ideal response at higher frequencies.

To further analyze the model behavior, the frequency spectra of the output signals are examined. Figure 5.3 shows the spectra of two words from the SpeechCommands dataset. To assess the model performance in a noise-cancellation context, the spectra of the corresponding error signals are also included. The error signal is defined as the difference between the target and the model output, analogous to the residual in an ideal feedback loop.

The word *forward* represents one of the better performing examples, mainly due to its limited high-frequency content. The error spectrum remains very low at around  $-100$  dB, and low frequencies are almost perfectly cancelled. In contrast, the word *stop* contains stronger high-frequency components, which the network still captures, but result in higher overall error magnitudes.

Overall, the baseline experiments demonstrate that the proposed models can reliably learn the desired one-step-ahead prediction behavior, but they also reveal systematic deviations at higher frequencies. Therefore, the following section investigates the influence of different sampling rates during training and inference and their effect on the learned frequency response.

## 5.2 Sampling Rate Experiments

A key property of SSMs is the ability to operate at different sampling rates while maintaining the desired frequency behavior. This property is commonly exploited in state-of-the-art approaches, for example in audio classification where inference at lower sampling rates can reduce computational cost [3]. To get a consistent frequency response across different sampling rates, the step size for the discretization can be scaled accordingly, resulting in an adjustment of the discrete matrices  $\mathbf{A}_d$  and  $\mathbf{B}_d$ . In this work, however, the eigenvalues do not need to be adjusted since the model learns a linear magnitude and phase response based on a normalized frequency representation. As a result, different sampling rates can be applied directly without explicitly switching the model. This motivates an investigation of how different sampling rates during training and inference affect the learned frequency response and the overall performance.

Two models with identical architecture and model size  $L$  are trained on the ESC-50 dataset, one at a sampling rate of 16 kHz and one at 44.1 kHz. Figure 5.4 shows the Bode plots of both models over a normalized frequency axis. The model trained at 16 kHz follows the ideal magnitude and phase response more closely, while the model trained at 44.1 kHz shows larger deviations. This behavior is likely caused by the spectral distribution of the dataset. High-frequency components are less prevalent, so a large portion of the normalized frequency range at 44.1 kHz contains only little informative content. The frequency content up to 8 kHz is well represented in both cases, but above that the energy in the dataset decreases significantly. As a result, the model trained at 16 kHz has access to more relevant training data.

This effect is also visible in Figure 5.5, which shows the normalized spectra of the same signal sampled at 16 kHz and 44.1 kHz. For the 44.1 kHz signal, the spectral magnitude already significantly decreases at half of the Nyquist frequency, whereas the spectrum of the 16 kHz signal remains relatively flat, with similar magnitudes across the frequency range. This indicates that the improved Bode response observed for the 16 kHz model is likely related to the more uniform availability of relevant training data over the normalized frequency axis.

Table 5.2 reports MAE and MSE for both models on the ESC-50 and SpeechCommands datasets. Both datasets are evaluated at 44.1 kHz. The SpeechCommands dataset is initially sampled at 44.1 kHz and is therefore resampled to the higher sampling rate with the torchaudio library [27]. In general, the model trained at the same sampling rate as the evaluation data performs slightly better. An exception is observed for the MAE on SpeechCommands where the 16 kHz model performs marginally better. Overall, the differences are small, indicating that the model can be operated at sampling rates different from those used during training with only minor performance degradation.

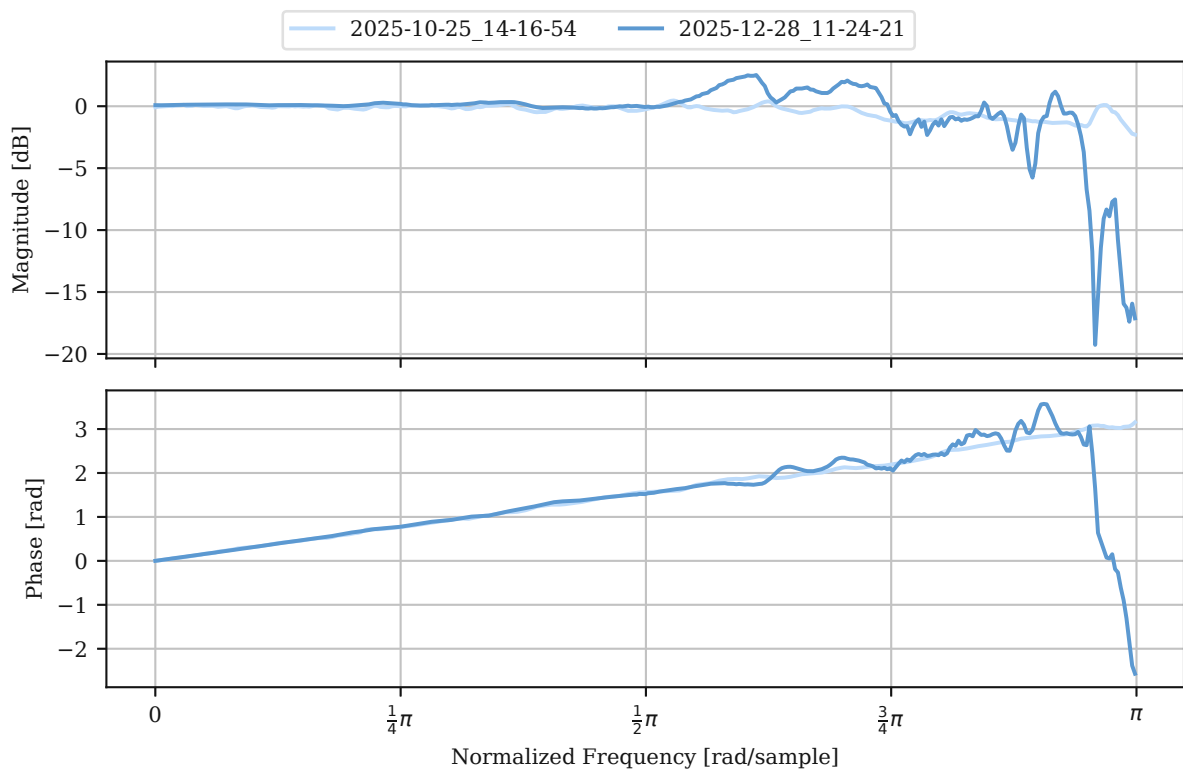


Figure 5.4: This figure depicts the magnitude and phase responses of two models trained at sampling rates of 16 kHz and 44.1 kHz.

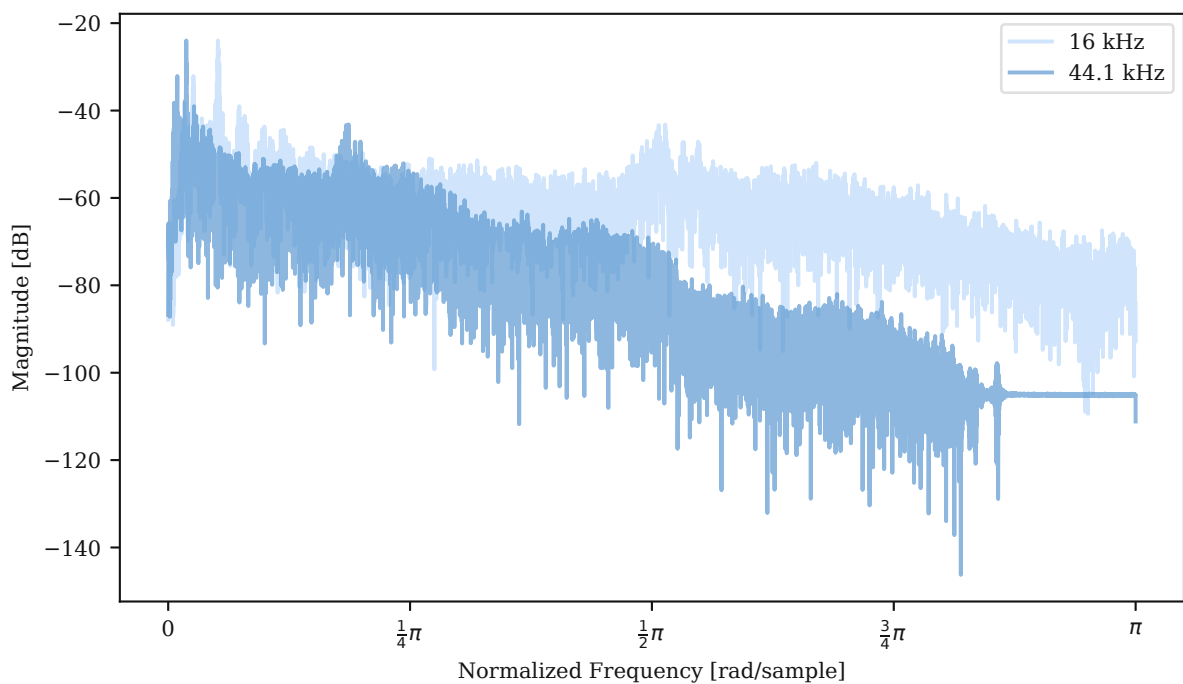


Figure 5.5: The normalized frequency spectra of an audio sample from the ESC-50 dataset is shown for two different sampling rates, 16 kHz and 44.1 kHz. The data is resampled with the torchaudio library.

Table 5.2: MAE and MSE for two models trained at 16 kHz and 44.1 kHz. Evaluation is performed on the ESC-50 and SpeechCommands datasets at a sampling rate of 44.1 kHz. The resampling of the datasets is done with the torchaudio library.

Trained @	ESC-50		SpeechCommands	
	MAE	MSE	MAE	MSE
16 kHz	4.85e-03	3.19e-04	4.89e-04	1.96e-06
44.1 kHz	3.44e-03	2.34e-04	5.37e-04	1.74e-06

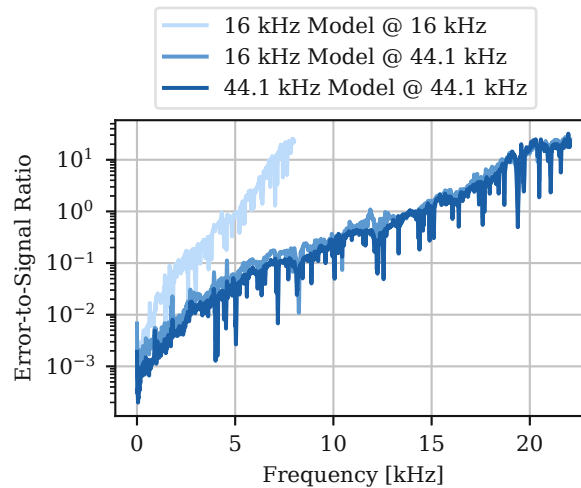


Figure 5.6: The ESR across frequencies for two different models for the SpeechCommands dataset is shown. The 16 kHz model is evaluated at both frequencies, while the 44.1 kHz model is only evaluated at 44.1 kHz.

Figure 5.6 supports this conclusion by showing the Error-to-Signal Ratio (ESR) for three scenarios. The model trained at 16 kHz is evaluated at both 16 kHz and 44.1 kHz, while the model trained at 44.1 kHz is evaluated at 44.1 kHz.

The comparison of the two 44.1 kHz evaluations indicates that the models can generalize well across sampling rates, with only minor accuracy degradation when evaluated at a different rate than they are trained on. This finding is significant for practical applications, as it allows for flexibility in deployment scenarios where the sampling rate may vary. Furthermore, training can be conducted at a lower sampling rate to reduce computational load, while still achieving good performance at higher rates during inference.

Additionally, the plot shows that inference at higher sampling rates leads to a lower one-step prediction error, since the time interval between consecutive samples is smaller. As a result, the signal changes less from one sample to the next, which simplifies the prediction task for the model. This can be advantageous in feedback-based ANC systems, although it comes with increased computational cost.

The sampling rate experiments show that the proposed models can be flexibly applied across dif-

ferent sampling rates with only minor performance degradation. Training at lower sampling rates is beneficial as it reduces computational cost and leads to a more uniform distribution of relevant frequency content, henceforth improving the learned frequency response. At the same time, inference at higher sampling rates results in a lower one-step prediction error due to the shorter time interval between consecutive samples. Across all experiments conducted so far, the models consistently perform better when the frequency content of the signals is more limited, as observed for speech compared to broadband environmental sounds. This observation motivates the following investigation of an explicit band-wise modeling approach, in which the prediction task is decomposed into multiple frequency-specific subproblems.

### 5.3 Parallel Approach

To analyze the modeling capacity of S-Edge layers independently of real-time constraints, a parallel layer structure, as described in Section 4.4.2, is employed. Since the zero-phase filtering is implemented with `filtfilt`, the resulting model is non-causal and therefore not suitable for real-time operation. This experiment is meant as an analysis of the underlying modeling capability rather than a deployable architecture.

For each predefined frequency band, an independent linear model of size  $S$  is trained, effectively reducing the overall task complexity, and the individual outputs are combined in parallel to form the final prediction. Figure 5.7 illustrates the resulting Bode diagram of the model. Each input-output pair is analyzed using the empirical transfer function analysis. The response of each layer within its respective frequency band can be observed. Outside the trained frequency bands, both magnitude and phase responses deviate from the desired prediction behavior. The increased magnitude gain in these regions does not impact the results since the input signals contain negligible out-of-band energy.

Table 5.3 shows that the parallel model achieves roughly one order of magnitude lower MSE and MAE on both datasets compared to the baseline. The only exception is the MAE on `SpeechCommands`, where the improvement is approximately a factor of two.

Figure 5.8 compares the ESR across frequencies for the baseline and the parallel models on the `SpeechCommands` dataset. The parallel approach significantly reduces the error across the spectrum, with the largest improvements observed at higher frequencies. This supports the hypothesis that decomposing the task into band-limited subproblems simplifies the learning of high-frequency dynamics.

Although this setup cannot be used in real-time, it demonstrates that S-Edge layers can accurately model raw audio even when only linear layers are used. The experiment also shows that splitting the frequency space allows the model to learn clean and stable transfer functions for both low and high

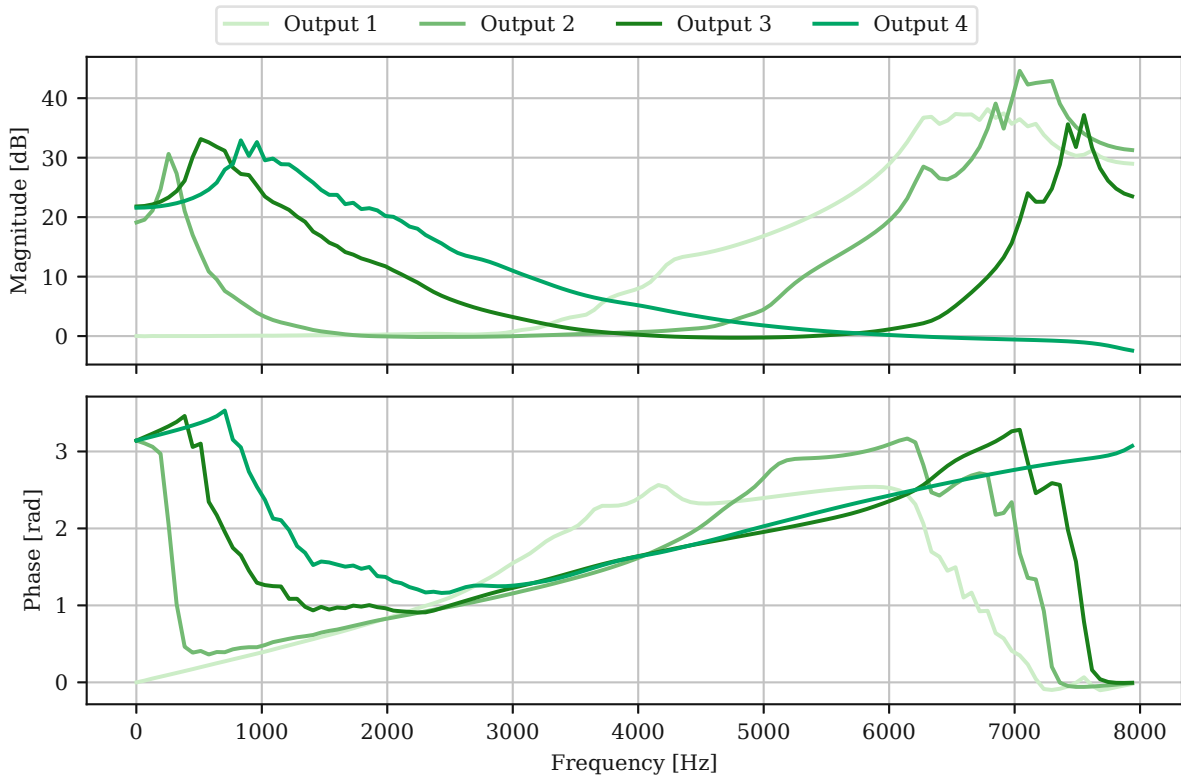


Figure 5.7: Magnitude and phase responses of the parallel model. Each curve corresponds to one input–output pair trained on a specific frequency band. Outside the respective bands the models have learned high gain and nonlinear phase responses.

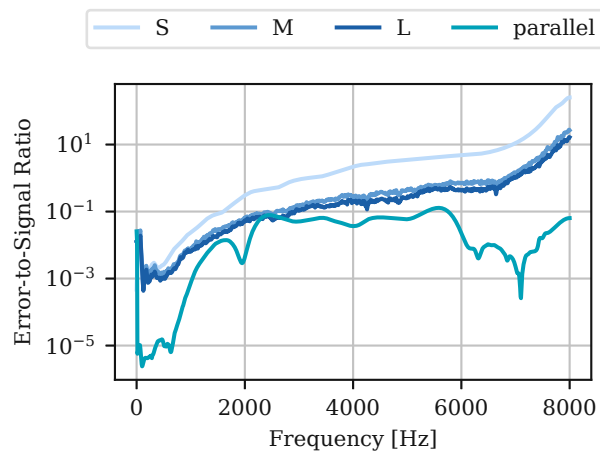


Figure 5.8: The ESR across frequencies, evaluated on the SpeechCommands dataset, is shown for both the baseline and the parallel models. The parallel model, although not real-time capable, achieves the lowest overall error.

Table 5.3: MAE and MSE losses for the parallel model on the SpeechCommands and ESC-50 datasets.

Model	Trained On	ESC-50		SpeechCommands	
		MAE	MSE	MAE	MSE
Parallel	ESC-50	4.53e-03	3.06e-04	1.37e-03	2.61e-05
Parallel	SpeechCommands	5.22e-03	4.78e-04	1.89e-03	2.94e-05

frequencies.

## 5.4 Zero-Phase Filtering

As shown in the previous experiment, the accuracy of one-step-ahead prediction can be improved by using a parallel frequency band approach. Motivated by this result, the goal of this experiment is to train SSMs on the zero-phase filtering task, similar to the approach described in [28]. In contrast to the referenced work, which relies exclusively on synthetic data, the models in this experiment are trained on the SpeechCommands dataset to assess their performance on real-world audio signals.

The ground truth signals are generated using `filtfilt` as described in the methodology section. The model architecture is identical to the baseline setup, with the exception that it produces four outputs instead of one. Each output corresponds to a distinct 2 kHz frequency band and is trained on its own zero-phase filtered target signals. For this experiment, the model size L is used.

Figure 5.9 shows the Bode diagrams of the four outputs, computed using the empirical transfer function approach. The magnitude responses exhibit the intended behavior as each output passes only its assigned frequency band and attenuates the remaining spectrum. The phase responses exhibit approximately zero-phase behavior within the passbands, but show nonlinear distortions in the stopbands. Since frequency components in the stopbands are already strongly attenuated, the associated phase distortions do not affect the reconstructed signal.

For synthetic test signals, the model achieves high filtering accuracy with minimal distortion, similar to the results presented in [28]. For complex real-world signals, however, the Bode diagrams are not sufficient to fully characterize the model behavior since nonlinear distortions are not captured by this analysis. Therefore, the harmonic distortion introduced by the model is analyzed in Figure 5.10. The THD ratio and absolute values are shown for all four outputs. A ratio of 0 dB indicates that the harmonic components in the output have the same power as the fundamental frequency. The results show that the harmonic content increases systematically with frequency, independent of the trained band. This trend suggests that the model introduces increasing nonlinearities at higher frequencies, likely due to limited temporal resolution and the increased difficulty of representing rapid signal variations. Furthermore, the THD ratio is lower inside the respective trained bands, while outside these

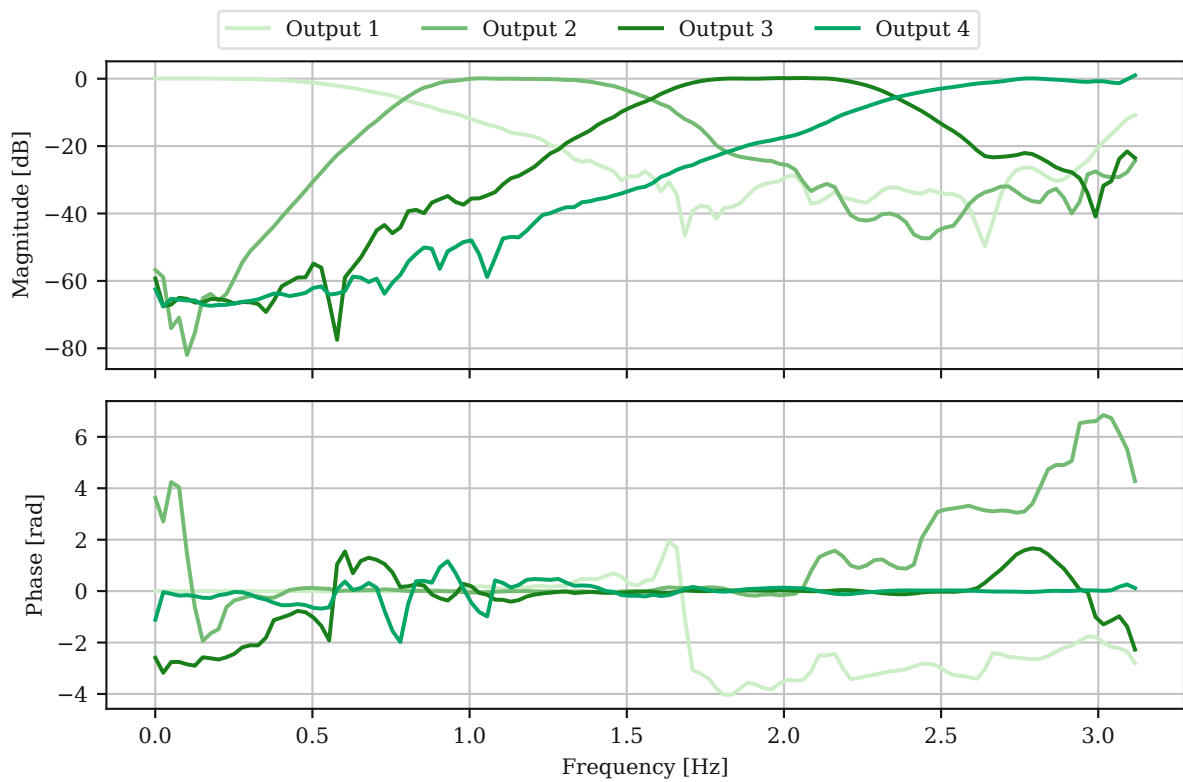


Figure 5.9: Bode plots of the four outputs of the zero-phase filtering model. The magnitude responses show clear lowpass, bandpass and highpass behavior, while the phase responses deviate from ideal zero-phase filtering.

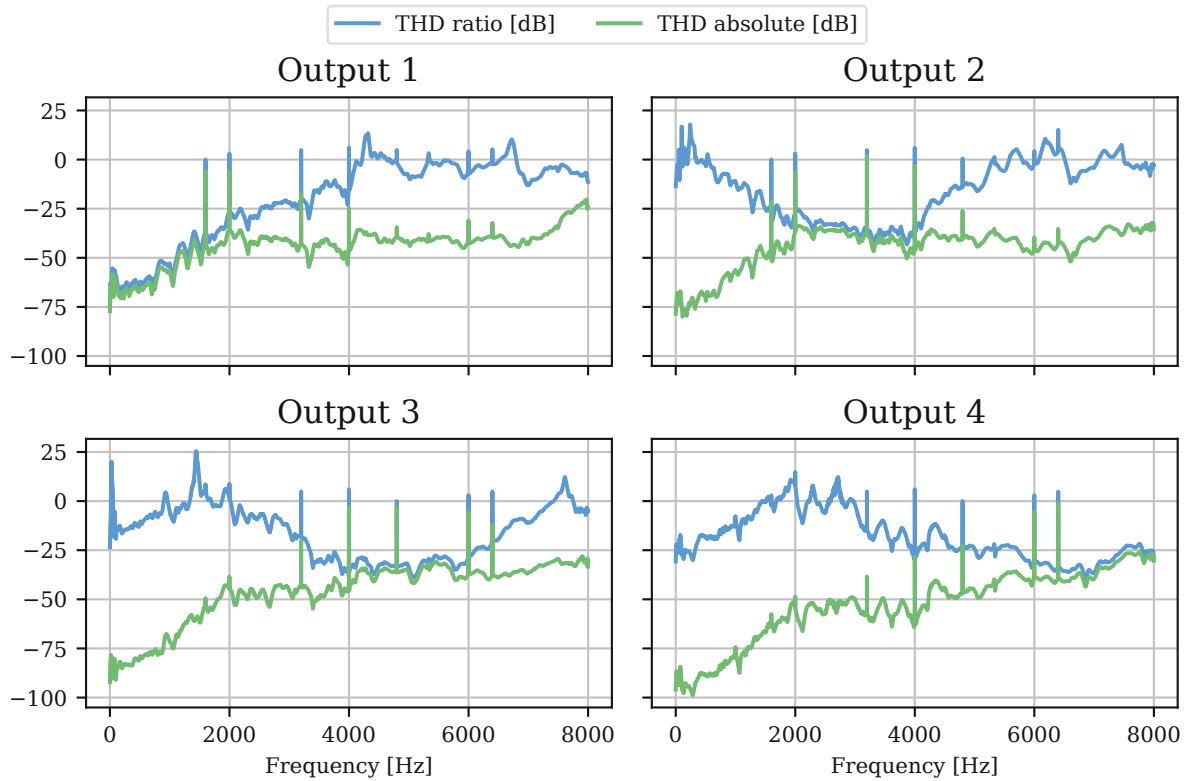


Figure 5.10: THD ratio and absolute values for the four outputs of the zero-phase filtering model. The THD increases with frequency and in some cases exceeds the main frequency component outside the trained bands.

bands the harmonic content rises and can even exceed the power of the main frequency component.

To evaluate the model on real-world data, the average attenuation on the SpeechCommands dataset is computed. Attenuation is defined as the ratio between output and input amplitude in decibels. Figure 5.11 shows the resulting attenuation curves. The four frequency bands are clearly distinguishable: passbands remain close to 0 dB, while stopbands reach attenuation levels of up to  $-50$  dB. This confirms effective band separation across the dataset. However, output 1 shows noticeably weaker stopband attenuation of approximately  $-10$  dB. Since this output corresponds to the lowest frequency band with most of the signal energy, it is more difficult for the model to isolate it from higher-frequency components with significantly lower amplitudes.

The objective of this experiment is to combine the advantages of the parallel prediction setup with a real-time capable filtering stage. In practice, this goal cannot be achieved. The observed phase distortions and insufficient stopband attenuation in some outputs lead to error amplification when combined with the prediction model, resulting in a significant degradation compared to the baseline. This degradation can be attributed to the combination of low attenuation in the filter model and high gain outside the trained bands in the prediction model.

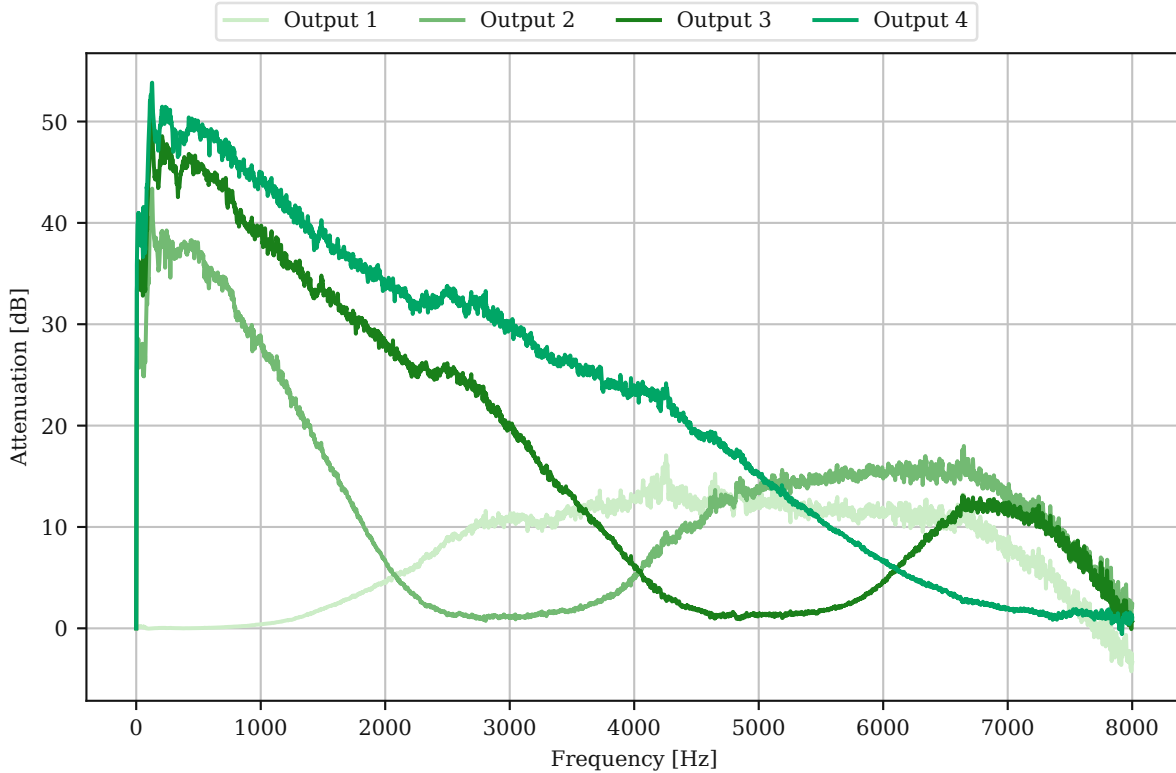


Figure 5.11: Average attenuation on the SpeechCommands dataset for the four outputs of the zero-phase filtering model.

## 5.5 Real-Time Capability

While the preceding experiments focused on prediction accuracy and frequency-domain behavior, practical deployment in an ANC scenario also imposes strict real-time constraints. In particular, the models must operate under tight computational and memory budgets. The following analysis therefore evaluates the real-time capability of the proposed forecasting models by relating their computational complexity and memory requirements to realistic constraints of typical System-On-Chip (SOC) platforms used in ANC.

Table 5.4 summarizes the number of trainable parameters and the required Multiply-Accumulates (MACs) per sample for the three model sizes. Since the models are intended for deployment in an ANC context, their computational cost must remain well below the processing budget available on typical low-power audio SOCs.

At a sampling rate of 16 kHz, the system must generate one prediction every

$$T_{\text{sample}} = \frac{1}{16,000} = 62.5 \mu\text{s}.$$

This processing window includes all operations required for the forward pass, as well as any additional

Table 5.4: The number of parameters, MACs per sample, and state memory requirements for the three model configurations. A sample in this case corresponds to one datapoint in the audio signal.

Model	Parameters	MACs / Sample	State Memory
S	515	513	2.1 kB
M	18,626	18,496	74.5 kB
L	75,842	75,328	303.4 kB

overhead introduced by buffering or audio I/O. In practice, only a fraction of this time can be used for the model itself, which makes a low computational footprint essential.

Modern bluetooth audio SOCs, such as the Qualcomm QCC5144 chipset, operate at 120 MHz [29]. Under the idealized assumption that one MAC operation can be executed per clock cycle, the available compute budget per audio sample is

$$N_{\text{cycles}} = \frac{120 \text{ MHz}}{16\,000} = 7,500 \text{ cycles/sample.}$$

This means that any model running at 16 kHz must stay significantly below 7,500 MACs per sample.

Comparing these limits to the values listed in Table 5.4, only the smallest model (S) satisfies this requirement with 513 MACs per sample. The medium (M) and large (L) models require 18,496 and 75,328 MACs per sample, respectively. These exceed the real-time budget by factors of approximately 2.5 and 10. Therefore, despite their potentially higher modeling capacity, these larger variants cannot be deployed on the target hardware at 16 kHz without violating the processing deadline.

In addition to runtime performance, memory consumption is also an important constraint on embedded audio processors. Assuming 32-bit floating-point parameters, the memory usage of the three model configurations is listed in table 5.4. The Qualcomm QCC5144 includes 448 kB of data RAM, which means that all three models would theoretically fit into memory, even without quantization or pruning.

Overall, this analysis shows that, among the evaluated configurations, only the smallest model (S) meets the real-time processing requirements imposed by a 16 kHz ANC system running on a typical low-power audio SOC.



## Chapter 6

# Conclusion

This work is embedded in the context of ANC, where a simple feedback-loop architecture with a neural network is proposed to predict and cancel noise before it reaches the ear. For this purpose, a series of experiments are conducted to evaluate the suitability of SSMs for audio signal prediction. The primary objective is to assess whether such models are capable of accurately predicting the next audio sample based on past samples and whether they are suitable for real-time operation.

The results show that SSMs are generally able to model speech and environmental sounds, although their performance strongly depends on the model size and the frequency content of the input signals. In particular, deeper models with a larger number of layers consistently outperform smaller models. Furthermore, the spectral characteristics of the signals play a crucial role. Narrowband signals that contain only a limited range of frequencies are easier to predict than broadband signals. This is reflected in the performance differences between datasets. The SpeechCommands dataset, which contains clear and non-noisy speech samples, can be modeled more accurately during training and inference than the ESC-50 dataset, which contains noisy and spectrally diverse environmental sounds.

Motivated by this observation, a band-based approach is investigated, in which the audio signal is decomposed into multiple frequency bands using zero-phase filtering and processed by separate models for each band. By reducing the effective frequency range per model, this approach significantly improves prediction accuracy. However, zero-phase filtering is inherently non-causal and therefore not compatible with real-time processing, which limits its applicability in ANC systems.

To address this limitation, models are trained to emulate low-pass, band-pass, and high-pass behavior without introducing phase distortion. While this approach yields promising results on synthetic signals such as pure sinusoids and their superpositions, the performance on real-world audio data remains limited. In particular, the attenuation of frequencies in the stopband is insufficient, making the approach unsuitable for a combination with the forecasting models. As a result, the band-based ap-

proach remains a theoretical concept that requires further investigation before it can be applied in a real ANC scenario.

To assess the real-time capability of the proposed forecasting models, a theoretical analysis of their computational complexity is conducted. The findings indicate that real-time operation with sample-wise processing is feasible for small models on modern hardware. However, larger models, which demonstrated superior prediction accuracy, may struggle to meet real-time constraints without further optimizations.

The limitations of this work also include the exclusive focus on offline training and evaluation, the absence of a real-time implementation, and the restriction to speech and environmental sound datasets. Future work could therefore focus on implementing and evaluating the proposed forecasting models in a real-time ANC system. In addition, further optimizations of the model architectures and pruning techniques could be explored to reduce model size while maintaining prediction accuracy. Finally, a more detailed investigation of oversampling strategies and the restriction to specific frequency ranges may provide further insights into improving performance in practical noise-cancellation applications.

# Bibliography

- [1] P. Lueg, "Process of silencing sound oscillations," *US pat ent 2043416*, 1936.
- [2] L. Lu, K.-L. Yin, R. C. de Lamare, Z. Zheng, Y. Yu, X. Yang, and B. Chen, "A survey on active noise control in the past decade—part i: Linear systems," *Signal Processing*, vol. 183, p. 108039, 2021.
- [3] M. Bittner, D. Schnöll, M. Wess, and A. Jantsch, "Efficient and interpretable raw audio classification with diagonal state space models," *Machine Learning*, vol. 114, no. 8, p. 175, 2025.
- [4] D. Dallinger, M. Bittner, D. Schnöll, M. Wess, and A. Jantsch, "Piano-ssm: Diagonal state space models for efficient midi-to-raw audio synthesis," 2025.
- [5] H. Zhang and D. Wang, "Deep anc: A deep learning approach to active noise control," *Neural Networks*, vol. 141, pp. 1–10, 2021.
- [6] S. Elliott, I. Stothers, and P. Nelson, "A multiple error lms algorithm and its application to the active control of sound and vibration," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 10, pp. 1423–1434, 1987.
- [7] D. R. Morgan, "History, applications, and subsequent development of the fxlms algorithm [dsp history]," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 172–176, 2013.
- [8] B. Widrow and S. Stearns, *Adaptive Signal Processing*, ser. EDITED BY ALAN V. OPPENHEIM. Prentice-Hall, 1985. [Online]. Available: <https://books.google.com/books?id=X74QAQAAMAAJ>
- [9] S. Liebich, J. Fabry, P. Jax, and P. Vary, "Signal processing challenges for active noise cancellation headphones," in *Speech Communication; 13th ITG-Symposium*. VDE, 2018, pp. 1–5.
- [10] T. Frey and M. Bossert, *Signal-und Systemtheorie*. Springer, 2008, vol. 2.
- [11] S. J. Orfanidis, *Introduction to signal processing*. Prentice-Hall, Inc., 1995.
- [12] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

- [13] A. Gu, K. Goel, A. Gupta, and C. Ré, “On the parameterization and initialization of diagonal state space models,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 35 971–35 983, 2022.
- [14] J. T. Smith, A. Warrington, and S. W. Linderman, “Simplified state space layers for sequence modeling,” *arXiv preprint arXiv:2208.04933*, 2022.
- [15] A. Gu and T. Dao, “Mamba: Linear-time sequence modeling with selective state spaces,” *arXiv preprint arXiv:2312.00752*, 2023.
- [16] A. Antoniou, *Digital Signal Processing: Signals, Systems, and Filters*. McGraw Hill LLC, 2005.
- [17] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [18] K. Goel, A. Gu, C. Donahue, and C. Ré, “It’s raw! audio generation with state-space models,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 7616–7633.
- [19] S. Ryu, J. Lim, and Y.-S. Lee, “Narrowband active noise control with ddpq based on reinforcement learning,” *International Journal of Automotive Technology*, pp. 1–9, 2024.
- [20] Y.-J. Jang, J. Park, W.-C. Lee, and H.-J. Park, “A convolution-neural-network feedforward active-noise-cancellation system on fpga for in-ear headphone,” *Applied Sciences*, vol. 12, no. 11, p. 5300, 2022.
- [21] Y.-J. Cha, A. Mostafavi, and S. S. Benipal, “Dnoiset: Deep learning-based feedback active noise control in various noisy environments,” *Engineering Applications of Artificial Intelligence*, vol. 121, p. 105971, 2023.
- [22] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu *et al.*, “Wavenet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, vol. 12, 2016.
- [23] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, “Samplernn: An unconditional end-to-end neural audio generation model,” *arXiv preprint arXiv:1612.07837*, 2016.
- [24] D. Dallinger, “Raw audio piano synthesis with structured state space models,” Master’s Thesis, Technische Universität Wien, 2025.
- [25] K. J. Piczak, “Esc: Dataset for environmental sound classification,” in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1015–1018.

- [26] P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” *arXiv preprint arXiv:1804.03209*, 2018.
- [27] Y.-Y. Yang, M. Hira, Z. Ni, A. Chourdia, A. Astafurov, C. Chen, C.-F. Yeh, C. Puhersch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang, J. Lian, J. Mahadeokar, J. Hwang, J. Chen, P. Goldsborough, P. Roy, S. Narenthiran, S. Watanabe, S. Chintala, V. Quenneville-Bélair, and Y. Shi, “Torchaudio: Building blocks for audio and speech processing,” *arXiv preprint arXiv:2110.15018*, 2021.
- [28] T. Sinjanakhom and S. Chivapreecha, “Real-time zero-phase digital filter using recurrent neural network,” in *2023 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, 2023, pp. 348–352.
- [29] Qualcomm, “Qcc5144 bluetooth audio soc,” <https://www.qualcomm.com/audio/products/qcc51xx-series/qcc5144>, accessed: 2026-02-03.