Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Interne Modelle zur Vorhersage der Eigengeräusche bei humanoiden Robotern

Internal models for humanoid robot ego noise prediction

Diplomarbeit

zur Erlangung des akademischen Grades
Diplominformatiker

eingereicht von: Claas-Norman Ritter
geboren am: 18.09.1976
geboren in: Rostock

Gutachter/innen: Prof. Dr. Verena V. Hafner
Prof. Dr. Hans-Dieter Burkhard

eingereicht am: ..... verteidigt am: .....

Inhaltsverzeichnis

Abstract

This thesis is about the predicion of a robots ego noise by using internal models. To predict the ego noise, forward models are implemented as multi layer perceptrons. To represent the acustic features of the ego noise, mel frequency cepstral coefficients are used. Them are learned by multi layer perceptrons, utilizing sensorimotor data as input. The choice of mel frequency cepstral coefficients is motivated by the widely use of mel frequency cepstral coefficients in automatic speech recognition systems. An investigation of differently configured multi layer perceptrons is made, to see, which configurations lead to good predictions of mel frequency cepstral coefficients from sensorimotor data.

The investigation is restricted to one hidden layer perceptrons, because they already are able to approximate any functional relation, if existing [HSW89]. As a result it could be shown, that the usage of just the measured change in the joints position is sufficient to generate a quite good prediction of the first mel frequency cepstral coefficient. It was found too, that coefficients, higher than the third coefficient, seem to have quite bad prediction quality. Nevertheless even the badly predicted higher mel frequency cepstral coefficients do contain relevant information about the underlying spectrum.

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Vorhersage der Eigengeräusche eines Roboters unter Verwendung von internen Modellen. Zur Vorhersage werden Vorwärtsmodelle mit Multi-Layer-Perzeptrons realisiert. Um die akustischen Merkmale der Eigengeräusche zu repräsentieren, werden Mel-Frequency-Cepstral-Coefficients (MFCC) genutzt. Diese werden von Multi-Layer-Perzeptrons gelernt, wobei Sensor- und Motordaten als Grundlage der Vorhersage dienen. Die Wahl von MFCCs ist motiviert durch die verbreitete Nutzung von MFCCs in Spracherkennungssystemen. Eine Untersuchung verschieden konfigurierter Multi-Layer-Perzeptrons wird unternommen, um zu sehen, welcher der Konfigurationen zu einer guten Vorhersage der MFCCs aus den Sensor- und Motordaten führen.

Die Betrachtung beschränkt sich auf einschichtige Multi-Layer-Perzeptrons, weil diese bereits jede funktionale Beziehung abbilden können, falls sie existiert [HSW89]. Als ein Ergebnis kann gezeigt werden, dass die Verwendung der gemessenen Änderung der Gelenkstellung ausreicht, um daraus den ersten MFCC-Koeffizienten vorherzusagen. Es zeigte sich auch, dass die Vorhersage der MFCC-Koeffizienten ab dem vierten Koeffizienten sehr ungenau ist. Trotzdem enthalten diese MFCC-Koeffizienten relevante Informationen über das zugrundeliegende Spektrum.

1 Einleitung und Motivation

In zunehmendem Maße werden Roboter entwickelt, die Menschen bei alltäglichen Arbeiten unterstützen sollen. Roboter die im direkten Umfeld des Menschen eingesetzt werden, müssen in der Lage sein, sich auf sichere und für Menschen intuitive Weise darin bewegen zu können. Eine der dazu nötigen Fähigkeiten ist, mittels Sprache kommunizieren zu können. Roboter werden zur Zeit häufig aus Elementen konstruiert, die weit verbreitet und gut verfügbar sind. Beispiele solcher Elemente sind Elektromotoren, Rechentechnik, Kameras und diverse andere Sensoren. Ein großes Problem bei der Verarbeitung akustischer Informationen ist für derart aufgebaute Roboter, dass einige ihrer Bestandteile laute Geräusche produzieren, die sich sehr störend auf die Fähigkeit eines Roboters auswirken, akustische Informationen zu verarbeiten.

Typische Geräuschquellen bei Robotern sind zur Zeit Lüfter, die zur Kühlung der Elektronik benötigt werden und Elektromotoren, die zur Bewegung der mechanisch beweglichen Teile eines Roboters verwendet werden. Hinzu kommen Geräusche, die durch Interaktion mit Objekten der Umgebung auftreten können, wie zum Beispiel der Aufprall der Füße auf dem Untergrund beim Laufen oder der Zusammenprall von Körperteilen, die sich während einer Bewegung gegenseitig im Wege sind. Die Elektromotoren und Lüfter sind oft eine der markanten und dauerhaft vorhandenen Geräuschquellen bei Robotern. Auch bei dem verwendeten humanoiden Roboter Nao, der im Kapitel 3 kurz vorgestellt wird, sind der Lüfter im Kopf und die Gelenkmotoren sehr dominant in den Audioaufnahmen des Naos vertreten. Bei der Analyse von Audioaufnahmen stellen die enthaltenen Motor- und Lüftergeräusche Störquellen dar, die gewünschte Information im Audiosignal überdecken und unkenntlich machen können.

Ziel dieser Arbeit ist deshalb zu untersuchen, inwiefern sich Eigengeräusche, die ein Roboter selbst produziert, mit internen Modellen abbilden lassen. Die Annahme ist hierbei, dass mit Hilfe der internen Modelle die Effekte der dem Roboter eigenen Störquellen aus den Audiosignalen extrahiert und entfernt werden können, wenn sie durch ein solches Modell genügend gut vorhergesagt werden. Interne Modelle werden in Kapitel 2.2 vorgestellt. Die Untersuchung der internen Modelle beschränkt sich auf Vorwärtsmodelle basierend auf dem neuronalen Multi-Layer-Perzeptron. Eine Einschränkung ist nötig, da eine sehr große Vielfalt an Möglichkeiten existiert, um interne Modelle zu implementieren. Ebenso gibt es ein sehr große Anzahl möglicher Organisationsformen neuronaler Netze. Das ein- oder mehrschichtige Perzeptron ist eine gut untersuchte und verstandene Organisationsform eines neuronalen Netzes. Unter der Annahme eines funktionalen Zusammenhangs zwischen den Bewegungen eines Roboters und den daraus resultierenden Motorgeräuschen sollten einschichtige Multi-Layer-Perzeptrons ausreichend sein, um diesen Zusammenhang abzubilden.

Eine Übersicht zu neuronalen Netzen und dem Perzeptron wird in Kapitel 2.3 behandelt.

Um die erzeugten Geräusche modellieren zu können, die durch eigene Bewegungen entstehen, müssen den im Audiosignal enthaltenen Merkmalen die verursachenden Bewegungen zugeordnet werden. Als Merkmale werden MFCC-Koeffizienten verwendet, da sie sehr kompakt sind, mit ihnen eine im Spektrum multiplikative Verknüpfung von Stör- und Nutzsignal auf eine additive Überlagerung überführt werden kann und auch weil sie in Spracherkennungssystemen eine wichtige Rolle spielen. Die Berechnung der Merkmale mit Standardmethoden der Audiosignalverarbeitung wird in Kapitel 2.1 beschrieben. Für die Zuordnung der berechneten Merkmale aus dem Audiosignal zu den verursachenden Bewegungen ist eine zeitliche Synchronisation der Audio- und Motordaten notwendig. Kapitel 3.2.1 beschreibt, wie eine solche Synchronisierung aussehen kann. In Kapitel 4 wird schließlich untersucht, mit welchen Parametern ein Multi-Layer-Perzeptron in der Lage ist, eine möglichst gute Abbildung der MFCC-Koeffizienten aus den Bewegungen des Roboters zu lernen. Ein zusammenfassende Diskussion und ein Ausblick auf mögliche weiterführende Arbeiten folgen abschließend in Kapitel 5.

1.1 Motivation

Die Idee zu dieser Arbeit entstand im Rahmen meiner Tätigkeit für das Projekt EARS (Embodied Audition For RobotS). Gegenstand des Projektes ist die Untersuchung von Verfahren zur Manipulation von Audiosignalen für eine Verbesserung der Extraktion darin enthaltener nützlicher Informationen, wie beispielsweise Sprache. Dabei stehen verschiedene Ansätze im Fokus des Projektes: die Unterdrückung der Effekte von Störquellen, wie den Eigengeräuschen des Roboters und die Extraktion und Synthese der Anteile gewünschter Informationen in mehreren Audiokanälen aus einer bestimmten räumlichen Richtung mit Hilfe von Source-Separation und Beamforming.

Ein Modell der selbst erzeugten Geräusche kann dazu genutzt werden, die Effekte der Eigengeräusche herauszufiltern und die Anteile anderer eventuell nützlicher Informationen besser erkennbar zu machen. Zum Beispiel ist die für die Mensch-Roboter-Interaktion wichtige Spracherkennung bei Robotern, in einer durch Menschen geprägten Umgebung ein Anwendungsfall, bei dem das Entfernen der Eigengeräusche eine Verbesserung der Spracherkennung bewirken kann. Eine gute Spracherkennung ist für Roboter wichtig, die mit Menschen interagieren, da gesprochene Sprache ein für den Menschen natürliches und häufig genutztes Kommunikationsmittel ist.

Eine mittels der Unterdrückung der Eigengeräusche verbesserte Spracherkennung kann hierbei zu einem natürlicheren und intuitiven Umgang mit Robotern beitragen. Zum Beispiel wenn Kommandos an einen Serviceroboter nicht mehrfach hintereinander wiederholt werden müssen, bevor sie vom Roboter erkannt werden. Eine verbesserte Spracherkennung ist auch notwendig, um den Abstand eines menschlichen Sprechers zu einem Roboter auf eine für Menschen angenehme und natürlichere Di-

stanz, als wenige Zentimeter, zu erhöhen und somit die Akzeptanz von Interaktion mittels Sprache zwischen Mensch und Roboter verbessern.

Neben Sprache können auch zahlreiche andere Informationen über die Umwelt in Audiosignalen enthalten sein, wie beispielsweise die Tok-Geräusche für blinde Menschen an Ampeln oder Motorgeräusche nahender Fahrzeuge und vieles andere mehr. Die Nutzung eines Modells der erzeugten Eigengeräusche, kann auch hier helfen die Erkennung und Einordnung solcher akustischer Informationen zu verbessern, wenn mit dem Modell der Anteil der Eigengeräusche im Audiosignal entfernt werden konnte.

Eine weitere mögliche Anwendung eines Modells der eigenen Geräusche ist mit deren Hilfe Aktionen anderer Roboter in der Nähe zu bestimmen. Um ein Beispiel zu nennen: der Roboter bewegt sich selbst nicht, nimmt aber Geräusche wahr, die ähnlich denen sind, wenn er selbst laufen würde. Ein Modell der Geräusche, die der Roboter selbst beim Laufen erzeugt und das Wissen, dass er selbst gerade nicht läuft, kann dazu genutzt werden, um darauf zu schließen, dass ein anderer Roboter in der Nähe sein müsste, der gerade läuft.

Mit der Annahme, dass sich das Geräusch eines Motors bei Verschleiß oder einem Defekt verändert, ist auch eine Art akustisches Selbst-Monitoring denkbar. In diesem Anwendungsfall würde der Roboter die erzeugten Geräusche seiner Motoren mit einem vorhandenen Modell dieser Eigengeräusche abgleichen, um Änderungen, wie beispielsweise Unregelmäßigkeiten, auf akustischem Weg zu ermitteln.

1.2 Verwandte Arbeiten

Dass Roboter ihre Fähigkeit aus Audiosignalen Informationen zu gewinnen durch selbstverursachte Geräusche stören, ist ein in der Robotik bekanntes Problem. Verschiedene Ansätze diesem Problem entgegenzutreten, wurden deshalb in den vergangenen Jahren untersucht. Dabei war sehr oft die Reduktion von Störgeräuschen für die Spracherkennung im Fokus der Arbeiten. Zur Spracherkennung werden unter anderem Mel-Frequency-Cepstral-Coefficents (MFCC) eingesetzt. MFCC-Koeffizienten sind aber nicht nur zur Klassifizierung und Erkennung von Sprache geeignet.

Auch andere akustische Phänomene lassen sich mit Hilfe von MFCC-Koeffizienten klassifizieren, wie [SMD10] am Beispiel der Klassifizierung von Musikinstrumenten zeigt. Zur Erkennung von Sprache werden recht häufig probabilistische Verfahren eingesetzt, um den Anteil des Eigenanteils der Störgeräusche im Audiosignal abzuschätzen.

So beschreiben [INR+11] die Unterdrückung der Motorgeräusche beim Roboter ASIMO auf mehrkanaligen Audioaufnahmen. Sie verwenden ein Spracherkennungssystem auf Basis der Missing Feature Theory (MFT), bei der ein Hidden-Markov-Modell (HMM) genutzt wird, um die fehlenden Sprachmerkmale zu schätzen. Es werden auch adaptive Verfahren zusätzlich zu einer MFT-ASR (Missing Feature Theory-Automated Speech Recognition) genutzt, um die Erkennung von Sprache zu verbessern.

In [NNN+06] werden Multi-Condition Training und Maximum-Likelihood-Linear-Re-

gression (MLLR) zu Adaption an den Störgeräuschanteil der Audiosignale, als ein Vorverarbeitungsschritt zur MFT-ASR, genutzt. Informationen über die durch Eigenbewegung verursachten Störgeräusche werden nur über eine Gewichtung der Parameter für die MFT-ASR, selektiert aus gespeicherten Vorlagen der Eigengeräusche, miteinbezogen.

Eine Unterdrückung der selbst erzeugten Störgeräusche durch direkte Vorhersage von Koeffizienten des logarithmierten Leistungsspektrums nehmen [IKSM10] vor. Hier werden die Koeffizienten des logarithmierten Leistungsspektrums aus den Gelenkstellungen mehrerer aufeinanderfolgender Zeitpunkte durch ein Multi-Layer-Perzeptron vorhergesagt und vom Spektrum des Audiosignals abgezogen.

1.3 Struktur

Im ersten Kapitel werden ein Einführung und Motivation gegeben.

Kapitel 2 behandelt die notwendigen theoretischen Grundlagen. Es enthält die Grundlagen zur Signalverarbeitung, etwa darüber wie MFCC-Merkmale gewonnen werden und was unter internen Modellen und neuronalen Feedforward Netzen, speziell Multi-Layer-Perzeptrons, zu verstehen ist.

Kapitel 3 enthält eine Beschreibung der verwendeten Hardware und eine kurze Beschreibung der benötigten Synchronisierung der Datenkanäle.

Kapitel 4 untersucht mit welchen Konfigurationen von Hyper-Parametern eines Multi-Layer-Perzeptrons eine möglichst gute Herleitung der MFCC-Koeffizienten aus den vorhandenen Sensor- und Motordaten gelingt.

In Kapitel 5, dem letzten Kapitel, werden die Ergebnisse der Untersuchungen diskutiert.

2 Theoretische Grundlagen

In diesem Kapitel werden die notwendigen theoretischen Grundlagen behandelt. Zunächst werden dazu im Abschnitt 2.1 die benötigten Grundlagen der Signalverarbeitung von Audiosignalen und der Berechnung von Merkmalen aus Audiosignalen behandelt. Anschließend werden interne Modelle im Abschnitt 2.2 kurz vorgestellt und zum Schluss dieses Kapitels wird in Abschnitt 2.3 auf neuronale Netze, speziell Feedforward Netze, eingegangen.

2.1 Analyse von Merkmalen in Audiosignalen

Die Analyse von Audiosignalen wird meist im Bereich eines Frequenzspektrums der Audiosignale durchgeführt. Einige der üblichen Transformationen eines Zeitsignals in ein Frequenzspektrum werden im dem folgenden Abschnitt 2.1.1 behandelt. Bei Systemen zur Spracherkennung ist die Arbeit mit cepstralen Merkmalen eine der Standardtechniken.

Viele aktuelle Spracherkennungsalgorithmen verwenden Mel-Frequency-Cepstral-Coefficents (MFCC). Die MFCCs sind eine Annäherung an die Prinzipien, die bei der Erzeugung und der Verarbeitung von Sprache beim Menschen zu finden sind. Näheres dazu wird in den Abschnitten 2.1.4 und 2.1.4 vorgestellt. Zwei zum Thema der Signalverarbeitung gute Bücher sind [MH04] und [Nie83]. Sie bilden die Grundlage für die Ausführungen in den Abschnitten 2.1.1 bis 2.1.4.

2.1.1 Frequenzanalyse

Zur Abbildung eines Audiosignals in den spektralen Bereich sind Reihenentwicklungen mit orthogonalen Funktionen und deren Verallgemeinerung, die kontinuierlichen orthogonalen Transformationen, ein oft genutztes Werkzeug. Die komplexe Fourier-Transformation ist dabei oftmals die erste Wahl. Abhängig vom untersuchten Signal und den erwünschten Eigenschaften des Spektrums sind Transformationen, wie die Sinus- und Kosinus-Transformation, die Walsh-Transformation, die Hartley-Transformation und weitere orthogonale Transformationen eine Möglichkeit ein Zeitsignal in ein Spektrum zu überführen. Im spektralen Bereich sind oftmals Eigenschaften des Zeitsignal erkennbar, die im Zeitbereich nicht erkenntlich sind.

Im Unterschied zur Fourier-Transformation bzw. Fourier-Reihenentwicklung ergeben die anderen genannten Transformationen bzw. Reihenentwicklungen nicht-komplexe Spektren. Die Walsh-Transformation und die Walsh-Reihenentwicklung basieren auf rechteckförmigen Basisfunktionen im Intervall [-1,1] und eignen sich daher gut für die

Analyse diskreter und digitaler bzw. rechteckförmiger Signale. Die Hartley-Transformation und Hartley-Reihenentwicklung basieren auf den harmonischen Funktionen Sinus und Kosinus.

Die Fourier-Transformation und die Kosinus-Transformation werden im folgenden Unterkapitel 2.1.2 näher betrachtet. Beide sind notwendig, um Mel-Frequenz-Cepstren zu berechnen.

2.1.2 Fouriertransformation

Die Fourier-Transformation $F(\omega)$ ist die am häufigsten verwendete Transformation eines Orts- oder Zeitsignal in dessen Frequenzspektrum. Für den kontinuierliche Fall ist die Fourier-Transformierte $FT\{f(t)\}$ definiert durch (vgl. [Nie83, S.63]):

$$FT\{f(t)\} = F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-i \cdot \omega \cdot t} dt \tag{2.1}$$

Die Fourier-Transformation kann auf in Ort oder Zeit kontinuierliche und diskrete Signale angewendet werden. Die Betrachtungen der folgenden Kapitel beschäftigen sich hauptsächlich mit der Verarbeitung und Vorhersage von Merkmalen aus Audiosignalen. Deshalb beschränkt sich die Betrachtung der Fourier-Transformation und ihrer Verwandten auf die Verwendung von Signalen des Zeitbereiches. Das Ergebnis der Fourier-Transformation, die Fourier-Transformierte, ist dabei immer ein kontinuierliches Frequenzspektrum[1]. Voraussetzung für die Existenz der Fourier-Transformierten ist die absolute Integrierbarkeit des Signals (vgl. [MH04, S.145]):

$$\int_{-\infty}^{\infty} |f(t)| dt < \infty \tag{2.2}$$

Eine wichtige Eigenschaft der Fourier-Transformation ist ihre Umkehrbarkeit. Die Anwendung der inversen Transformation auf das Spektrum eines Orts- bzw. Zeitsignal rekonstruiert das ursprüngliche Signal, falls keine das Spektrum verändernden Operationen durchgeführt wurden. Die inverse Fourier-Transformation ist definiert durch (vgl. [Nie83, S.63]):

$$FT\{F^{-1}(\omega)\} = f(t) = \frac{1}{2 \cdot \pi} \int_{-\infty}^{\infty} F(\omega) \cdot e^{i \cdot \omega \cdot t} d\omega \tag{2.3}$$

Das Resultat der Fourier-Transformation und ihrer Inversen ist immer periodisch. Da jede Periode den gleichen Informationsgehalt hat, kann eine Analyse oder Verarbeitung auf jeweils eine Periode beschränkt werden.

[1] Im Unterschied dazu ergibt die Fourier-Reihenentwicklung ein Linienspektrum.

Ist ein Signal, das mit der Fourier-Transformation transformiert werden soll, nicht periodisch, so kann man dieses als periodisch fortgesetzt annehmen, falls das periodisch fortgesetzte Signal absolut integrierbar ist und somit auch eine endliche Energie hat (vgl. [MH04, S.147]):

$$\int_{-\infty}^{\infty} |f(t)|^2 dt < \infty \tag{2.4}$$

Diskrete Fouriertransformation (DFT)

Bei Anwendung der Fourier-Transformation auf ein diskretes Signal $f_n = f(nT_A)$ mit dem Abtastintervall T_A können die Integrale durch Summen ersetzt werden. Somit sind die Bedingung für die Existenz der Fourier-Transformierten im Fall diskreter Signale (vgl. [MH04, S.150]):

$$\sum_{n=-\infty}^{\infty} |f_n| < \infty \text{ und damit auch: } \sum_{n=-\infty}^{\infty} |f_n|^2 < \infty \tag{2.5}$$

Die Fourier-Transformierte ist dann definiert durch (vgl. [MH04, S.150]):

$$FT\{f_n\} = F(\omega) = \sum_{n=-\infty}^{\infty} f_n \cdot e^{-i \cdot \omega \cdot t} \tag{2.6}$$

Und ihre Inverse durch (vgl. [MH04, S.150]):

$$FT\{F^{-1}(\omega)\} = f_n = \frac{T_A}{2 \cdot \pi} \sum_{n=-\infty}^{\infty} F(\omega) \cdot e^{i \cdot \omega \cdot n \cdot T_A} \tag{2.7}$$

Um auch ein diskretes Spektrum zu erhalten, muss auch das kontinuierliche Spektrum abgetastet werden. Dies erfolgt in den Abständen (vgl. [MH04, S.158]):

$$\Delta\omega = 2 \cdot \pi = \frac{2 \cdot \pi}{N \cdot T_A} \tag{2.8}$$

Damit kann die diskrete Fourier-Transformation (DFT) abgeleitet werden und ergibt sich zu (vgl. [MH04, S.159]):

$$\begin{aligned} DFT\{f_n\} = F(\omega_m) &= F\left(\frac{2 \cdot \pi \cdot m}{N \cdot T_A}\right) \qquad \text{mit } m = 0, 1, ..., N-1 \\ &= F_m = \sum_{n=0}^{N-1} f_n \cdot e^{-i \cdot 2 \cdot \pi \cdot \frac{m \cdot n}{N}} \end{aligned} \tag{2.9}$$

Für die inverse diskrete Fourier-Transformation (DFT^{-1} bzw. IDFT) folgt damit (vgl. [MH04, S.159]):

$$\begin{aligned} IDFT\{F_m\} &= DFT^{-1}\{F_m\} && \text{mit } n = 0, 1, ..., N-1 \\ &= f_n = \frac{1}{N} \sum_{m=0}^{N-1} F_m \cdot e^{i \cdot 2 \cdot \pi \cdot \frac{m \cdot n}{N}} \end{aligned} \tag{2.10}$$

Wie man den an den Formel leicht sieht, erfolgt die Berechnung eines komplexen Spektrums mittels der Fourier-Transformation in einem von der Anzahl der Abtastwerte N abhängigen Aufwand von $\mathcal{O}(N^2)$.

Schnelle Fouriertransformation (FFT)

Bei sehr langen Signalen oder sehr kleinen Abtastabständen ist die Anwendung der diskreten Fourier-Transformation eventuell zu aufwändig. Unter Ausnutzung der Symmetrie und Periodizität der Fourier-Transformation können Algorithmen hergeleitet werden, die eine schnellere Berechnung der diskreten Fourier-Transformation ermöglichen. Ein Beispiel für eine schnelle diskrete Fourier-Transformation (FFT) ist der Cooley-Tukey-Algorithmus. Mit dem Cooley-Tukey-Algorithmus kann der Berechnungsaufwand auf $\mathcal{O}(N \cdot ldN)$ reduziert werden. Voraussetzung dafür ist, dass N eine Zweierpotenz $N = 2^n$ ist. Für den Fall, dass man ein diskretes Signal f'_n mit $N' \neq 2^n$ Abtastwerten transformieren möchte, so kann das Signal mit Nullen bis zur nächsten größeren Zweierpotenz $N' < N = 2^n$ aufgefüllt werden und dann die FFT darauf angewendet werden:

$$f_n = \begin{cases} f'_n & , 0 \leq n < N' \\ 0 & , N' \leq n < N \end{cases} \tag{2.11}$$

Der Cooley-Tukey-Algorithmus ist einer der ersten Algorithmen für eine schnelle Fourier-Transformation, die gefunden und im Laufe der Zeit weiterentwickelt wurden. Es wurden auch Algorithmen entwickelt, die nicht nur die Eigenschaften der Fourier-Transformation, sondern auch die numerischen Eigenheiten der Rechnersysteme ausnutzen, auf denen die Algorithmen ausgeführt werden.

Ob und wie diese Eigenheiten ausgenutzt werden hat Einfluss auf die Ausführungsgeschwindigkeit und die Genauigkeit der Fourier-Transformation. Aus diesem Grund werden in Kapitel 3.2.2 verschiedene existierende Implementierungen von Algorithmen der schnellen Fourier-Transformation untersucht.

Diskrete Kosinus-Transformation (DCT)

Aus der Fourier-Transformation kann mit Hilfe der eulerschen Identität ein Zusammenhang zu Kosinus-Transformation und Sinus-Transformation hergeleitet werden. Unter Beachtung der Symmetrie-Eigenschaften der Fourier-Transformation und unter Verwendung der Eulerschen Identität:

$$e^{j \cdot x} = cos(x) + j \cdot sin(x) \tag{2.12}$$

kann die Fourier-Transformation in ihre geraden f_g und ungeraden f_u Anteile zerlegt werden (vgl. [MH04, S.148f]):

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} f_g(t) \cdot cos(\omega \cdot t)dt - i \cdot \int_{-\infty}^{\infty} f_u(t) \cdot sin(\omega \cdot t)ds \\ &= F_g(\omega) - i \cdot F_u(\omega) \end{aligned} \tag{2.13}$$

An dieser Definition der Fourier-Transformation kann man sehen, dass die Kosinus-Transformation als Realanteil in der Fourier-Transformation enthalten ist.

Die Kosinus-Transformation ist definiert durch:

$$KT\{f(t)\} = F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot cos(\omega \cdot t)dt \tag{2.14}$$

Für die Berechnung der Kosinus-Transformation wird wie bei der Fourier-Transformation das zu transformierende Signal periodisch erweitert. Allerdings gibt es einen Unterschied in der Art und Weise wie diese periodische Fortsetzung erfolgt. Bei der Kosinus-Transformation wird das nicht-periodische Signal gespiegelt angehängt, anstatt es einfach nur anzuhängen.

Durch diese Spiegelung entspricht die Kosinus-Transformation dem Realteil der Fourier-Transformation die auf ein periodisches Signal doppelter Periodenlänge angewendet wird, dass durch Spiegelung einer Periode und dessen periodischer Fortsetzung erzeugt wurde. Bei der diskreten Kosinus-Transformation kann diese Spiegelung in gerader und ungerader Form vorgenommen werden. Daher kann die diskrete Kosinus-Transformation im Detail verschieden definiert sein.

Ein mögliche Definition der diskreten Kosinus-Transformation ist gegeben durch (vgl. [Nie83, S.177]):

$$DCT\{f_n\} = F_m = \begin{cases} \dfrac{\sqrt{2}}{N} \displaystyle\sum_{n=0}^{N-1} f_n & , m = 0 \\ \dfrac{2}{N} \displaystyle\sum_{n=0}^{N-1} f_n \cdot cos\left(\dfrac{2 \cdot \pi \cdot (n+1) \cdot m}{2 \cdot N}\right) & , m = 1, 2, ..., N-1 \end{cases} \tag{2.15}$$

Die Kosinus-Transformation auf dem gespiegelt periodischen Signal doppelter Länge anzuwenden, birgt einige Vorteile. Beispielsweise werden dadurch Sprünge in Signal vermieden, die ansonsten falsche Frequenzanteile im Spektrum erzeugen würden. Eine weitere wichtige Eigenschaft der Kosinus-Transformation ist eine hohe energetische Dichte in den niederen Koeffizienten des Spektrums. Zudem eignet sie sich zur Dekorrelation eines Signals, da sie eine gute Approximation an die Karhunen-Loewe-Transfomation (KLT) ist. Auf Grund dieser Eigenschaften wird die diskrete Kosinus-Transformation oft zur Datenkompression und bei der Mustererkennung eingesetzt.

Auch für die diskrete Kosinus-Transformation gibt es schnelle Algorithmen, die oftmals mit der FFT zusammen implementiert verfügbar sind. In Bezug auf Ausführungsgeschwindigkeit und Genauigkeit der verschiedenen existierenden Implementierungen sind ähnliche Bedingungen gültig wie für die FFT, weshalb in Kapitel 3.2.2 auch die Implementierungen der reellen FFT-Algorithmen verglichen werden, die unter anderem die Kosinus-Transformation abbilden.

2.1.3 Kurzzeit-Transformation

Im folgenden werden die Betrachtungen weiterhin auf Zeitsignale beschränkt sein, sie gelten aber auch für 1D-Signale des Ortsbereiches. Ebenso kann der Begriff Frequenz, so er verwendet wird, gegen den Begriff Sequenz ausgetauscht werden, etwa bei Anwendung der diskreten Walsh-Transformation.

In der Praxis können unendlich lange Signale aufgrund begrenzter Verarbeitungsressourcen nicht im Ganzen betrachtet werden. Diese müssen also über eine beschränkte Dauer T_B beobachtet werden. Weiterhin kann aus einer auf das gesamte endliche Signal angewandten orthogonalen Transformation nur ein Aussage über die insgesamt darin enthaltenen Frequenzen vorgenommen werden. Es kann aber keine Aussage darüber getroffen werden, zu welchem Zeitpunkt, welche Frequenzen enthalten sind, falls die Verteilung der Frequenzen im Spektrum des Signals über die Zeit veränderlich ist.

Es ist also sinnvoll das Signal in Abschnitte zu zerlegen und diese getrennt zu betrachten. Für diskrete Transformationen gilt die folgende Beziehung (vgl. [MH04, S.180]):

$$N \cdot \Delta t \cdot \Delta f = const \tag{2.16}$$

Aus dieser Beziehung folgt, dass das Ergebnis einer diskreten Transformation kein scharfer Punkt in der durch die Zeit t und die Frequenz f aufgespannten Ebene sein kann, sondern einem Bereich der Größe $\Delta t \cdot \Delta f$ entspricht. Bei einer Beobachtungsdauer von $T_B = N \cdot T_A$ kann für Δf (vgl. [MH04, S.180]):

$$\Delta f = \frac{1}{T_B} = \frac{1}{N \cdot T_A} \tag{2.17}$$

hergeleitet werden. Der Abtastabstand T_A ist durch das Abtasttheorem und ebenso durch die Hardware des Aufnahmesystems bestimmt. Das Abtasttheorem besagt,

dass die Abtastfrequenz f_A mindestens doppelt so hoch sein muss, wie die höchste im Signal enthaltene Frequenz, um Aliasing-Effekte zu vermeiden. Die Hardware des Aufnahmesystems ist deshalb ein einschränkender Faktor, weil sie die maximal mögliche Abtastfrequenz bestimmt. Die Auflösung des Spektrums kann also nur noch über die Wahl der Größe von N beeinflusst werden. Dabei sollte N so groß gewählt werden, dass eine gute spektrale Auflösung erreicht wird, ohne Frequenzanteile zu verlieren oder zu verschmieren. Gleichzeitig sollte N so klein gewählt werden, dass die Berechnung nicht unnötig aufwendig wird und eine gute zeitliche Auflösung gewährleistet ist.

Außer der Größe hat auch die Position des Zeitfensters, abhängig von der Form des Signals, unter Umständen negative Auswirkungen auf das Spektrum nach einer diskreten Transformation. Bei periodischen Signalen sollte das Zeitfenster ein Vielfaches einer Periode des Signal abdecken, um Diskontinuitäten an den Grenzen des Zeitfensters zu vermeiden. Zufallssignale können mit einer Fensterfunktion, die Diskontinuitäten an den Grenzen des Zeitfensters vermeidet, multipliziert werden, um Artefakte im Spektrum zu mindern.

Die einfachste Fensterfunktion ist die Rechteckfunktion, die implizit angewandt wird, wenn einfach nur ein Teilstück eines Signals betrachtet wird. Der große Nachteil des Rechteckfensters ist, dass Diskontinuitäten an den Grenzen und damit Artefakte im Spektrum mit hoher Wahrscheinlichkeit auftreten. Hanning- (von-Hann) und Hammingfunktionen bieten eine wesentliche bessere Vermeidung von Diskontinuitäten an den Grenzen des Zeitfensters. Beide basieren auf einer Kosinus-Fensterfunktion (vgl. [MH04, S.183]):

$$f_n = \alpha + (1-\alpha) \cdot cos\left(\frac{2 \cdot \pi \cdot t_n}{N \cdot T_A}\right), \text{ mit } |t_n| < \frac{1}{2} \cdot N \cdot T_A \tag{2.18}$$

Sie unterscheiden sich lediglich in der Wahl des Faktors α. Der Faktor α ist bei dem Hanningfenster auf 0,5 und beim Hammingfenster auf 0,54 festgelegt. Eine weitere oft angewandte Fensterfunktion ist die Gaborfensterfunktion, die auf der Nutzung einer Gauß-Glocke basiert.

Eine Fensterung des Signals kann direkt in eine Transformationsgleichung integriert werden. Setzt man z.B. in der Gleichung der kontinuierlichen Fourier-Transformation für $f(t)$ das Produkt $f(t) = f'(t) \cdot g(t-\tau)$ aus dem Originalsignal $f'(t)$ mit einer von dem Zeitpunkt τ abhängigen Fensterfunktion $g(\tau)$, so erhält man die Gleichung für die kontinuierliche Kurzzeit-Fourier-Transformation (vgl. [MH04, S.194]):

$$STFT(\omega, \tau) = \int_{-\infty}^{\infty} f'(t) \cdot g(t-\tau) \cdot e^{-i \cdot \omega \cdot t} dt \tag{2.19}$$

Das Ergebnis der Kurzzeit-Fourier-Transformation (STFT) ist dann das komplexe Spektrum der Umgebung von τ. Durch Variation von τ kann das gesamte Signal, auf einen von dem gewählten Zeitpunkt τ abhängigen Frequenzgehalt, analysiert werden.

2.1.4 Spektrum eines Spektrums (Cepstrum)

Wendet man eine lineare Transformation auf das Spektrum eines Signals an, wird das resultierende Spektrum eines Spektrums auch Cepstrum genannt. Im cepstralen Bereich können Merkmale von Signalen getrennt werden, die im spektralen Bereich nicht oder nur sehr schwer trennbar sind. Angelehnt an [Nie83, S.171ff] werden im folgenden am Beispiel der diskreten Fourier-Transformation die Beweggründe einer Verwendung von Cepstren näher betrachtet.

Der Merkmalsanalyse im spektralen Bereich liegt die Annahme zugrunde, dass sich das zu analysierenden Signal f aus einer additiven Überlagerung $f = f_s + f_n$ aus Störsignal f_s und Nutzsignal f_n zusammensetzt. Die Koeffizienten der diskreten Fourier-Transformation überlagern sich dann auch additiv:

$$DFT\{f\} = DFT\{f_s\} + DFT\{f_n\} = F_s + F_n \tag{2.20}$$

Also können Störsignal f_s und Nutzsignal f_n mit einer linearen Transformation, wie der diskreten Fourier-Transformation, voneinander getrennt werden. Ist diese Überlagerung nicht-additiv, d.h entspricht die Überlagerung von Störsignal f_s und Nutzsignal f_n einer Faltung $f_* = f_s * f_n$, reicht eine lineare Transformation allein nicht aus, denn die Koeffizienten des Störsignals sind dann mit den Koeffizienten des Nutzsignals multiplikativ verknüpft:

$$DFT\{f_*\} = DFT\{f_s\} \cdot DFT\{f_n\} = F_s \cdot F_n \tag{2.21}$$

Um die Merkmale von Stör- und Nutzsignal dennoch zu trennen, kann das Cepstrum f_c für eine Merkmalsextraktion und -analyse genutzt werden. Unter Verwendung des Logarithmus wird die multiplikative Verknüpfung der Koeffizienten in eine Additive umgeformt:

$$log(DFT\{f_*\}) = log(DFT\{f_s\}) + log(DFT\{f_n\}) \tag{2.22}$$

Anschließend werden durch Anwendung einer linearen Transformation die Merkmale des Störsignals und des Nutzsignals im cepstralen Bereich voneinander trennbar. Da das Spektrum der diskreten Fourier-Transformation komplex ist, muss auch der komplexe Logarithmus verwendet werden. Die auf das logarithmierte Spektrum angewandte Transformation ist in diesem Fall die inverse diskrete Fourier-Transformation:

$$f_c = DFT^{-1}\{log[DFT\{f_*\}]\} \tag{2.23}$$

Das Resultat ist das komplexe Cepstrum f_c. Oft sind die gesuchten Merkmale nicht oder nur geringfügig von der im Imaginäranteil enthaltenen Phaseninformation abhängig.

Wenn zudem die Klassifizierung von Merkmalen im Vordergrund steht bzw. eine Signalrekonstruktion nicht vorgesehen ist, kann statt des komplexes Cepstrums f_c auch das reelle Cepstrum f_{c_0} verwendet werden:

$$f_{c_0} = DFT^{-1}\{log[|DFT\{f_*\}|^2]\} \tag{2.24}$$

Die Koeffizienten $F_m = DFT\{f_*\}$ des Spektrum ergeben dann die Koeffizienten des reellen Cepstrums (vgl. [Nie83, S.172]):

$$c_m = DFT^{-1}\{log[|F_m|^2]\} \tag{2.25}$$

Üblich für c_m sind auch:

$$c_m = log[|F_m|^2] \tag{2.26}$$

und:

$$c_m = |log[|F_m|^2]|^2 \tag{2.27}$$

Die Folge $[|F_m|^2]$ ist das Leistungsspektrum von f und entspricht der DFT der Autokorrelationsfunktion von f. Es besteht also ein Zusammenhang zwischen der Autokorrelationsfunktion von f und dem reellen Cepstrum f_{c_0}. Offensichtlich gehen in das reelle Cepstrum Real- und Imaginäranteil des Spektrums mit ein. Welchen Beitrag, welcher Anteil liefert, kann anhand des reellen Cepstrum aber nicht mehr unterschieden werden.

MFCC (Mel Frequency Cepstral Coefficients)

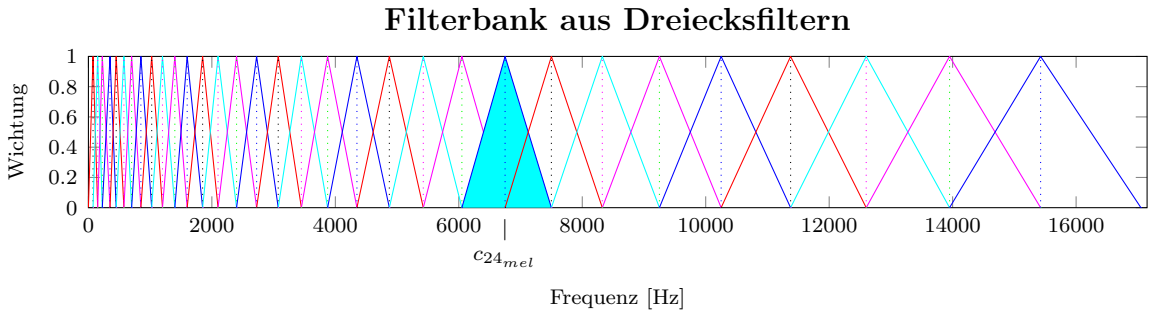


Abbildung 2.1: *Filterbank aus 32 in der Mel-Frequenzskala gleich großen Dreiecksfiltern abgebildet auf die physikalische Frequenzskala von 0 bis ca. 16kHz*

Die Koeffizienten eines Cepstrums zu verwenden, ist seit einiger Zeit ein oft genutzter Ansatz zur Merkmalsgewinnung bei Spracherkennungsverfahren. Für den Anwendungsfall der Spracherkennung kommen verschiedene Formen von Cepstren zum Einsatz, wie u.a. LPCCs (Linear Prediction Cepstral Coefficents) und MFCCs (Mel Frequency Cepstral Coefficents). MFCC's haben sich dabei als sehr gut geeignet erwiesen, Sprache zu verarbeiten.

Die Standardvorgehensweise zur Berechnung von MFCCs ist nach [Nie83, S.209ff]:

1. Es wird das Leistungsspektrums $L_{Hz} = |DFT\{s_t\}|^2$ von einem Ausschnitt s_t des Signals s berechnet.
2. Auf das Leistungsspektrums L_{Hz} wird eine Filterbank aus K Filtern angewendet. Die Filterbank besteht meist aus Dreiecksfiltern, es können aber auch andere Filter, wie Trapez- oder Rechteckfilter genutzt werden. Die Filter werden halb überlappend in der Mel-Frequenzskala äquidistant und mit gleicher Breite berechnet. Die Mel-Frequenzskala ist die Skala einer empirisch ermittelten Größe, die die subjektiv empfundene Tonhöhe repräsentiert, auch melodische Tonheit (Mel) genannt. Eine Annäherung an die nicht-lineare Beziehung zwischen physikalischer Tonhöhe f_{Hz} und subjektiv empfundener melodischer Tonheit f_{mel} ist gegeben durch ([Nie83, S.214]):

   $$f_{mel} = 2595 \cdot log\left[1 + \frac{f_{Hz}}{700}\right] \tag{2.28}$$

   Jeder Filter der Filterbank berechnet einen neuen Koeffizienten $c_{k_{mel}}$, der der Fläche unter dem jeweiligen Filter auf der physikalischen Frequenzskala entspricht. Von niedrigen zu hohen Frequenzen wächst die durch einen Filter abgedeckte Fläche in der physikalischen Frequenzskala (siehe Abbildung 2.1).
3. Anschließend werden die Koeffizienten $c_{k_{mel}}$ logarithmiert.
4. Auf die logarithmierten Koeffizienten $log(c_{k_{mel}})$ wird zum Schluss die diskrete Kosinus-Transformation angewandt:

   $$c_{k_{MFCC}} = DCT\{log(c_{k_{mel}})\} \text{ mit } k = 0, 1, ..., K-1 \tag{2.29}$$

Für Berechnung der MFCC's sind einige die resultierenden Koeffizienten beeinflussende Parameter einzustellen, die entsprechend dem vorgesehenen Anwendungsfall gewählt werden können:

- Auswahl der Fensterfunktion und ihrer Parameter
- Auswahl der Anzahl, des Typs und der Größe der Filter der Filterbank
- Auswahl der Anzahl der zu nutzenden Koeffizienten.

2.2 Interne Modelle

In der Psychologie wird seit einigen Jahren die Annahme, dass Kognition durch amodale abstrakte Symbole im menschlichen Gehirn repräsentiert wird, durch Vertreter der Grounded Cognition als nicht zutreffend angesehen. In der Grounded Cognition steht die Annahme im Vordergrund, dass sensorische, motorische und mentale Erfahrungen in multimodalen Repräsentationen im Gehirn abgebildet sind [Bar]. Diese aus Erfahrungen gelernten multimodalen Repräsentationen bilden im Konzept der Grounded Cognition die Basis für die Simulation von zu erwartender Erfahrung bei bestimmten Aktionen, bevor eine Aktion durchgeführt wird oder auch ohne das eine Aktion durchgeführt wird. Ein Beispiel für eine solche multimodale Repräsentation ist die Vorstellung bzw. das mentale Bild, dass ein Mensch von einem Alltagsgegenstand wie einer Couch oder einem Tisch hat. Darin enthalten sind: wie sieht ein Tisch oder eine Couch aus, wie fühlt es sich an, was kann man damit machen, welche Empfindungen und Gefühle verbinden sich damit und so weiter.

Bei Untersuchungen mit Menschen, die eines der seltenen neurologischen Defizite: den Verlust der Tiefenwahrnehmung und den Verlust das Tastsinnes der Haut haben, wurden Verhaltensunterschiede zu gesunden Menschen festgestellt. Die in [BCPK05] festgestellten Verhaltensunterschiede weisen auf die Existenz der Simulation von zu erwartender, eigener Wahrnehmung aus früheren Erfahrungen hin, um die Erwartungshaltung anderer Personen bei einer Handlung einschätzen zu können, während diese Personen beobachtet werden. Diese Untersuchungen unterstützen die Annahmen der Grounded Cognition.

Im Bereich der Computational Neuroscience und der künstlichen Intelligenz können multimodale Repräsentationen mit Kombinationen interner Modelle abgebildet werden. Interne Modelle bilden die Zusammenhänge zwischen Aktionen und Beobachtungen bzw. Beobachtungen und Aktionen ab. Interne Modelle kann man in zwei Gruppen einteilen: Vorwärtsmodelle und Rückwärtsmodelle.

Erkenntnisse der Computational Neurosience unterstützen die Theorie der Grounded Cognition ebenfalls. Die in den natürlichen neuronalen Strukturen des Cerebellums ablaufenden Prozesse, die bei der Kontrolle und der Korrektur von Fehlern, bei motorischen Ansteuerungen, zu finden sind, zum Beispiel bei Augen- oder Armbewegungen, lassen sich sehr gut mit internen Modellen erklären [WGJ95][WMK98]. Dies weist darauf hin, dass die neuronalen Strukturen im Cerebellum eine natürliche Realisierung der Funktionsprinzipien interner Modelle darstellen.

2.2.1 Vorwärtsmodell

Vorwärtsmodelle bilden bekannte Zusammenhänge zwischen eigenen Aktionen und den Veränderungen ab, die aus den Aktionen resultieren. Die Zusammenhänge können a priori gegeben oder durch Lernverfahren angeeignet sein. Mit einem Vorwärtsmodell kann abgeschätzt werden, welchen Effekt eine Aktion hätte, würde diese ausgeführt.

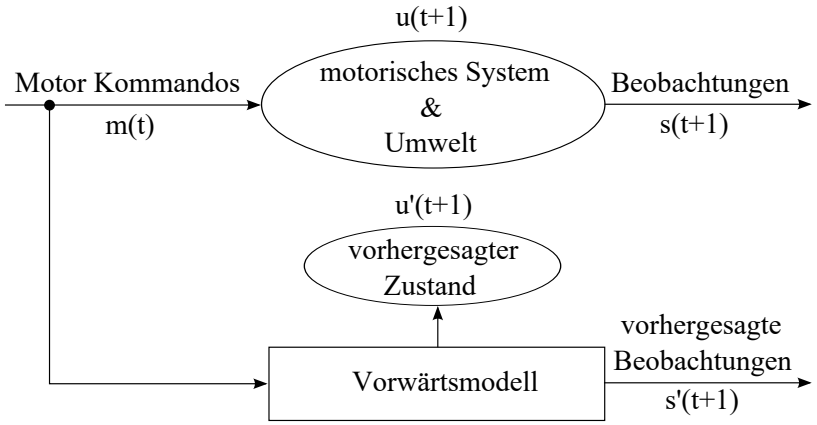


Abbildung 2.2: *Schema eines Vorwärtsmodells bei einem Roboter; aus [DD05] (modifiziert)*

Bezogen auf die Anwendung interner Modelle bei Robotern, berechnet ein Vorwärtsmodell beispielsweise aus Motorkommandos eine Vorhersage der daraus resultierenden sensorischen Beobachtung. Mit einem solchen Vorwärtsmodell ist es dann möglich eine Aussage zu den zu erwartenden Beobachtungen über den System- und Umweltzustand zu treffen, bevor durch Anwendung eines Motorkommandos eine Änderung dieser Zustände tatsächlich eintritt. Vorwärtsmodelle können dazu genutzt werden, Simulationen der Konsequenzen von Aktionen durchzuführen, um anhand der vorhergesagten Konsequenzen eine optimale Aktion je nach Problemstellung auszuwählen.

Zum Beispiel beschreibt [DD05] die Anwendung von Vorwärtsmodellen, um einen ActiveMedia Peoplebot lernen zu lassen, welche visuellen Effekte auftreten, wenn er seine Greifer öffnet und schließt. Das Vorwärtsmodell wird dabei mit einem Bayes'sches Netz realisiert.

2.2.2 Rückwärtsmodell

Rückwärtsmodelle sind quasi die Umkehrung der Vorwärtsmodelle. Sie modellieren den Zusammenhang zwischen der Beobachtung einer Änderung des Umwelt- bzw. Systemzustandes und der Aktion, die zu diesem Ergebnis geführt hat. Mit einem Rückwärtsmodell kann aus einer Beobachtung eine bekannte Aktion zugeordnet werden, deren Ausführung eine Beobachtung zur Folge haben würde, die der aktuellen Beobachtung entspricht.

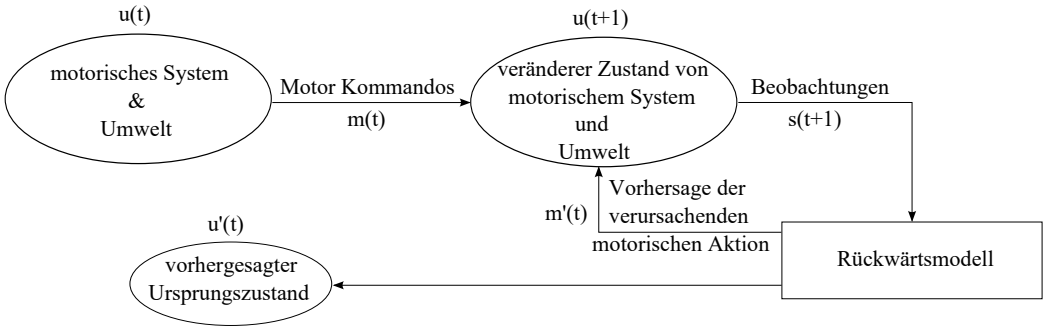


Abbildung 2.3: *Schema eines Rückwärtsmodells bei einem Roboter*

Ein Beispiel der Realisierung eines Rückwärtsmodells wird von [DD05] beschrieben. Dabei wird die Umkehrung des im Abschnitt zuvor aufgeführten Vorwärtsmodells genutzt, um die motorische Aktion auszuführen, die durch visuelle Simulation des Öffnens und Schließens eines Greifers mit zwei menschlichen Händen dem Roboter vorgespielt wird.

2.2.3 Kombination von Vorwärts- und Rückwärtsmodell

Durch Kombination von Vorwärts- und Rückwärtsmodell ist es möglich die Konsequenzen der eigenen Aktionen mit den beobachteten Konsequenzen abzugleichen. Es besteht damit die Möglichkeit abzuschätzen, ob die beobachteten Konsequenzen selbst verursacht oder eventuell durch Andere verursacht wurden.

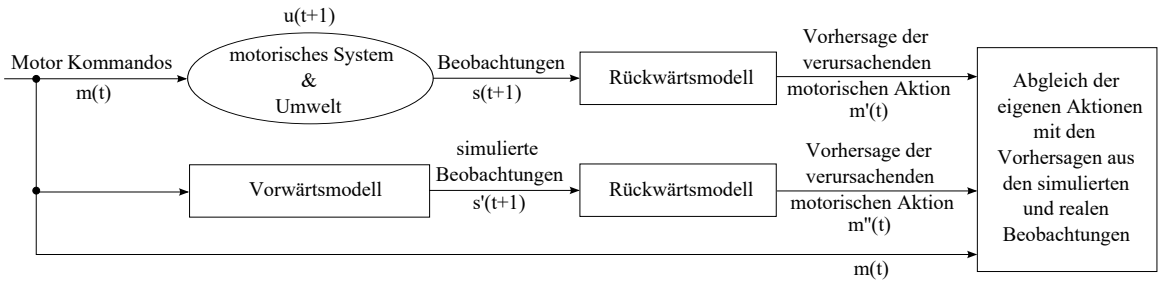


Abbildung 2.4: *Beispielschema einer Kombination von Vorwärts- und Rückwärtsmodell bei einem Roboter*

2.3 Neuronale Netze

Neuronale Netze basieren in ihrer Konzeption auf den Zellstrukturen von Nervenzellen und deren Vernetzung, wie sie im Gehirn des Menschen und vieler anderer Lebewesen dieses Planeten zu finden sind. Die Zellstruktur einer Nervenzelle ist in Abbildung 2.5 dargestellt.

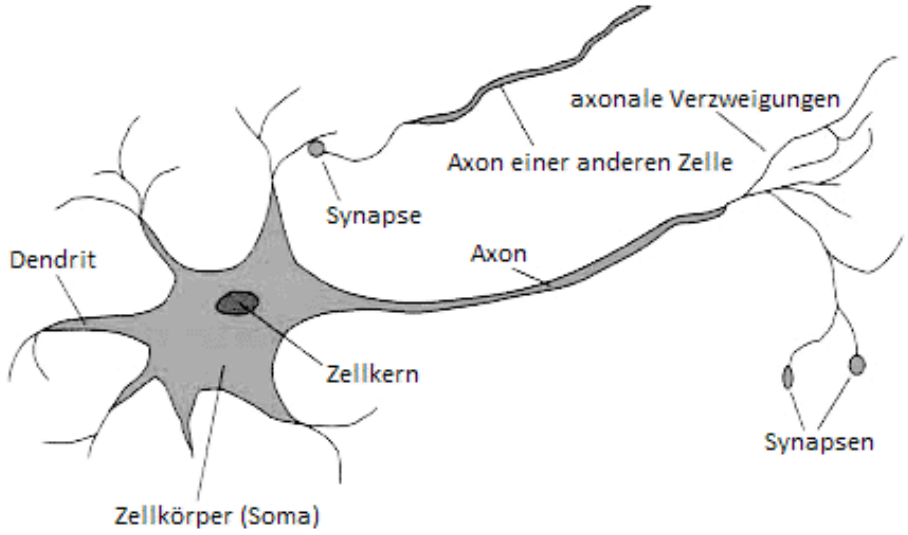


Abbildung 2.5: *Aufbau einer Nervenzelle. Die Axone sind verkürzt dargestellt. Sie sind ungefähr 100 mal größer als der Zellkörper, können aber auch mehrere Meter lang sein. aus [RN95, S.565]*

Bei einer Nervenzelle werden die funktionalen Bestandteile Zellkörper (Soma), Dentrit, Axon und Synapse unterschieden:

- Im Zellkörper finden biochemische Synthesen statt und er umgibt den Zellkern.
- Die Dendriten sind kurze, stark verzweigte Fortsätze (Nervenfasern) des Zellkörpers.
- Das Axon ist die eigentliche, leitende Nervenfaser. Ein Axon ist wenig verzweigt und hat eine Länge von ungefähr dem einhundert-fachen des Zellkörperdurchmessers. Es kann aber auch mehrere Meter lang sein.
- Die Synapsen sind die Verbindungsstellen zwischen den Nervenfasern. Sie verbinden das Axon einer Nervenzelle mit einem Dendriten anderer Nervenzellen.

Die Weiterleitung der Erregung einer Nervenzelle ist gerichtet. Die Erregung einer Zelle kann bei der Weiterleitung zu anderen verbundenen Zellen verstärkt oder gehemmt werden. Der Erregungsfluss geht von den Dendriten einer Nervenzelle, über ihren Zellkörper und das Axon, zu den über Synapsen verbundenen Dendriten anderer Nervenzellen. Die Reizweiterleitung von einem Axon einer Nervenzelle zu einem Dendriten einer anderen Nervenzelle geschieht in den Synapsen auf biochemischem Weg. Unter dem Gesichtspunkt der Nervenzelle als informationsverarbeitendes Element weisen Nervenzellen nicht-lineares Verhalten auf.

Das Basiselement eines künstlichen neuronalen Netzes, das Neuron, ist eine vereinfachte mathematische Approximation an die Funktionsweise der Nervenzellen. Den schematischen Aufbau eines Neurons zeigt Abbildung 2.6.

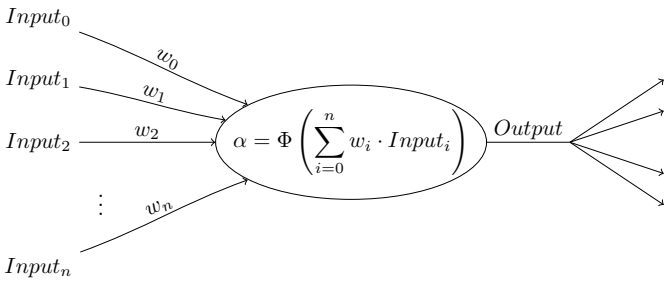
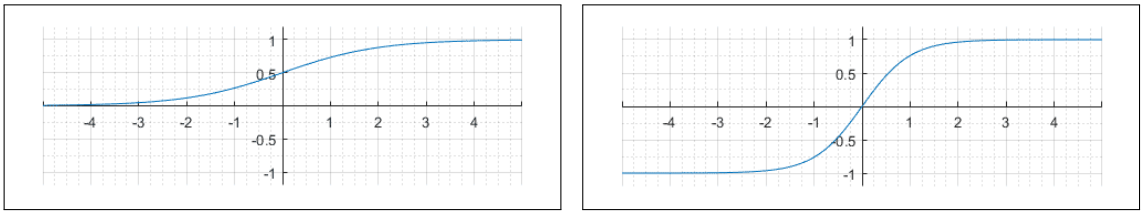


Abbildung 2.6: *Schematische Darstellung eines Neurons.*

Die gewichteten Eingangssignale $w_i \cdot Input_i$ werden aufsummiert und einer meistens nicht-linearen Transferfunktion Φ als Argument übergeben. Das Ergebnis der Transfer- bzw. Aktivierungsfunktion wird auch als Aktivierung α bezeichnet.

Häufig werden sigmoide Transferfunktionen verwendet, wie die logistische Funktion oder der Tangens-Hyperbolicus. Beide sind nicht-linear und haben den Vorteil, dass sie einfach abgeleitet und umgekehrt werden können.



(a) *Logistische Funktion* **(b)** *Tangens-Hyperbolicus*

Abbildung 2.7: *Häufig genutzte sigmoide Transferfunktionen.*

Es werden in einigen Fällen auch andere Transferfunktionen verwendet. Einige davon sind Annäherungen an die sigmoiden Funktionen, die auf Kosten eines entsprechenden Fehlers eine schnellere Berechnung ermöglichen. Andere sind etwa die Gauß-Glocke, Periodenabschnitte der Sinus- oder Kosinus-funktion, die lineare Funktion oder die Sprungfunktion. Je nach genutzter Transferfunktion sind deren Definitions- und Wertebereich zu beachten. Das heißt Eingangsdaten und Ausgangsdaten müssen entsprechend der verwendeten Transferfunktion auf deren Definitions- und Wertebereich hin normalisiert werden. Meist haben Transferfunktionen einen Wertebereich im Intervall [0,1] oder im Intervall [-1,1].

2.3.1 Backpropagation

Ein Standard-Lernverfahren für Feedforward Netze ist das Backpropagation-Verfahren. Es folgt eine Beschreibung des Backpropagation-Verfahrens, die sich an [RB93] anlehnt. Beim Back-Propagation-Verfahren wird wiederholt die Kettenregel angewandt, um die Eingangsgewichte der Neuronen der jeweiligen Schicht in Abhängigkeit der Fehlerfunktion E zu aktualisieren. Die Kettenregel ist nach [RB93] definiert mit:

$$\frac{\delta E}{\delta w_{ij}} = \frac{\delta E}{\delta s_i} \cdot \frac{\delta s_i}{\delta net_i} \cdot \frac{\delta net_i}{\delta w_{ij}} \tag{2.30}$$

Dabei ist w_{ij} das Eingangsgewicht der Verbindung zwischen Neuron j und i von Neuron i, s_i bezeichnet den Ausgang des Neurons i und net_i die gewichtete Summe des Eingangs von Neuron i. Nachdem die Ableitungen aller Gewichte berechnet worden ist, wird die Fehlerfunktion mit dem Gradientenabstiegsverfahren minimiert. Definition des Gradientenabstiegs [RB93]:

$$w_{ij}(t+1) = w_{ij}(t) - \epsilon \cdot \frac{\delta E}{\delta w_{ij}}(t) \tag{2.31}$$

Eine schlechte Wahl der Lernrate ϵ kann zu langsamem lernen oder Oszillationen führen. Um diese Probleme zu mildern, wurde ein zusätzlicher Term eingeführt, der über den Parameter μ den Einfluss des vorhergehenden Berechnungsschritts auf den aktuellen mit einbezieht und reguliert. Definition des Gradientenabstiegs mit μ-Term [RB93]:

$$\Delta w_{ij}(t) = -\epsilon \cdot \frac{\delta E}{\delta w_{ij}}(t) + \mu \Delta w_{ij}(t-1) \tag{2.32}$$

In der Praxis hat sich gezeigt, dass der μ-Term genauso vom gestellten Problem abhängig ist, wie die Lernrate ϵ. Weshalb andere Möglichkeiten gesucht wurden, die Abhängigkeit der Hyper-Parameter von der Problemstellung zu entfernen. Das Resilient-Propagation-Verfahren von [RB93] hat den Anspruch dies zu erreichen.

2.3.2 Resilient-Propagation

Um die Abhängigkeit der Hyper-Parameter von der Problemstellung beim Training von neuronalen Netzen zu vermeiden, wurde das Resilient-Propagation-Verfahren von [RB93] vorgeschlagen. Es folgt eine Beschreibung des Resilient-Propagation-Verfahrens, die sich an [RB93] anlehnt.

Das Verfahren führt für jedes Eingangsgewicht w_{ij} einen zusätzlichen Aktualisierungsparameter Δ_{ij} ein, der die Größe der Aktualisierung von w_{ij} bestimmt. Der Aktualisierungsparameter verändert seine Größe auf Basis der zum Neurons i lokalen Umgebung in der Fehlerfunktion.

Die Lernregel für Δ_{ij} ist [RB93]:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ \cdot \Delta_{ij}^{(t-1)} & , \text{ falls } \frac{\delta E}{\delta w_{ij}}^{(t-1)} \cdot \frac{\delta E}{\delta w_{ij}}^{(t)} > 0 \\ \eta^- \cdot \Delta_{ij}^{(t-1)} & , \text{ falls } \frac{\delta E}{\delta w_{ij}}^{(t-1)} \cdot \frac{\delta E}{\delta w_{ij}}^{(t)} < 0 \\ \Delta_{ij}^{(t-1)} & , \text{ sonst} \end{cases} \tag{2.33}$$

mit $0 < \eta^- < 1 < \eta^+$

Mit dem Aktualisierungsparameter Δ_{ij} wird dann die Änderung des Eingangsgewichts Δw_{ij} berechnet:

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} & , \text{ falls } \frac{\delta E}{\delta w_{ij}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)} & , \text{ falls } \frac{\delta E}{\delta w_{ij}}^{(t)} < 0 \\ 0 & , \text{ sonst} \end{cases} \tag{2.34}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \tag{2.35}$$

Damit ergibt sich das folgende Verhalten:

- Ändert sich das Vorzeichen des Gradienten nicht und ist der Gradient positiv, wird das Gewicht w_{ij} verringert, um dem Anstieg des Gradienten der Fehlerfunktion langsamer zu folgen.
- Ändert sich das Vorzeichen des Gradienten nicht und ist der Gradient negativ, wird das Gewicht w_{ij} vergrößert, um dem abfallenden Gradienten der Fehlerfunktion schneller zu folgen.
- Ändert sich das Vorzeichen des Gradienten und ist der Gradient positiv, wird das Vorzeichen des um Δw_{ij} verringerten Gewichts w_{ij} umgekehrt, da ein Minimum übersprungen wurde. Damit wird erreicht, dass der zu weite Schritt nur teilweise zurückgenommen wird und das weitere Folgen in Richtung des übersprungenen Minimums ermöglicht wird.
- Ändert sich das Vorzeichen des Gradienten und ist der Gradient negativ, wird das Vorzeichen des um Δw_{ij} vergrößerten Gewichts w_{ij} umgekehrt, da ebenfalls ein Minimum übersprungen wurde. Damit wird auch dann erreicht, dass der zu weite Schritt teilweise zurückgenommen wird und das weitere Folgen in Richtung des übersprungenen Minimums ermöglicht wird.

Falls sich das Vorzeichen des Gradienten geändert hat, muss verhindert werden, dass im nächsten Schritt die Korrektur der Bewegungsrichtung in der Fehlerfunktion wieder zurückgenommen wird. Dies kann erreicht werden, indem im nächsten Schritt bei

der Berechnung des Aktualisierungsparameters Δ_{ij} der Gradient $\frac{\delta E}{\delta w_{ij}}^{(t-1)}$ auf null gesetzt wird: $\frac{\delta E}{\delta w_{ij}}^{(t-1)} := 0$.

Die Berechnung der Aktualisierungsparameter und der Gewichte erfolgt immer dann, wenn alle Daten einer Epoche dem neuronalen Netz präsentiert worden sind.

2.3.3 Feedforward Netze (Feedforward Neuronal Networks)

Die Neuronen werden in Abhängigkeit ihrer Funktion in die Schichten eingeordnet. Input-Neuronen bilden die Eingangsschicht (Input-Layer) des Netzes, Output-Neuronen bilden die Ausgangsschicht (Output-Layer). Schichten mit Neuronen zwischen Input-Schicht und Output-Schicht, deren Verhalten von außen nicht direkt beobachtbar ist, werden versteckte Schicht (Hidden-Layer) genannt. Neuronen einer solchen Schicht heißen Hidden-Neuronen [RN95, S.571].

Ein Feedforward Netz (Feedforward Neuronal Network) ist ein neuronales Netzwerk, beim dem die Transitionen bzw. Verbindungen von den Neuronen einer Schicht (Layer) immer zu den Neuronen der nächsten Schichten ausgerichtet sind. Es gibt keine Transitionen zu Neuronen vorheriger Schichten oder Querverbindungen zu Neuronen der gleichen Schicht.

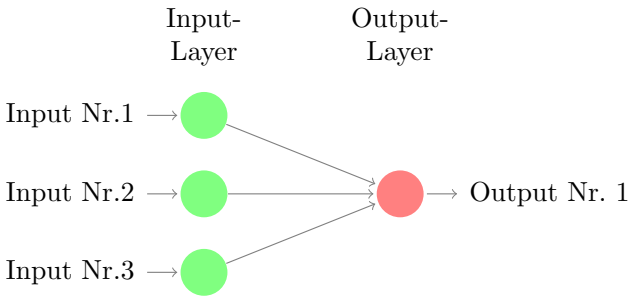


Abbildung 2.8: *Single Layer-Perzeptron mit 3 Input-Neuronen und einem Output-Neuron.*

Feedforward Netze, bei denen die Verbindungen zwischen den Neuronen nur von einer Schicht zur nächst folgenden Schicht bestehen, heißen Perzeptron. Die einfachste Version eines Perzeptrons besteht aus nur einem einzigen Neuron. Perzeptrons ohne einen Hidden-Layer werden als Single-Layer-Perzeptron (SLP) und solche mit einem oder mehr Hidden-Layern als Multi-Layer-Perzeptron (MLP) bezeichnet. Abbildung 2.8 zeigt ein Beispiel eines Single-Layer-Perzeptron und Abbildung 2.9 ein Beispiel für ein Multi-Layer-Perzeptron.

Viele mit neuronalen Netzen lösbare Problemstellungen lassen sich mit einem Single-Layer-Perzeptron lösen, solange keine relevanten Unstetigkeiten in einem abzubildenden funktionalen Zusammenhang zwischen Eingangsdaten und abzubildenden Ausgangsdaten bestehen [RN95, S.571]. Sind Unstetigkeiten vorhanden, können diese in der Regel von einem einschichtigen MLP abgebildet werden [HSW89].

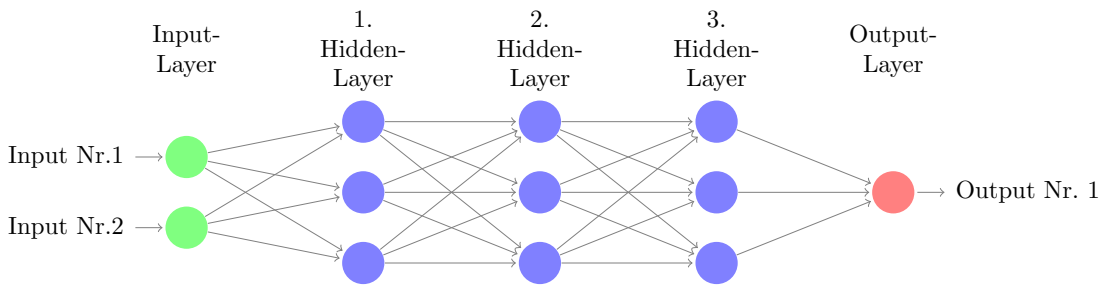


Abbildung 2.9: *Dreischichtiges Multi-Layer-Perzeptron mit 2 Input-Neuronen, 3 Hidden-Neuronen je Hidden-Layer und einem Output-Neuron.*

2.3.4 Rekurrente Netze (Recurrent Neuronal Networks)

Rekurrente neuronale Netze werden ähnlich den Feedforward Netzen in Schichten eingeteilt. Sie besitzen ebenso, wie Feedforward Netze, einen Input-Layer und einen Output-Layer. Der wesentliche Unterschied zu Feedforward-Netzen ist, dass bei rekurrenten Netzen auch Verbindungen zwischen Neuronen der gleichen Schicht, Verbindungsschleifen zu sich selbst und auch Verbindungen zu Neuronen in beliebigen anderen Schichten vorhanden sind.

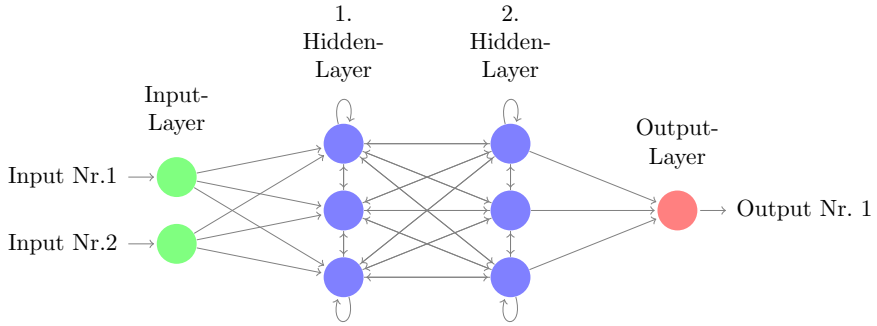


Abbildung 2.10: *Rekurrentes neuronales Netz mit 2 Input-Neuronen, 3 Hidden-Neuronen je Hidden-Layer und einem Output-Neuron.*

Mit rekurrenten Netze können Zustände und Speicherfunktionalitäten abgebildet werden. Sie sind aufgrund ihrer größeren Komplexität auch schwerer zu trainieren. Zum Beispiel beeinflusst eine Verbindung des Ausgangs eines Neurons zum Eingang des gleichen Neurons das Neuron im nächsten Zeitschritt der Berechnung mit der Aktivierung dieses Neurons vom aktuellen Zeitschritt. Für rekurrente Netze geeignete Lernverfahren müssen diese zeitliche Abhängigkeit berücksichtigen.

3 Nao - Ein humanoider Roboter

In diesem Kapitel wird die verwendete Roboterplattform, der Nao, im Abschnitt 3.1 vorgestellt und im Abschnitt 3.2 werden einige Maßnahmen beschrieben, die notwendig sind, um aus den vorhandenen Sensor- und Motordaten die auditorische Konsequenz lernen zu können.

3.1 Ausstattung des Nao

Abbildung 3.1: *Werbeplakat zum Nao vom Hersteller Aldebaran Robotics*

Der humanoide Roboter Nao ist ca. 57 cm hoch. Er wird von der französischen Firma Aldebaran Robotics hergestellt. Den Nao gibt es inzwischen in der fünften Generation und in verschieden ausgestatteten Modellen.

3.1.1 Hardware

Die Naos der fünften Generation haben 25 Freiheitsgrade[1]. Die ungerade Anzahl an Freiheitsgraden ist beim Nao dadurch bedingt, dass statt zwei Hüftgelenken nur ein kombiniertes Hüftgelenk vorhanden ist. In welchem Winkel ein Gelenk steht, wird mit Hall-Effekt-Sensoren an den Gelenken ausgemessen. Jeder Fuß besitzt einen einfachen

[1] Abgekürzt DOF aus dem Englischen: degrees of freedom.

Kollisionssensor vorne und vier Kraftsensoren an der Unterseite. Des Weiteren sind ein drei-Achsen-Gyroskop und ein drei-Achsen-Beschleunigungssensor im Torso, zwei Infrarotempfänger und -sender in den Augen, zwei Ultraschallempfänger und -sender im Torso vorne, vier Mikrophone, zwei Lautsprecher und zwei Kameras im Kopf verbaut. Die zwei Kameras haben eine recht ungewöhnliche Position im Kopf, da eine Kamera im Mund und eine Kamera in der Stirn eingebaut ist. Die Kameras können eine Auflösung bis 1280x960 Pixel bei 30 Hz liefern.

Der Nao ist mit zwei Recheneinheiten ausgestattet, die miteinander kommunizieren. Im Torso ist eine ARM basierte Steuerplatine für das Auslesen der Sensoren und das Ansteuern der Aktuatoren zuständig, diese ist aber für andere Aufgaben nicht vorgesehen. Im Kopf befindet sich neben den Kameras und den Mikrophonen ein Kleinstrechner mit einer Intel Atom Z530 CPU (1.60 GHz Taktfrequenz), 10 GB Arbeitsspeicher, 1 GB Ethernet LAN, 802.11abgn WLAN, 2 GB interner Flash-Speicher, eine integrierte 8 GB MicroSDHC Speicherkarte und USB 2.0.

3.1.2 Software

Als Betriebssystem kommt auf den Naos ein Gentoo-basiertes, echtzeitfähiges Linux zum Einsatz. Auf diesem läuft eine modulare Middleware des Herstellers Aldebaran Robotics mit dem Namen NaoQi. Ab Werk ist diese Middleware mit vielen Modulen für Grundfunktionalitäten eines humanoiden Roboters ausgestattet.

Für das EARS Projekt wurde von Aldebaran Robotics die Erweiterung Modularity der NaoQi-Middleware bereitgestellt. Das Modularity-Framework erlaubt die Implementierung modular aufgebauter Filterketten. Diese sind eigenständige dynamische Bibliotheken, die auch dynamisch geladen und entladen werden können. Zudem erlaubt das Modularity-Framework Prozesse mit unterschiedlichen Ausführungszyklen zu definieren und zwischen ihnen, mit Nutzung von Shared-Pointern, Daten auszutauschen.

3.2 Vorbereitung der Experimente

Im folgenden Abschnitt werden notwendige Vorbereitungen für die Experimente behandelt. Zuerst wird die Synchronisierung von Motor-, Sensor und Audiodaten beschrieben. Am Anschluss daran folgen einige Überlegungen zur Auswahl von Softwarebibliotheken, die eine effizient Implementierung der Berechnung von schnellen Fourier-Transformationen realisieren.

3.2.1 Signalsynchronisierung

Bevor eine Modellierung akustischer Merkmale als Konsequenz einer motorischen Aktion vorgenommen werden kann, muss eine zeitliche Zuordnung des Auftretens der motorischen Aktion mit dem zeitlichen Auftreten der akustischen Konsequenz vorgenommen werden. Der Grund hierfür sind zueinander verschiedene Ausführungszyklen

der Prozesse zur Audioaufnahme und der Prozesse zur Sensor- und Motordatenaufnahme. Die Aufnahme zueinander synchroner Daten aus verschiedenen sensorischen Quellen ist immer ein plattformspezifisches Problem. Beim Nao können Sensor- und Motordaten in etwa alle 10 ms aufgenommen werden. Audiodaten dagegen können nur alle 80 ms bis 170 ms aufgenommen werden. Obwohl auf dem Nao ein Echtzeitbetriebssystem zu Einsatz kommt, sind exakte Ausführungszeiten nicht garantiert. Das heißt die realen Zeiten zwischen zwei Aufrufen eines Aufnahmeprozesses schwanken um einige wenige Millisekunden um die exakte Ausführungszeit herum.

Eine Besonderheit beim Nao ist, dass während der Aufnahme der Sensor- und Motordaten kein Speicher alloziert werden darf, da ansonsten der Motorkontrollprozess ins Stocken gerät und der Roboter seine Motoren nicht mehr korrekt kontrollieren und ansteuern kann. Aus diesem Grund wird hier zur Aufnahme ein einmalig allozierter Zwischenspeicher in doppelter Ausführung verwendet, der eine feste Größe besitzt und bei jedem vierten Aufnahmezyklus zur Hälfte überschrieben wird. Die Größe des Zwischenspeichers ist so gewählt, dass die Sensor- und Motordaten aus 4 Zyklen in einer Hälfte des Speichers Platz finden. Die zwei Hälften des Zwischenspeichers sind jeweils mit einem Zugriffsindex indiziert, wobei ein Index auf den Teil zeigt, der gelesen werden darf und der andere auf den Teil, der überschrieben werden kann. Nach dem Überschreiben von je 4 Sensor- und Motordatenblöcken werden Schreib- und Leseindex vertauscht. Ein anderer Prozess der alle 40 ms ausgeführt wird, liest dann die Sensor- und Motordaten ein, die im durch den Leseindex indizierten Teil des Zwischenspeichers liegen und hängt diese in einem weiteren doppelten Zwischenspeicher an. Der Unterschied zum ersten Zwischenspeicher ist, dass dieser Zwischenspeicher nicht mit einer festen Größe alloziert wird, sondern anwächst. Der Audioaufnahmeprozess wird ungefähr alle 80 ms ausgeführt. Auch hier wird ein doppelter anwachsender Zwischenspeicher verwendet. Die ungefähr alle 80 ms bis 90 ms aufgenommenen Audiodaten werden an die Daten des schreib-indizierten Teils einfach angehängt.

Die eigentliche Synchronisation der Sensor-, Motor- und Audiodaten erfolgt in einem weiteren Prozess der die Daten aus den lese-indizierten Teilen der zwei dynamischen Zwischenspeicher ausliest und unter Verwendung der Systemzeitstempel der Sensor-, Motor- und Audiodaten diese zueinander zeitlich zuordnet. Dieser Synchronisationsprozess wird im Durchschnitt alle 200 ms ausgeführt. Mit seiner Hilfe werden zeitlich direkt aufeinander folgende Datenpakete zusammengestellt, die die mit Zeitstempeln synchronisierten Daten eines 200 ms Zeitabschnitts enthalten. Anschließend können die synchronisierten Daten durch andere Prozesse und Module, wie Modularity-Filterketten, weiterverarbeitet werden. Zur Berechnung der MFCC-Koeffizienten werden dazu einander überlappende Zeitfenster genutzt, für die die MFCC-Koeffizienten berechnet werden. Nach der Berechnung der Koeffizienten werden jedem MFCC-Merkmalsvektor die Sensor- und Motordaten zum Beginn des Zeitfensters und die Sensor- und Motordaten zum Ende des Zeitfensters zugeordnet.

3.2.2 Benchmarks von Fast-Fourier Implementierungen

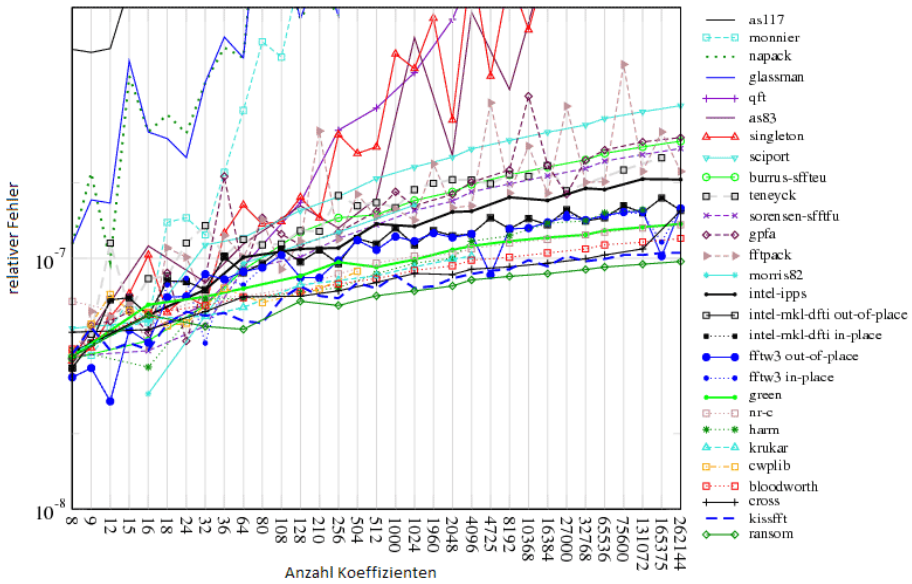
Da bei der Implementierung der schnellen Fouriertransformation (FFT) oft numerische Tricks zu Einsatz kommen und auch weil die Genauigkeit und Ausführungsgeschwindigkeit der Berechnung eines Spektrums mit einer FFT stark vom verwendeten Rechnersystem abhängig sind, ist es sinnvoll einen Vergleich der Genauigkeit und Geschwindigkeit verschiedener FFT Implementierungen vorzunehmen.

Umfangreiche und detaillierte Benchmarks von FFT Implementierungen wurden von Steven Johnson und Matteo Frigo in [JFb] für die Geschwindigkeit und in [JFd] für die Präzision der Berechnung von Fourier-Transformationen auf verschiedenen Plattformen durchgeführt. Aus den zahlreichen Benchmarks sind zwei auf einer Plattform durchgeführt worden, die der Hardware des Nao und auch der Hardware meines eigenen Rechners ähnlich genug ist, um anhand dieser Benchmarks eine Auswahl von hinreichend guten FFT Implementierung für die Experimente online auf dem Nao und offline auf meinem Rechner zu treffen. Die zwei Benchmarks unterscheiden sich darin, mit welchem Compiler der Quellcode der Testprogramme optimiert wurde. Die Testprogramme wurden mit einem Compiler von Intel und dem freien GNU-C-Compiler erstellt. Beide Compiler unterscheiden sich in der Art der durchgeführten Optimierungen und der resultieren Ausführungsgeschwindigkeit und Genauigkeit auf manchen Systemen erheblich. In den Abbildungen Abb. 3.2 und Abb. 3.3 sind exemplarisch die Ergebnisse der Testläufe der mit dem GNU-C-Compiler erstellten Testprogramme aufgeführt.

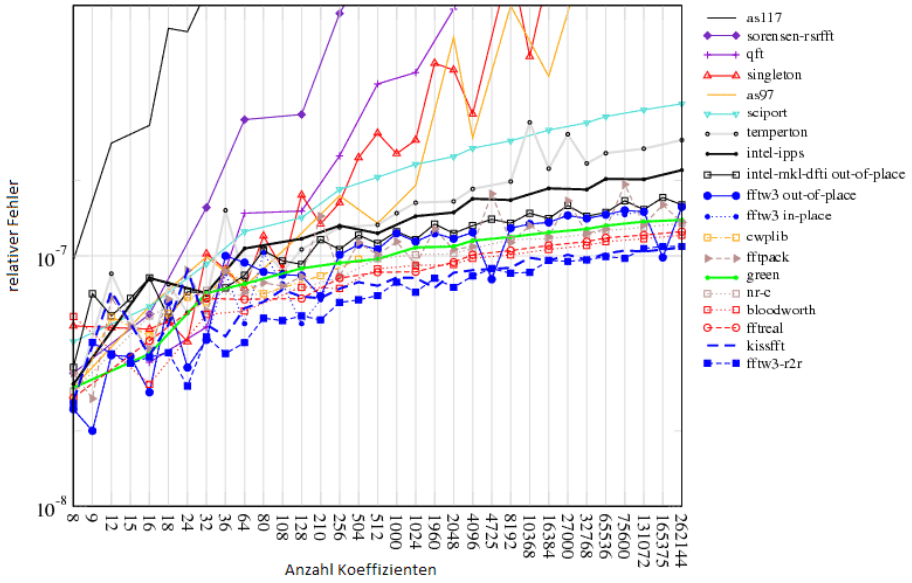
Auf dem Nao hersteller-seitig mitgeliefert, wird die FFT-Bibliothek fftw in der Version 3. Aus den Vergleichen in Abbildung Abb. 3.2a und Abb. 3.2b kann man entnehmen, dass diese auf dem Nao verfügbare FFT-Implementierung mit zu den besten Implementierung bei der Berechnung von komplexen und reellen Spektren mit einfacher Genauigkeit gehört. Der relative Fehler der zwölf besten Implementierungen befindet sich in dem Bereich der siebenten Nachkommastelle. Ähnliches gilt für Berechnung mit doppelter Genauigkeit. Auch hier ist die FFT-Implementierung fftw unter den schnellsten Zehn, wie in Abbildungen Abb. 3.3a und Abb. 3.3b zu sehen ist. Der relative Fehler der elf besten Implementierungen bei Berechnungen mit doppelter Genauigkeit bewegt sich nahe der 16. Nachkommastelle. Die Berechnung mit doppelter Genauigkeit ist damit um einen Faktor von ungefähr 10^9 genauer als die Berechnung mit einfacher Genauigkeit. Die sehr kleine und schlanke FFT-Implementierung von Takuya Ooura [Oou06] ist eine in Open-Source verfügbare Alternative. Es ist aber nur eine Version für die Berechnung mit doppelter Genauigkeit verfügbar.

In den Abbildungen 3.4 und 3.5 sind exemplarisch die Messergebnisse aus [JFb] zum Vergleich der Ausführungsgeschwindigkeit verschiedener FFT-Implementierungen abgebildet. Die Messkurven in diesen Abbildungen zeigen die Messergebnisse für die Anzahl der Koeffizienten der Länge 2^n, wobei die in den Abbildungen gezeigten Werte als ein 'Best-Case'- Szenario zu verstehen sind.

Bei Betrachtung der Grafiken fällt auf, dass die Berechnung der Spektren mit einfacher Genauigkeit bei einigen Implementierungen fast doppelt so schnell erfolgt, im Vergleich zur Berechnung mit doppelter Genauigkeit. Auch ist interessant, dass sich bei einfacher Genauigkeit die Maxima der schnellsten Implementierungen bei einer



(a) *Komplexes Spektrum, einfache Genauigkeit, aus [JFc]*

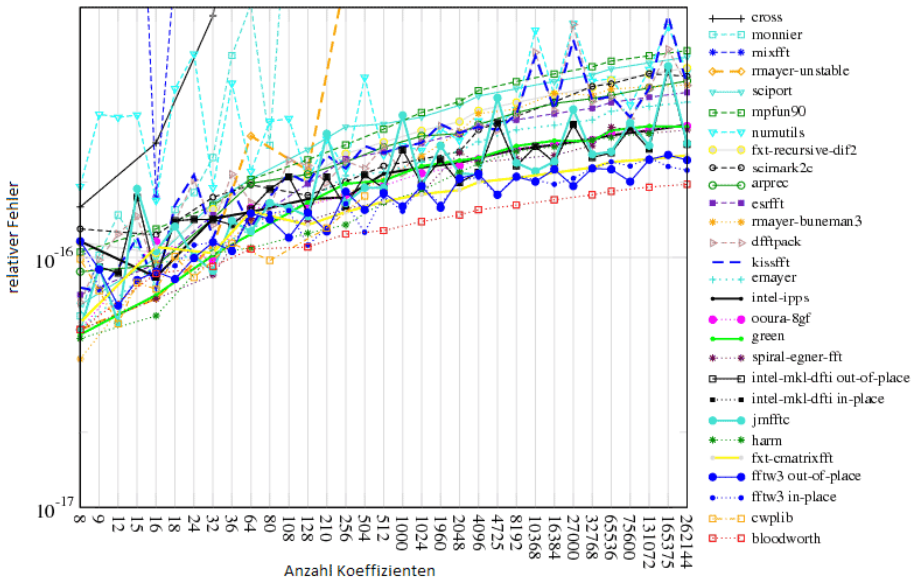


(b) *Reelles Spektrum, einfache Genauigkeit, aus [JFc]*

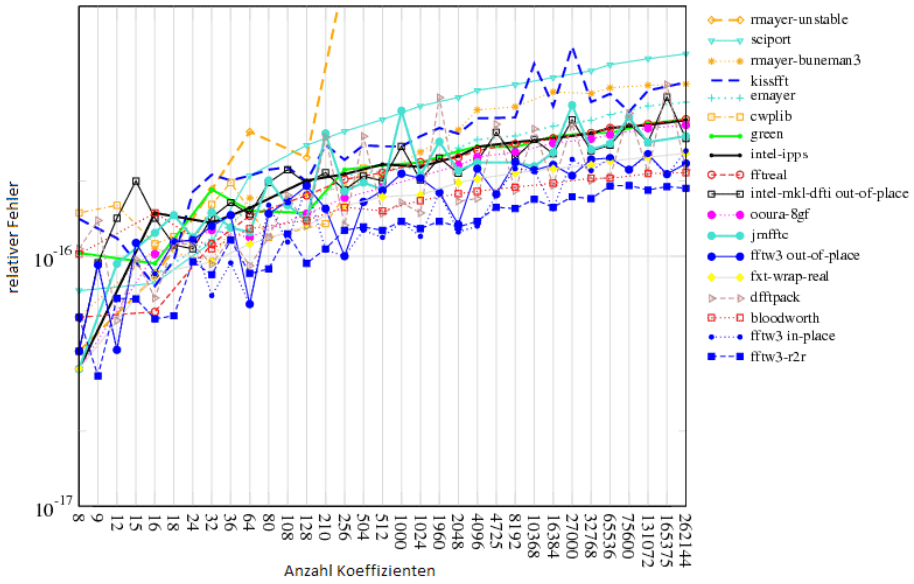
Abbildung 3.2: *An den zwei Detailansichten kann man sehen, dass die zwölf besten getesteten Implementierungen eine Genauigkeit im Bereich der siebenten und achten Nachkommastelle erreichen, wenn der Datentyp float zur Berechnung eines komplexen sowie eines reellen Spektrums verwendet werden. Die Grafiken zeigen auch, dass die fftw-Implementierung zu den Genauesten gehört.*

Koeffizientenanzahl von 512, 1024 oder 2048 befinden und bei doppelter Genauigkeit bei Koeffizientenanzahlen von 64 bis 2048.

Damit würde die Berechnung eines Spektrums eines 1D-Signals von 40 ms zeitlicher



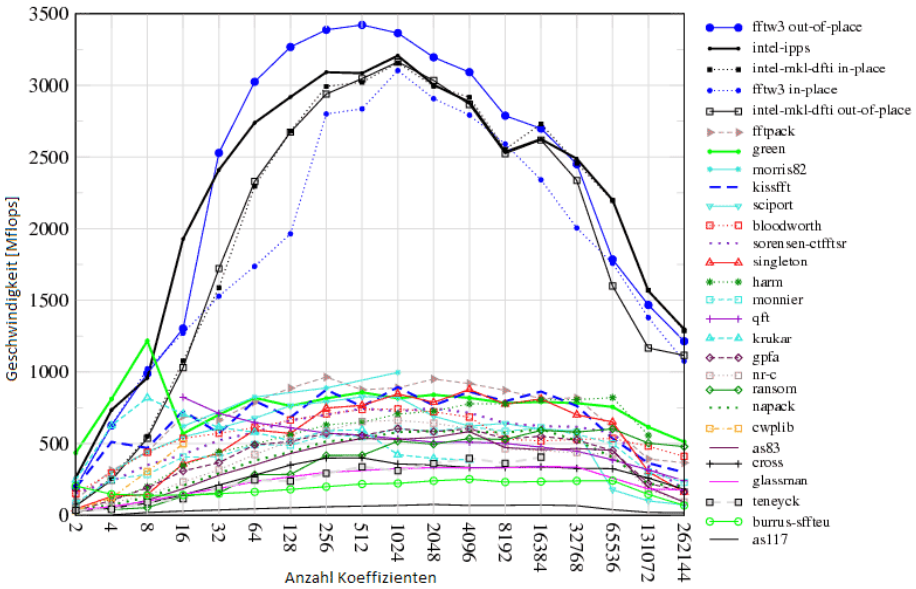
(a) *Komplexes Spektrum, doppelte Genauigkeit, aus [JFc]*



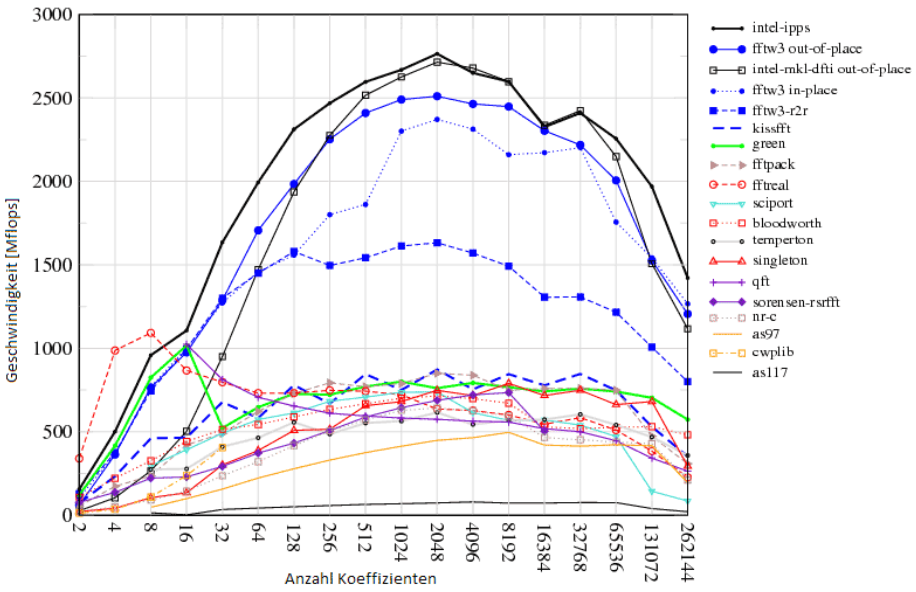
(b) *Reelles Spektrum, doppelte Genauigkeit, aus [JFc]*

Abbildung 3.3: *An den zwei Detailansichten kann man sehen, dass die zwölf besten getesteten Implementierungen eine Genauigkeit im Bereich der sechzehnten und siebzehnten Nachkommastelle erreichen, wenn der Datentyp double zur Berechnung eines komplexen sowie eines reellen Spektrums verwendet werden. Die Grafiken zeigen auch, dass die fftw-Implementierung auch dann zu den Genauesten gehört.*

Länge bei einer Abtastfrequenz von 48 kHz innerhalb des Bereiches der nahezu optimalen Geschwindigkeit guter FFT-Implementierungen liegen,. Bei einfacher Genauigkeit sind die Implementierungen von Intel und fftw sichtbar besser als die anderen



(a) *Komplexes Spektrum, einfache Genauigkeit, aus [JFa]*

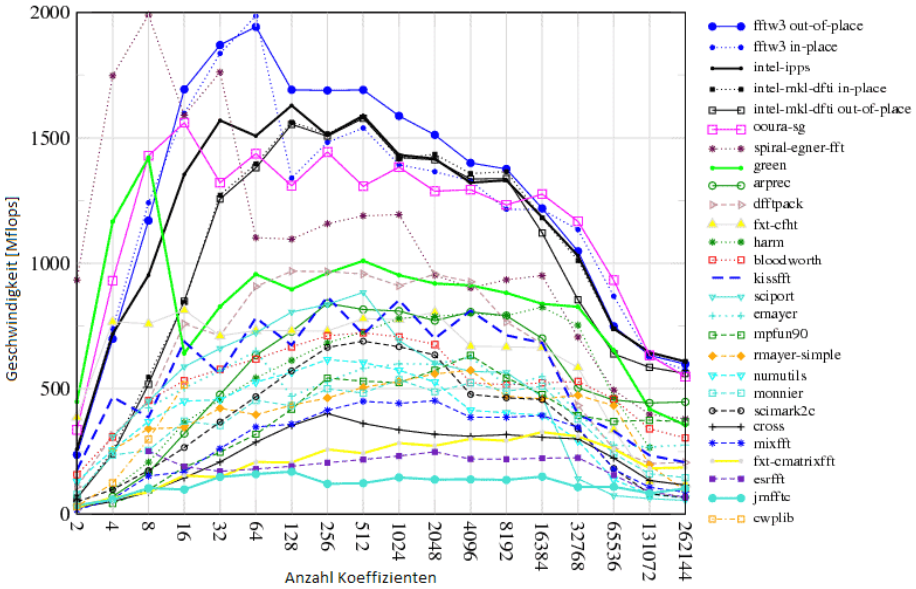


(b) *Reelles Spektrum, einfache Genauigkeit, aus [JFa]*

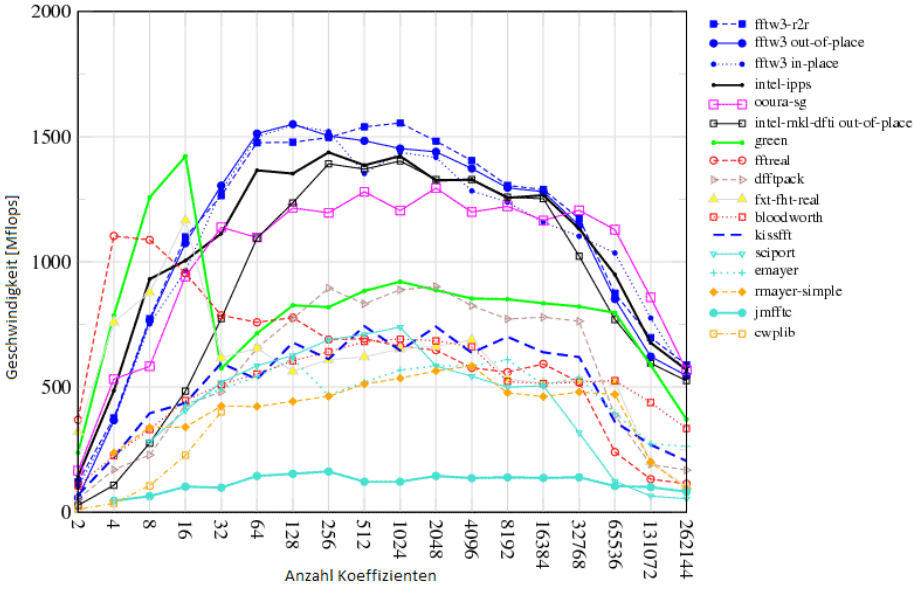
Abbildung 3.4: *Beide Grafiken zeigen deutlich die Abhängigkeit der Ausführungsgeschwindigkeit von der Anzahl der zu berechnenden Koeffizienten, bei der Berechnung reeller oder komplexer Spektren mit einfacher Genauigkeit. Es fällt auf, dass diese Abhängigkeit bei den vier schnellsten Implementierungen besonders markant ist.*

Getesteten. Bei doppelter Genauigkeit ist auch die FFT-Implementierung von Ooura mit unten den schnellsten 5 Implementierungen.

Die FFT-Implementierung fftw ist auf dem Nao schon vorhanden und eine der sehr guten Implementierungen bei Berechnungen mit einfacher und doppelter Genauigkeit



(a) *Komplexes Spektrum, doppelte Genauigkeit, aus [JFa]*



(b) *Reelles Spektrum, doppelte Genauigkeit, aus [JFa]*

Abbildung 3.5: *Beide Grafiken zeigen deutlich die Abhängigkeit der Ausführungsgeschwindigkeit von der Anzahl der zu berechnenden Koeffizienten, bei der Berechnung reeller oder komplexer Spektren mit doppelter Genauigkeit. Im Vergleich zu 3.4 wird deutlich, dass die Ausführungsgeschwindigkeit fast auf die Hälfte singt, wenn mit doppelter Genauigkeit gerechnet wird.*

bezüglich Geschwindigkeit und Präzision. Somit verlangt die Einbindung und Nutzung auf dem Nao den geringsten Aufwand, sie stellt deshalb auf dem Nao die erste Wahl dar.

Für eine Realisierung von Offline Lern- und Klassifizierungsverfahren bietet sich auch die Implementierung von Ooura an, da lediglich eine einzelne Datei in ein Programm mit eingebunden werden muss, anstatt gegen eine ganze Bibliothek linken zu müssen, die zudem vorher für das Zielsystem kompiliert werden muss. Der Aufwand sie zu verwenden, ist also sehr klein.

Es soll die Eignung neuronaler Modelle zur Abbildung der Eigengeräusche anhand der Eigenbewegung untersucht werden, deshalb erscheint es sinnvoll zunächst die höhere numerische Präzision der doppelt genauen FFT-Implementierungen zu nutzen, trotz der geringeren Ausführungsgeschwindigkeit. Somit fällt die Wahl der zu verwendenden FFT-Implementierung auf fftw und alternativ Ooura.

4 Vorhersage der Eigengeräusche aus motorischer Aktion

Für das Lernen der Modelle mit einem Multi-Layer-Perzeptron (MLP) kommt die C-Bibliothek FANN zum Einsatz. Diese Bibliothek bietet als Methoden der Fehlerpropagation inkrementelle Backpropagation, QuickProp und Resilient Propagation (RProp) an. QuickProp und RProp sind, anders als die inkrementelle Backpropagation, nur im Batch-Modus anwendbar. Das heißt sie sind für ein Online Lernen nicht geeignet. Für sehr viele Aufgabenstellungen ist eine Hidden-Layer-Schicht und damit ein einschichtiges Multi-Layer-Perzeptron (MLP) ausreichend. Ob dabei das Problem des Underfittings oder das Problem des Overfittings auftritt, hängt dann häufig von der Anzahl der Neuronen im Hidden-Layer, der Lernrate, der Länge der Lernphase bzw. der Anzahl der Iterationen (wie z.B. bei dem Batch-Lernverfahren Resilient Propagation) und auch vom eingesetzten Lernverfahren und den dabei genutzten Regularisierungsmethoden ab.

Für die folgenden Untersuchungen dienen als Grundlage Audioaufnahmen mit einer Sampling-Rate von 48 kHz und Daten über die aktuell angesteuerte Position sowie die aktuell gemessene Position der Gelenke des Roboters mit einer Rate von 100 Hz. Aus den Audiodaten werden unabhängig von der verwendeten Fenstergröße Spektren bis zu einer Frequenz von maximal 24 kHz berechnet, wovon nur die Koeffizienten bis zu einer Frequenz von ungefähr 16 kHz genutzt werden. Daraus werden dann 32 MFCC-Koeffizienten berechnet, von denen maximal 26 genutzt werden.

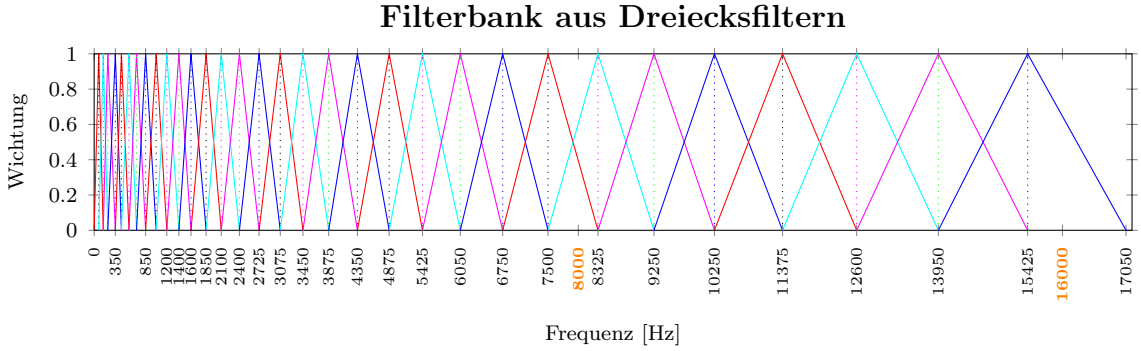


Abbildung 4.1: *Die Grafik zeigt die Filterbank zur Berechnung der 32 MFCC-Koeffizienten. Die gestrichelten Linien markieren die Position der Mittenfrequenz eines Filters und die durchgezogenen Linien den Verlauf der Wichtungsfunktion eines Filters.*

Diese Wahl wurde getroffen, da Spracherkennungssysteme üblicherweise nur mit einem Frequenzbereich von 0 kHz bis 8 kHz arbeiten und eine direkte Manipulation

der MFCC-Koeffizienten für die Sprachverarbeitung möglich sein sollte, beispielsweise durch Subtraktion der geschätzten MFCC-Koeffizienten der Störgeräusche von den original berechneten MFCC-Koeffizienten.

Multi-Layer-Perzeptron zeigen sehr unterschiedliche Verhalten in Abhängigkeit von der Wahl ihrer Hyper-Parameter. Die Wahl der Hyper-Parameter des neuronalen Netzes hat starken Einfluss bezüglich der Fähigkeit des Netzes die erwünschten Zusammenhänge abbilden zu können. Für die Wahl der Hyper-Parameter eines neuronalen Netzen gibt es keine im Allgemeinen gültige Regel, weshalb verschiedene Konfigurationen der Hyper-Parameter überprüft werden müssen, um eine möglichst gute Auswahl treffen zu können. In den folgenden Unterkapiteln werden deshalb verschiedene Konfigurationen von Hyper-Parametern untersucht. Das Ziel ist, eine Konfiguration von Hyper-Parametern zu finden, die zu der Problemstellung gut geeignet ist, aus den verfügbaren Sensor- und Motordaten die auditorische Konsequenz mit einem neuronalen Vorwärtsnetzwerk abzubilden. Zunächst wird mit einigen Experimenten untersucht, wie gut MLPs, mit unterschiedlichen Hyper-Parameter-Konfiguration, den ersten MFCC-Koeffizient aus den verfügbaren Sensordaten vorhersagen können.

Die zu approximierenden MFCC-Koeffizienten werden aus den Audiodaten eines Audiokanals mit Anwendung eines Rechteckfensters von 40 ms Länge berechnet. In jedem Verarbeitungsschritt wird das Fenster um 10 ms verschoben. Dem Vektor aus den berechneten MFCC-Koeffizienten des 40 ms Zeitfensters werden die zwei zeitlich nächsten Sensordatensätze zugeordnet: s_{t_0} am Anfang des Zeitfensters t_0 und s_{t_1} am Ende des Zeitfensters t_1. Die Sensordatensätze enthalten die Gelenkpositionen des jeweiligen Zeitpunktes.

4.1 Experimentalreihe 1: Untersuchung verschiedener Anzahlen an Hidden-Neuronen

In den Experimenten 1.1 bis 1.8 wird die Anzahl der Hidden-Neuronen sukzessive gesteigert. Die Resultate können dann genutzt werden, eine Anzahl von Hidden-Neuronen zu finden, die eine gute Approximation des ersten MFCC-Koeffizienten erlaubt. In allen 8 Experimenten wird ausgehend von dem aktuellen Gelenkwinkel s_{t_0} und der Änderung des Gelenkwinkels $\Delta s = s_{t_1} - s_{t_0}$ eine Vorhersage des ersten MFCC-Koeffizienten berechnet. Es wird je Experiment ein MLP auf einer ca. 80 Sekunden langen Teilaufnahme über 10000 Epochen mit der Resilient Propagation Methode trainiert und mit dem Testdatensatz getestet. Das Training und anschließende Testen wird je Experiment zehn mal wiederholt, um ein Abschätzung über die Stabilität des Lernerfolgs treffen zu können. Die beiden Datensätze enthalten zufällige Auf- und Abbewegungen des linken Roboterarms bei einer eingestellten maximalen Bewegungsgeschwindigkeit von 50 % der maximal möglichen Geschwindigkeit.

Experiment 1.1

In Experiment 1.1 ist die Konfiguration des MLP auf drei Hidden-Neuronen festgelegt. Das entspricht einer Daumenregel, die besagt, dass die Summe der Anzahl der Input-Neuronen und der Output-Neuronen gleich der Anzahl der Hidden Neuronen sein sollte. Der Erfolg des Trainings kann an dem Testresultat, dargestellt in Abbildung 4.2, abgeschätzt werden.

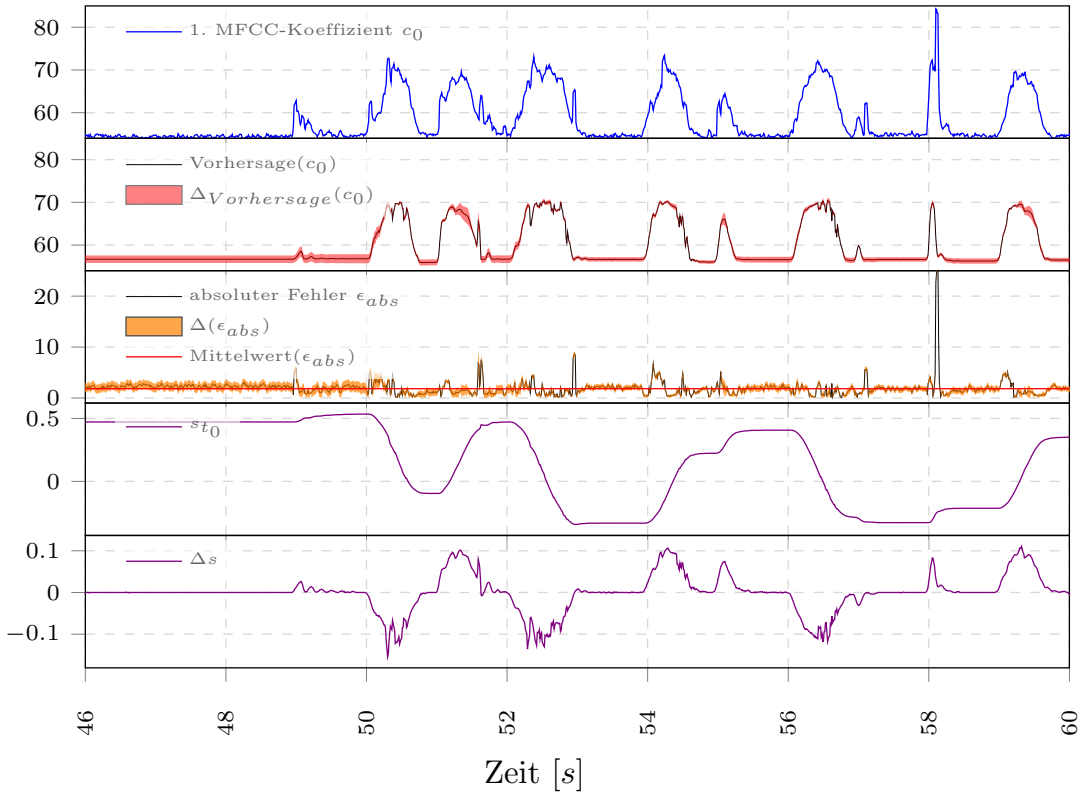


Abbildung 4.2: *Die Grafik zeigt den Lernerfolg der MLP zu Experiment 1.1.*
Der erste MFCC-Koeffizient kann annähernd vorgesagt werden, wobei die Vorhersagen aus den zehn Durchläufen des Experimentes deutliche Abweichungen zueinander aufweisen.
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter folgend den zum Beginn des MFCC-Zeitfensters gemessenen Gelenkwinkel s_{t_0} und in der untersten Grafik ist die Änderung des Gelenkwinkels Δs abgebildet.

Man kann an dem Verlauf der Vorhersage des ersten MFCC-Koeffizienten und dem darunter abgebildeten absoluten Fehler erkennen, dass die Vorhersage einen zum ersten MFCC-Koeffizienten ähnlichen Verlauf aufweist. Man sieht auch, dass nicht alle Anteile des ersten MFCC-Koeffizienten durch das MLP vorhergesagt werden können. Ursache hierfür können andere Störquellen sein, wie beispielsweise Sensorrauschen,

der Lüfter oder Geräusche die durch die Beanspruchung des Materials des Roboters entstehen. Der große Fehler beim Zeitpunkt von ca. 58 Sekunden ist unter Umständen dem MFCC-Koeffizienten-Anteil einer Kollision des Arms mit dem Körper des Roboters zuzuschreiben. Es fällt auch auf, dass die Vorhersagen in den zehn Durchläufe des Experimentes deutlich variieren.

Experiment 1.2

Für Experiment 1.2 wurde die Anzahl der Hidden-Neuronen der MLPs auf vier erhöht, um zu sehen, ob eine leichte Veränderung der Daumenregel nach oben eine Veränderung der Vorhersagequalität erreichen kann. Abbildung 4.3 illustriert die Vorhersagequalität dieser MLPs.

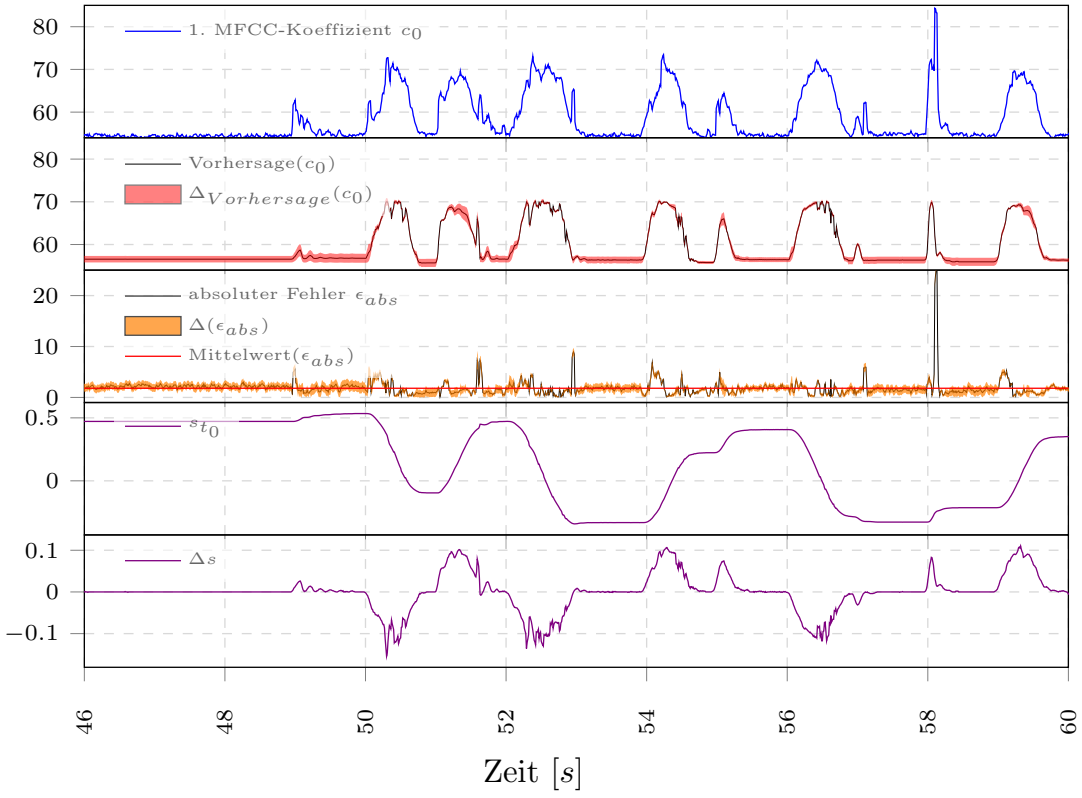


Abbildung 4.3: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 1.2. Im Vergleich mit Abbildung 4.2 sind keine wesentlichen Unterschiede zu sehen. Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter folgend den zum Beginn des MFCC-Zeitfensters gemessenen Gelenkwinkel s_{t_0} und in der untersten Grafik ist die Änderung des Gelenkwinkels Δs abgebildet.*

Auch hier kann man an dem Verlauf der Vorhersage des ersten MFCC-Koeffizienten und dem darunter abgebildeten absoluten Fehler erkennen, dass die Vorhersage un-

gefähr dem ersten MFCC-Koeffizienten entspricht. Es sind keine wesentlichen Unterschiede zum Ergebnis von Experiment 1.1 zu sehen. Der mittlere Fehler ist bei beiden Experimenten mit ungefähr 2 % bis 3 % nicht sehr groß.

Experiment 1.3

Die Anzahl der Hidden-Neuronen der MLPs wurde für Experiment 1.3 auf fünf erhöht. Wie weit sich die Vorhersage verbessert, ist in Abbildung 4.4 dargestellt.

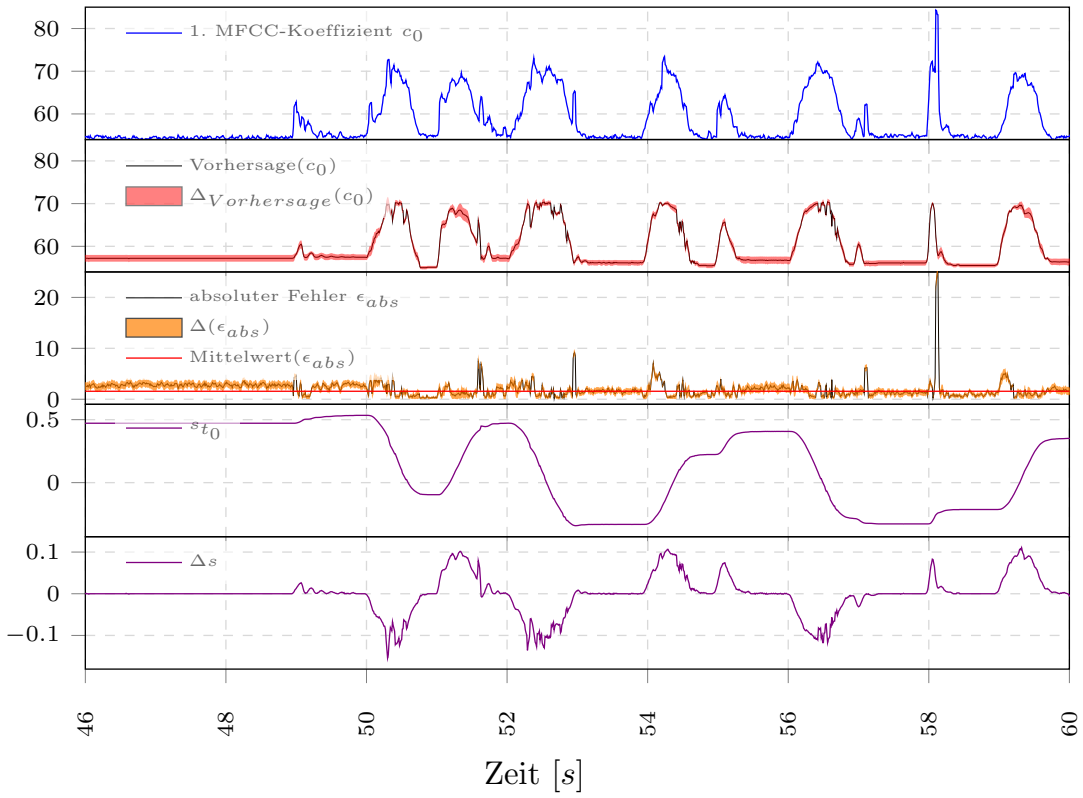


Abbildung 4.4: *Dargestellt ist der Lernerfolg der MLPs zu Experiment 1.3. Verglichen mit den Abbildungen 4.2 und 4.3 ist eine leichte Verbesserung der Vorhersage zu sehen. Deutlich wird die leichte Verbesserung der Vorhersage unter anderem in dem Zeitbereich um die 49. Sekunde herum.*
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter folgend den zum Beginn des MFCC-Zeitfensters gemessenen Gelenkwinkel s_{t_0} und in der untersten Grafik ist die Änderung des Gelenkwinkels Δs abgebildet.

Erneut kann man an dem Verlauf der Vorhersage des ersten MFCC-Koeffizienten und dem darunter abgebildeten absoluten Fehler erkennen, dass die Vorhersage ungefähr dem ersten MFCC-Koeffizienten gleicht. An dem kleinen Zeitabschnitt um die 49. Sekunde ist eine leichte Verbesserung der Vorhersage, verglichen mit den vorhergehenden

Ergebnissen, zu sehen. Die Vorhersagen aus den zehn Durchläufen des Experimentes variieren ähnlich stark, wie zuvor.

Experiment 1.4

Die Anzahl der Hidden-Neuronen des MLP wurde für Experiment 1.4 ein weiteres Mal leicht erhöht. Es werden nun sechs Hidden-Neuronen genutzt.

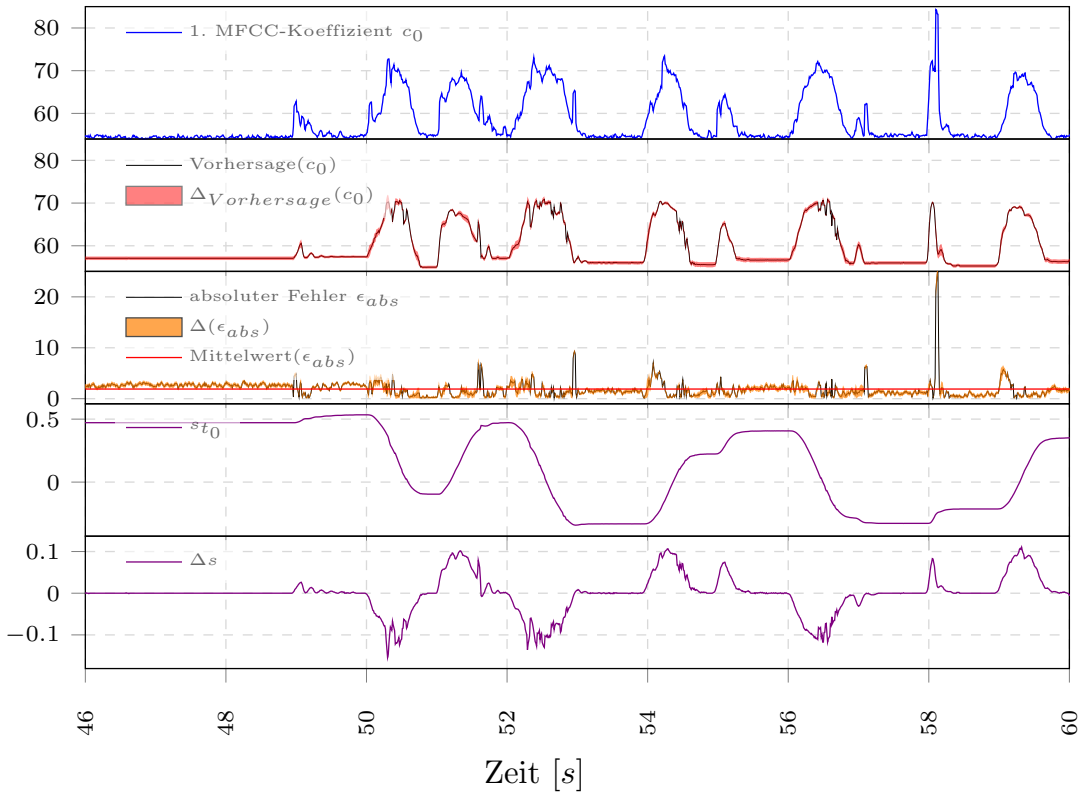


Abbildung 4.5: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 1.4.*
Man sieht, dass die Vorhersagen des ersten MFCC-Koeffizienten weniger stark variiert, als in den vorhergehenden Experimenten.
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter folgend den zum Beginn des MFCC-Zeitfensters gemessenen Gelenkwinkel s_{t_0} und in der untersten Grafik ist die Änderung des Gelenkwinkels Δs abgebildet.

Auch bei dem vierten Experiment kann man an den Resultaten erkennen, dass die Vorhersage dem des ersten MFCC-Koeffizienten annähernd entspricht. Die Vorhersage des ersten MFCC-Koeffizienten variiert deutlich weniger, im Vergleich zum ersten bis dritten Experiment.

Experiment 1.5

Im fünften Experiment wurde die Anzahl der Hidden-Neuronen des MLP ein weiteres Mal um Eines erhöht. In jedem Durchlauf des Experiments sind somit sieben Hidden-Neuronen in dem Hidden-Layer des MLP vorhanden.

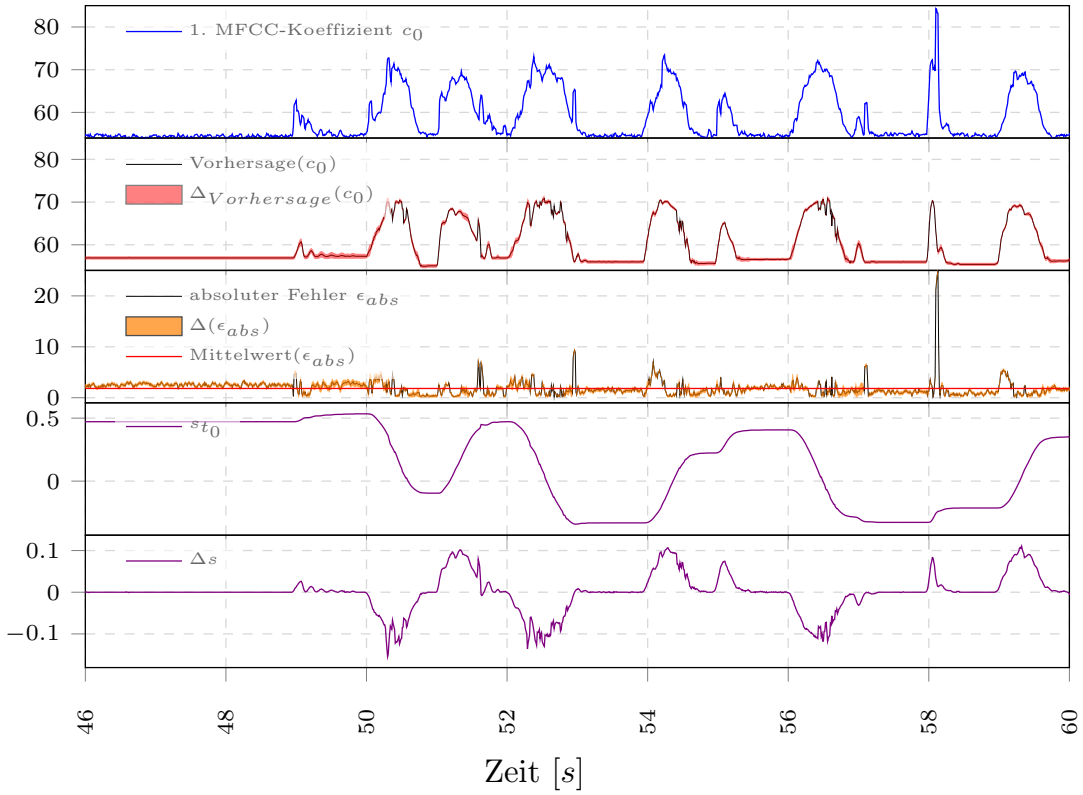


Abbildung 4.6: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 1.5.*
Im Vergleich zu Abbildung 4.5 ist eine Änderung der Vorhersagen kaum erkennbar. Bei genauerer Betrachtung dieser Abbildung und der Abbildungen der vorherigen Experimente fällt auf, dass die MLPs, zu Zeitpunkten an denen keine Bewegung erfolgt, zu große MFCC-Koeffizienten vorhersagen.
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter folgend den zum Beginn des MFCC-Zeitfensters gemessenen Gelenkwinkel s_{t_0} und in der untersten Grafik ist die Änderung des Gelenkwinkels Δs abgebildet.

Die Vorhersage des ersten MFCC-Koeffizienten ist sehr ähnlich zu der aus Experiment 1.4. Bei genauerer Betrachtung der Abbildungen 4.2 bis 4.6 fällt auf, dass die MLPs, zu Zeitpunkten an denen keine Bewegung erfolgt, zu große MFCC-Koeffizienten vorhersagen, während sonst die Vorhersagen recht nahe an den vorherzusagenden MFCC-Koeffizienten liegen. Zu solchen Zeitpunkten ist hauptsächlich noch das Geräusch des Lüfters im Audiosignal vorhanden. Die MLPs sind also eher in der Lage die Bewegungsgeräusche der Gelenkmotoren abzubilden, als die Geräusche des Lüfters. Wobei

das Lüftergeräusch komplett zu erfassen nicht vorgesehen und auch nicht zu erwarten ist.

Experiment 1.6

Im Experiment 1.6 werden MLPs mit 11 Hidden-Neuronen untersucht.

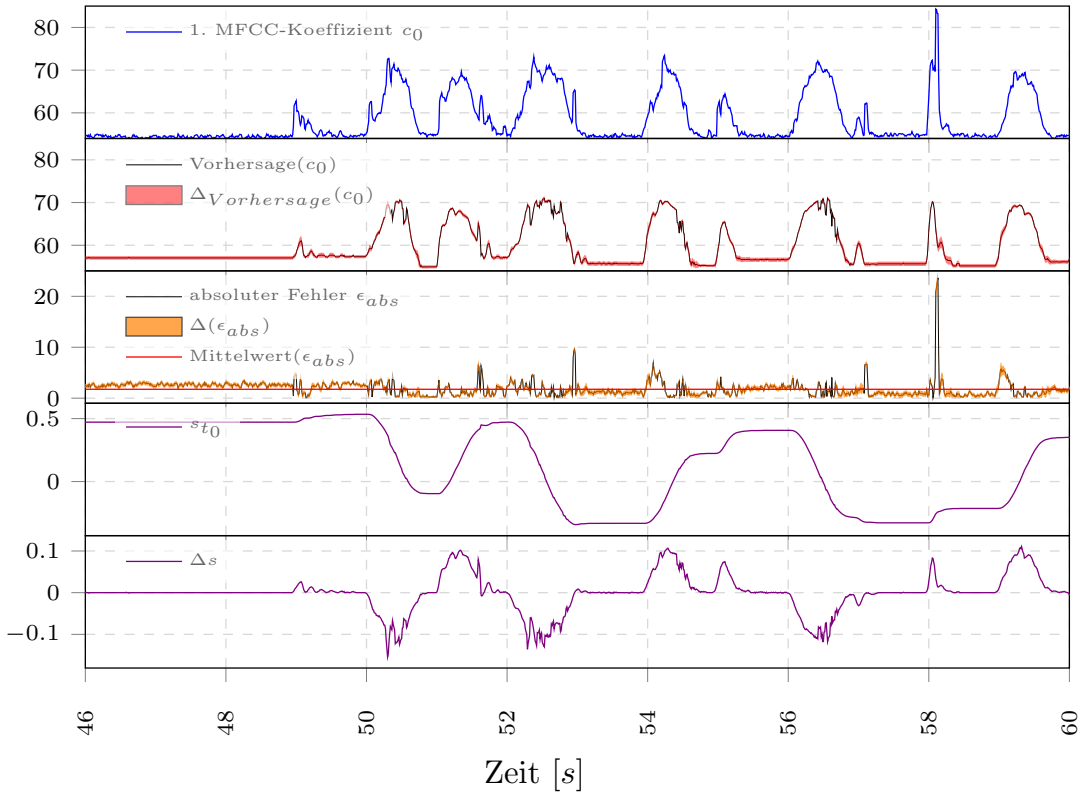


Abbildung 4.7: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 1.6. Eine Verbesserung der Vorhersage im Vergleich zu Abbildung 4.6 ist an der Vorhersage kaum ersichtlich. Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter folgend den zum Beginn des MFCC-Zeitfensters gemessenen Gelenkwinkel s_{t_0} und in der untersten Grafik ist die Änderung des Gelenkwinkels Δs abgebildet.*

Erneut ist die Verbesserung der Vorhersage so marginal, dass diese anhand der Darstellungen der Vorhersage des ersten MFCC-Koeffizienten kaum erkennbar ist. Es stellt sich die Frage, ob eine deutlich höhere Anzahl an Hidden-Neuronen auch eine deutlichere Verbesserung der Vorhersage zu Folge hat.

Experiment 1.7

Da die Vorhersagen bisher nur eine leichte Verbesserung brachten , wird die Anzahl der Hidden-Neuronen für Experiment 1.7 deutlich auf 33 erhöht.

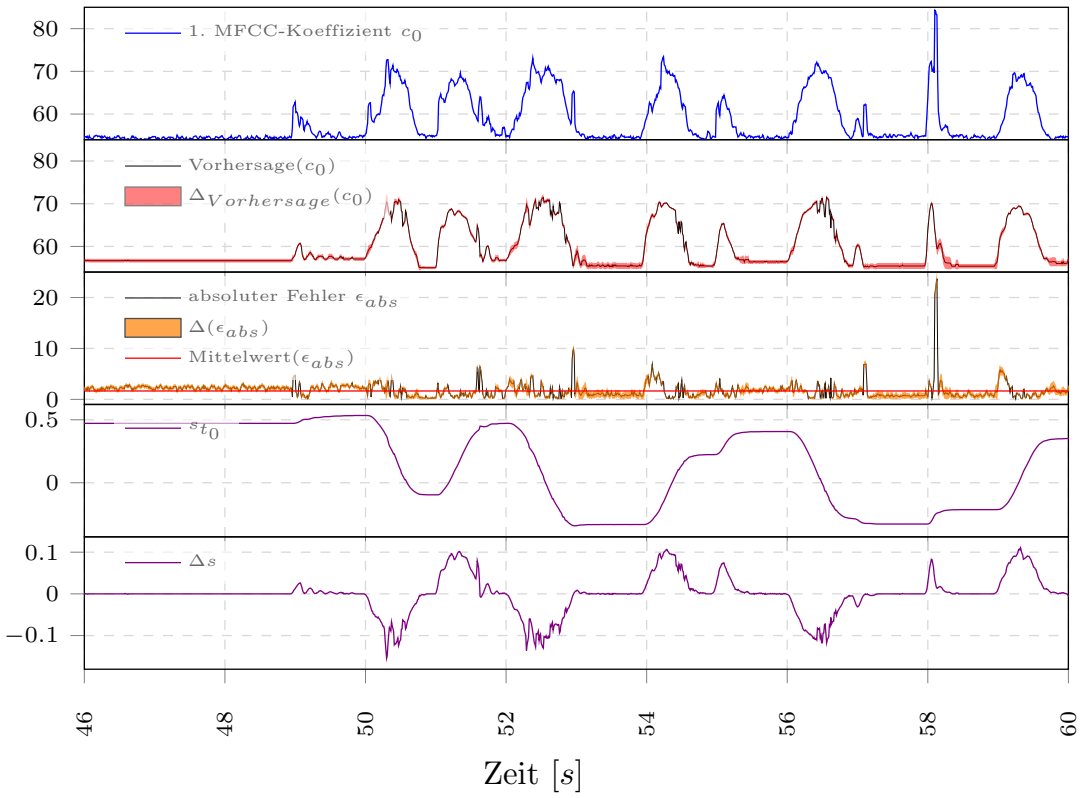


Abbildung 4.8: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 1.7.*
Trotz der wesentlich größeren Anzahl an Hidden-Neuronen, ist die Verbesserung der Vorhersage erneut recht gering.
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter folgend den zum Beginn des MFCC-Zeitfensters gemessenen Gelenkwinkel s_{t_0} und in der untersten Grafik ist die Änderung des Gelenkwinkels Δs abgebildet.

Nachdem das MLP in allen bisherigen Konfigurationen in der Lage war den ersten MFCC-Koeffizienten auf Basis der aktuellen Gelenkstellung s_{t_0} und der Gelenkwinkeländerung Δs anzunähern, steht fest, dass sich s_{t_0} und Δs dazu prinzipiell als Eingangsdaten für die MLPs eignen. Die Frage ist nun: kann eine weitere deutliche Erhöhung der Anzahl der Hidden-Neuronen überhaupt noch eine Verbesserung bringen?

Experiment 1.8

Die Anzahl der Hidden-Neuronen ist für Experiment 1.8 auf 100 festgelegt. Da in allen bisherigen Experimenten jedes Perzeptron in der Lage war den ersten MFCC-Koeffizienten annähernd vorherzusagen, kann man erwarten, dass dies erneut der Fall sein wird. Das Hauptinteresse in diesem Experiment besteht darin, ob noch eine Verbesserung der Vorhersage erreicht werden kann oder ob eventuell eine Verschlechterung eintritt.

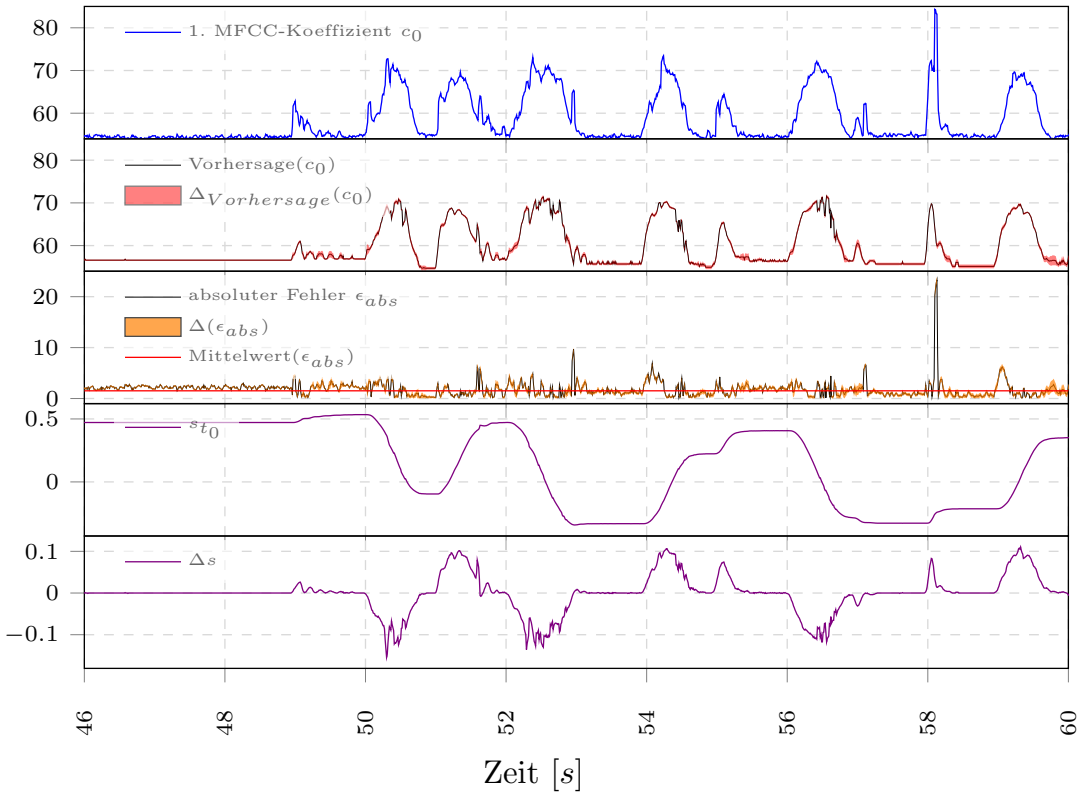


Abbildung 4.9: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 1.8.*
Der deutlichste Unterschied zu den vorhergehenden Experimenten ist, dass die Vorhersagen der zehn Durchläufe dieses Experiments zueinander weniger stark variieren.
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter folgend den zum Beginn des MFCC-Zeitfensters gemessenen Gelenkwinkel s_{t_0} und in der untersten Grafik ist die Änderung des Gelenkwinkels Δs abgebildet.

Wie erwartet kann ein 100-Neuronen-MLP eine annähernde Vorhersage des ersten MFCC-Koeffizienten vornehmen. Es ist auch eine leichte Verbesserung der Vorhersage zu verzeichnen, wobei der deutlichste Unterschied zum vorhergehenden Experiment ist, dass die Vorhersagen der zehn Durchläufe dieses Experiments zueinander weniger stark variieren als zuvor. Auf den ersten Blick scheint ein 100-Neuronen-MLP die beste

Approximation des ersten MFCC-Koeffizienten aus den Gelenkdaten des Roboters abbilden zu können.

Auswertung der Experimentalreihe 1

Da die Vorhersage des ersten MFCC-Koeffizienten allen Konfigurationen ähnlich gut gelingt und um abschätzen zu können, wie stark die Varianz der Vorhersage und des absoluten Fehlers sind, also wie stabil der Lernerfolg ist, wurden mehrere Durchläufe eines Experiments durchgeführt. Jedes Experiment wurde jeweils zehn mal durchgeführt. Dazu wurde bei jeder Durchführung eines Experiments das MLP randomisiert, neu trainiert und der mittlere quadratische Fehler über den gesamten Testdatensatz für jede Durchführung eines Experimentes berechnet. Abbildung 4.10 illustriert, wie sich der mittlere quadratische Fehler über die zehn Durchläufe je Experiment verteilt.

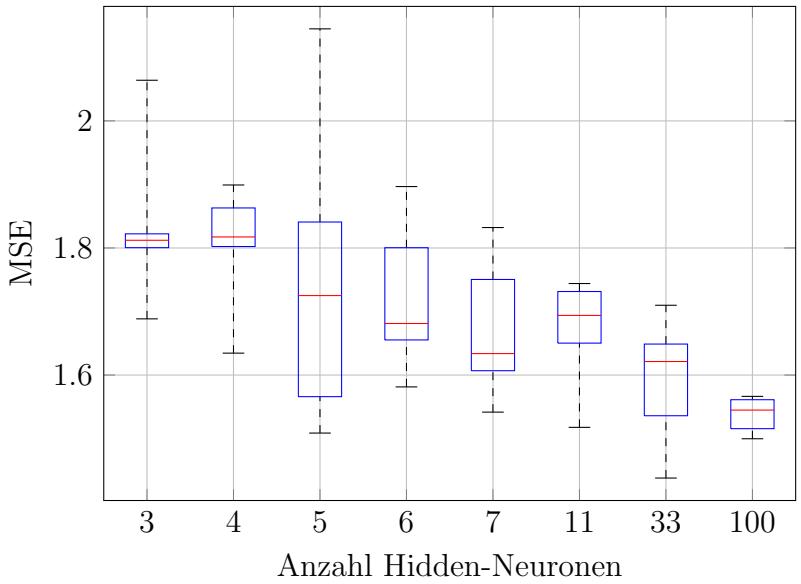


Abbildung 4.10: *Die Darstellung zeigt die Verteilung der mittleren quadratischen Fehler der Vorhersagen des ersten MFCC-Koeffizienten, ermittelt aus zehn Durchführungen je Experiment. Man sieht, dass die Varianz des mittleren quadratischen Fehlers zunächst ansteigt und dann wieder fällt, während der Median des mittleren quadratischen Fehlers meist absinkt.*
Rote Linien markieren den Median, blaue Boxen den Bereich zwischen dem 25 %-Quartil und dem 75 %-Quartil und die Markierungen am Ende der gestrichelten Linien den kleinsten und den größten durchschnittlichen Fehler.

Vergleicht man die Abbildungen 4.2 bis 4.9, ist ersichtlich, dass sich mit zunehmender Anzahl an Hidden-Neuronen den Unterschied zwischen dem kleinsten und größten vorhergesagten Koeffizientenwert eines Zeitpunktes verringert. Daraus folgt sofort, dass auch der absolute Fehler der Vorhersage mit steigender Hidden-Neuronen-Anzahl weniger variiert, was in den Abbildungen ebenfalls zu sehen ist. Die Darstellungen zeigen zudem eine leichte Verbesserung der Approximation des ersten MFCC-Koeffizienten.

Man sieht in Abbildung 4.10, dass die Varianz des mittleren quadratischen Fehlers zunächst ansteigt und dann wieder fällt, während der Median des mittleren quadratischen Fehlers mit wenigen Ausnahmen absinkt.

Bei Betrachtung der Abbildungen 4.2 bis 4.9 fällt auch auf, dass der Verlauf des absoluten Fehlers sehr verrauscht ist, an einigen Stellen zudem sehr groß. Stellen an denen der Fehler sehr groß ist, könnten anderen Geräuschquellen zugeordnet werden, die in der Ansteuerung der Motoren nicht ihre direkte Ursache haben.

Der kleinste mittlere quadratische Fehler trat in einem Durchlauf des Experiments 1.7 auf. Obwohl ein MLP mit 33 Hidden-Neuronen potentiell den kleinsten mittleren quadratischen Fehler erreichen kann, scheidet es als Kandidat aus, aufgrund der im Vergleich recht großen Streuung und der Schiefe der Verteilung, die zu eher größeren Fehlern hin tendiert, sichtbar am Median. Die größte Streuung und Varianz trat für mittlere quadratische Fehler in Experiment 1.3 auf. Diese Konfiguration ist die im Vergleich schlechteste, obwohl sie auch sehr kleine Fehlerwerte vorweisen kann.

Die beiden Konfigurationen mit der geringsten Varianz der Fehler sind nach Abbildung 4.10 die Konfiguration mit drei Hidden-Neuronen aus Experiment 1.1 und die Konfiguration mit 100 Hidden-Neuronen aus Experiment 1.8. Der Median der Fehler für die MLPs aus Experiment 1.8 hat zudem den kleinsten Wert und die Varianz ist im Vergleich recht gering. Dieses Konfiguration weist außerdem auch die geringste Streuung auf. Von den acht untersuchten Konfigurationen ist damit ein 100-Hidden-Neuronen-MLP das Stabilste und Genaueste dieser Testreihe.

4.2 Experimentalreihe 2: Untersuchung verschiedener Belegungen und Anzahlen der Input-Neuronen

In den nächsten Experimenten werden verschiedene Konfigurationen der Belegung und Anzahl der Input-Neuronen untersucht. Die Anzahl der Hidden-Neuronen ist für die folgenden Experimente auf 100 festgelegt, da damit in Experimentalreihe 1 das beste Ergebnis erzielt wurde.

Das Training und anschließende Testen wird je Experiment zehn mal wiederholt, um ein Abschätzung über die Stabilität des Lernerfolgs treffen zu können. Je Experiment und Durchlauf wird erneut ein MLP auf einer ca. 80 Sekunden langen Teilaufnahme des Trainingsdatensatzes über 10000 Epochen mit der Resilient Propagation Methode trainiert und auf dem vollständigen Testdatensatz überprüft. Beide Datensätze enthalten zufällige Auf- und Abbewegungen des linken Roboterarms bei einer eingestellten maximalen Bewegungsgeschwindigkeit von 50 % der maximal möglichen Geschwindigkeit. Als Transferfunktion kommt erneut der Tangens-Hyperbolicus zu Einsatz. Der zu approximierenden MFCC-Koeffizient ist weiterhin der erste von maximal 32 möglichen Koeffizienten, die aus den Audiodaten eines Audiokanals mit Anwendung eines Rechteckfensters von 40 ms Länge berechnet werden. Die Verschiebung des Zeitfenster ändert sich ebenfalls nicht und verbleibt bei 10 ms. Dem Vektor aus den berechneten MFCC-Koeffizienten des Zeitfensters werden wieder die zwei zeitlich nächsten Sensordatensätze zugeordnet: s_{t_0} am Anfang des Zeitfensters t_0 und s_{t_1} am Ende des Zeitfensters t_1. In den Experimenten wird ausgehend von dem aktuellen Gelenkwinkel s_{t_0}, der Änderung des Gelenkwinkels $\Delta s = s_{t_1} - s_{t_0}$ oder Kombinationen daraus, auch mit Einbeziehung zeitlich vorher gelegener Werte, eine Vorhersage des ersten MFCC-Koeffizienten berechnet.

Zweck dieser Experimentalreihe ist herauszufinden, welche Kombination der verfügbaren Sensordaten, im Sinne der Approximation nur eines MFCC-Koeffizienten, die Beste ist. Eine mögliche Kombination ist in den Experimenten der Reihe 1 schon probiert worden. Für eine gute Vergleichbarkeit werden deshalb alle Parameter der MLPs, bis auf die Anzahl und Belegung der Input-Neuronen, auf die Hyper-Parameter des MLP aus Experimentalreihe 1 festgelegt, das am Besten abgeschnitten hat. Die ersten drei Experimente nutzen nur Sensordaten des aktuellen Zeitpunktes t_0. Die anschließend folgenden drei Experimente beziehen Sensordaten vorheriger Zeitpunkte t_{-1} bis t_{-3} mit ein, um zu sehen, was durch Hinzunahme von Daten aus mehreren vorherigen Zeitschritten erreicht werden kann.

Experiment 2.1

In diesem ersten Experiment der zweiten Experimentalreihe werden die MLPs nur ein Input-Neuron verwenden, das mit der Gelenkwinkeländerung Δs belegt ist.

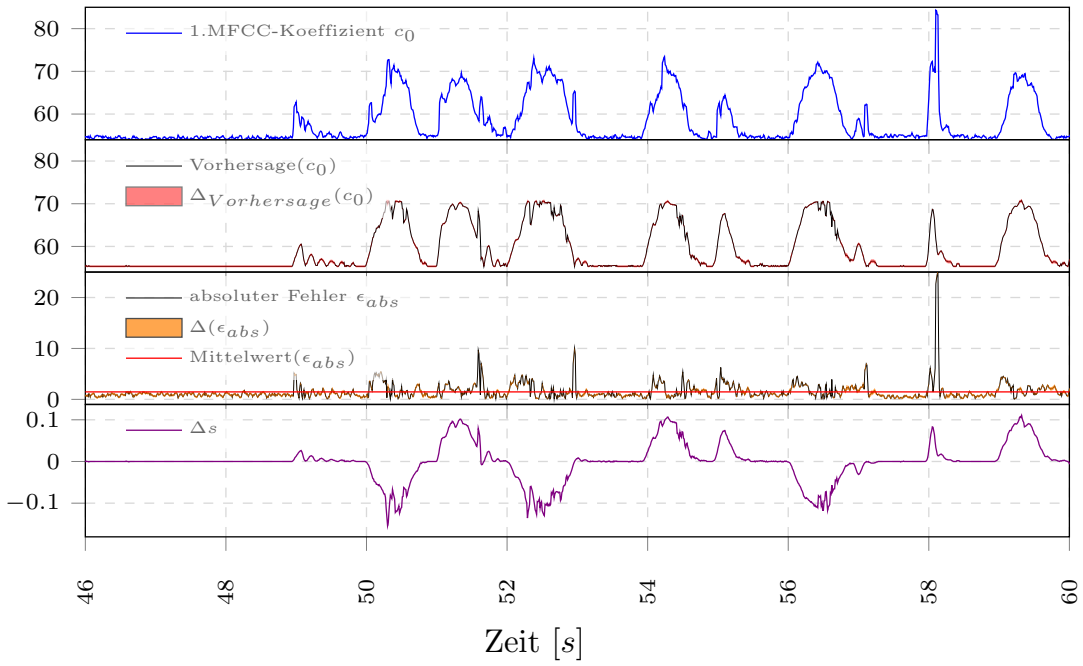


Abbildung 4.11: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 2.1.*
Im Vergleich zu Experiment 1.8 (Abbildung 4.9) ist zu sehen, dass nur Δs zu verwenden eine ähnlich genaue und weniger variierende Vorhersage erlaubt.
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert) und in der untersten Grafik ist die Änderung des Gelenkwinkels Δs abgebildet.

Wie in Abbildung 4.11 zu sehen ist, ist es ausreichend nur die Änderung des Gelenkwinkels Δs zu verwenden, um den ersten MFCC-Koeffizienten vorherzusagen. Die Vorhersage ist augenscheinlich sogar etwas besser, als bei Verwendung von Δs und $|\Delta s|$. Inwiefern die Hinzunahme weiterer verfügbarer Sensordaten ein Verbesserung oder Verschlechterung bewirkt, muss am Ende wieder im zusätzlichen Vergleich der Verteilungen der mittleren quadratischen Fehler über die zehn Durchläufe je Experiments abgeschätzt werden.

Experiment 2.2

In Experiment 2.2 wird das Input-Neuron eines MLPs mit dem aktuellen Gelenkwinkel s_{t_0} belegt.

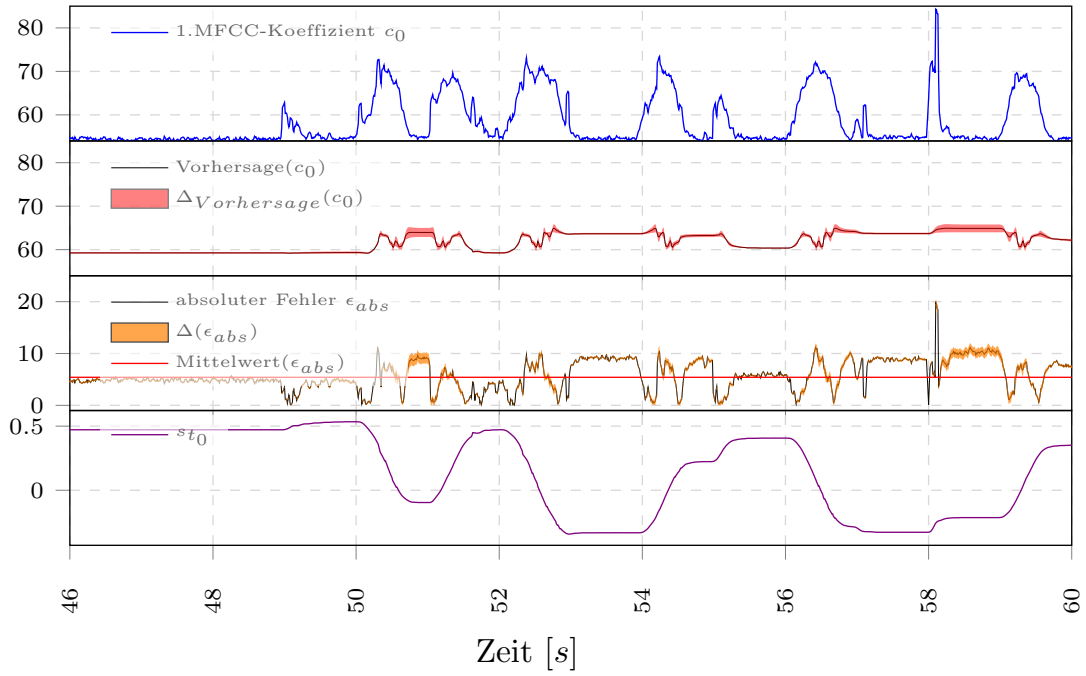


Abbildung 4.12: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 2.2.*
Es ist sehr deutlich zu sehen, dass die Vorhersage nicht dem ersten MFCC-Koeffizienten entspricht.
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert) und in der untersten Grafik ist der zum Beginn des MFCC-Zeitfensters gemessene Gelenkwinkel s_{t_0} abgebildet.

Am Verlauf des Fehlers und auch an der Vorhersage des ersten MFCC-Koeffizienten in Abbildung 4.12 ist gut zu sehen, dass allein der aktuelle Gelenkwinkel s_{t_0} nicht ausreicht, um den MFCC-Koeffizienten vorherzusagen. Dies gilt zumindest für die gewählte Konfiguration aus Lernrate, Lernverfahren, Länge der Lernphase und so weiter.

Experiment 2.3

In Experiment 2.3 wird nun das Input-Neuron eines MLPs mit dem Betrag der Gelenkwinkeländerung $|\Delta s|$ belegt. Vergleicht man den Betrag der Gelenkwinkeländerung mit dem ersten MFCC-Koeffizienten in Abbildung 4.13, ist der recht ähnliche Verlauf beider Kurven auffallend. Diese Ähnlichkeit lässt erwarten, dass die Verwendung des Betrages der Gelenkwinkeländerung geeignet sein sollte, den ersten MFCC-Koeffizienten vorherzusagen.

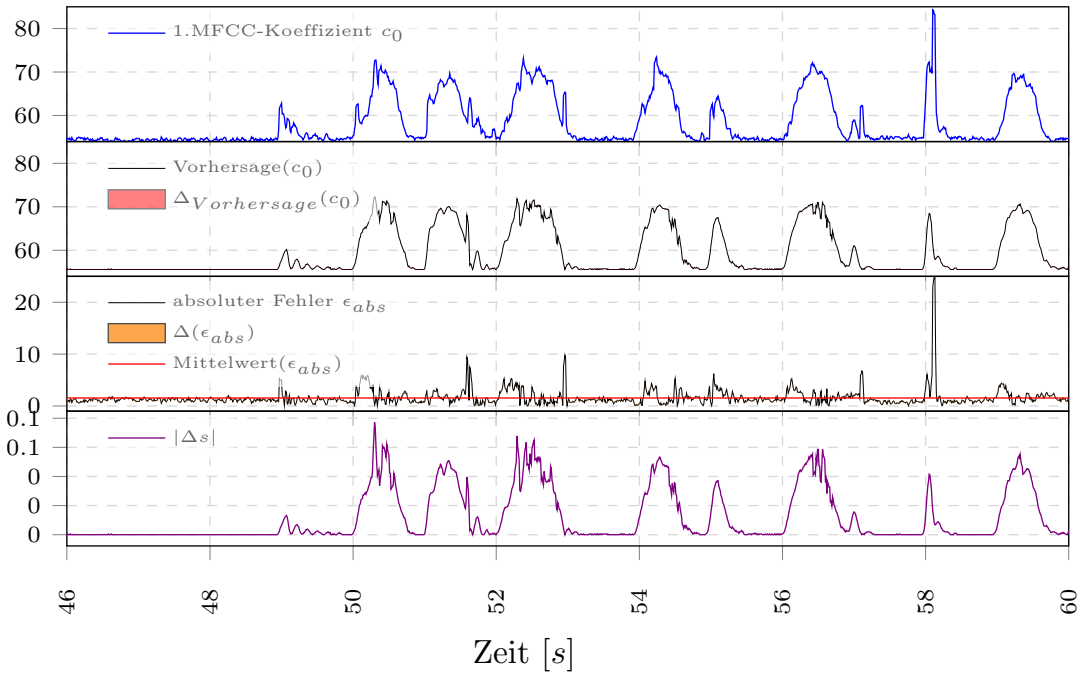


Abbildung 4.13: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 2.3. Man sieht deutlich, wie sehr sich der Verlauf des ersten MFCC-Koeffizienten und der Verlauf von $|\Delta s|$ ähnlich sind. Außerdem variieren der absolute Fehler und die Vorhersage so wenig, dass es in der dargestellten Grafik so gut wie nicht sichtbar ist. Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert) und in der untersten Grafik ist der Betrag der Gelenkwinkeländerung $|\Delta s|$ abgebildet.*

Es zeigt sich, dass der Betrag der Gelenkwinkeländerung $|\Delta s|$, wie vermutet, geeignet ist, um den ersten MFCC-Koeffizienten annähernd vorherzusagen. Interessant ist hierbei, ob die Verwendung von $|\Delta s|$ anstatt von Δs als Belegung des Input-Neurons bessere Ergebnisse liefern kann oder nicht. Falls die Belegung des Input-Neurons mit Δs schlechter abschneiden würde, wäre dies ein Hinweis darauf, dass die akustischen Unterschiede in Abhängigkeit von der Bewegungsrichtung nicht oder zu wenig in den MFCC-Koeffizienten kodiert enthalten sind oder das Unterschiede in der Bewegungsrichtung akustisch keinen wesentlichen unterscheidbaren Effekt haben.

Experiment 2.4

Für Experiment 2.4 werden vier Input-Neuronen verwendet. Die vier Input-Neuronen sind mit den Gelenkwinkeln der Zeitpunkte t_0, t_{-1}, t_{-2} und t_{-3} belegt.

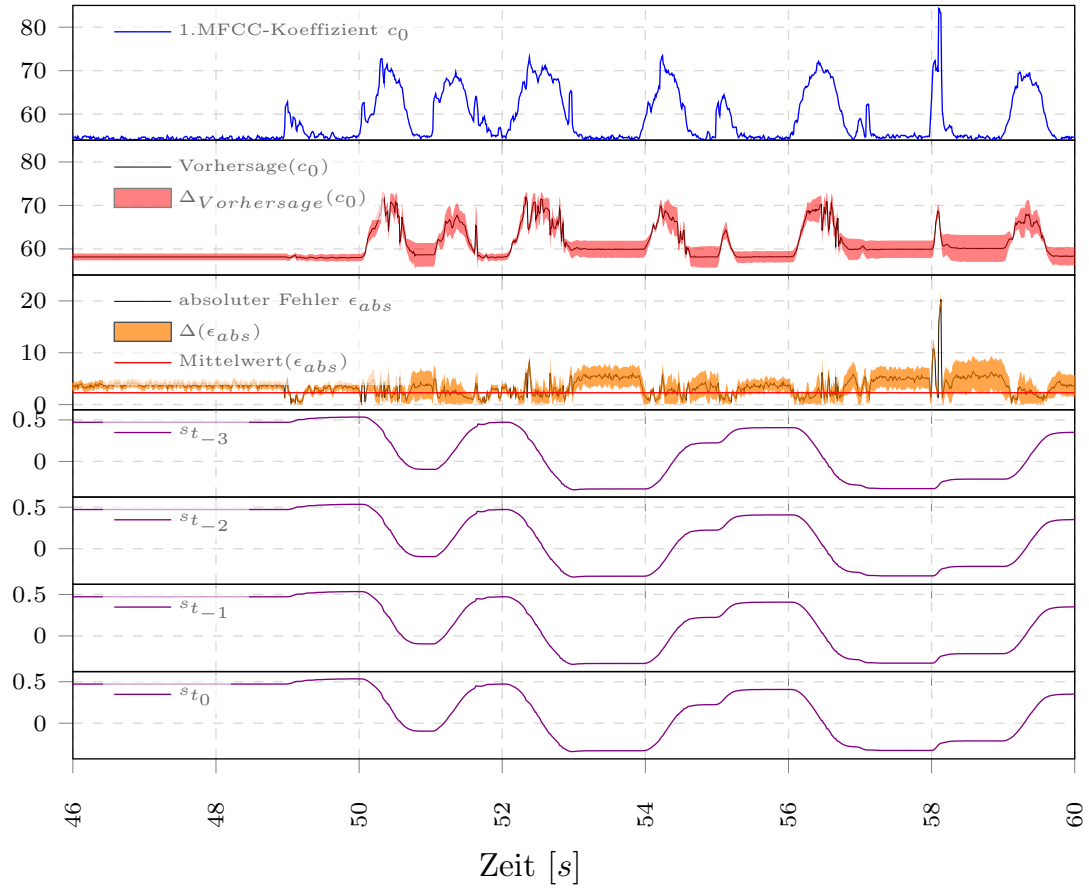


Abbildung 4.14: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 2.4. Man sieht, dass Gelenkwinkel mehrerer Zeitpunkte einzubeziehen eine Verbesserung im Vergleich zu Abbildung 4.12 darstellt. Die Vorhersagen variieren jedoch zwischen Durchführungen des Experimentes sehr stark.*
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert) und in den unteren vier Grafiken ist der Gelenkwinkel zu den Zeitpunkten t_{-3}, t_{-2}, t_{-1} und t_0 abgebildet.

In Abbildung 4.14 wird deutlich, dass die Hinzunahme von Gelenkwinkeln mehrerer vergangener Zeitpunkte zum aktuellen Gelenkwinkel s_{t_0}, im Vergleich zum Ergebnis von Experiment 2.2, viel besser geeignet ist, um den ersten MFCC-Koeffizienten vorhersagen zu können. Die Genauigkeit und Invariabilität der MLPs aus Experiment 2.1 oder 2.3 werden aber nicht erreicht.

Experiment 2.5

Für Experiment 2.5 werden wieder vier Input-Neuronen verwendet. Die vier Input-Neuronen sind mit der aktuellen Änderung des Gelenkwinkels Δs und den Gelenkwinkeln s der Zeitpunkte t_0, t_{-1} und t_{-2} belegt.

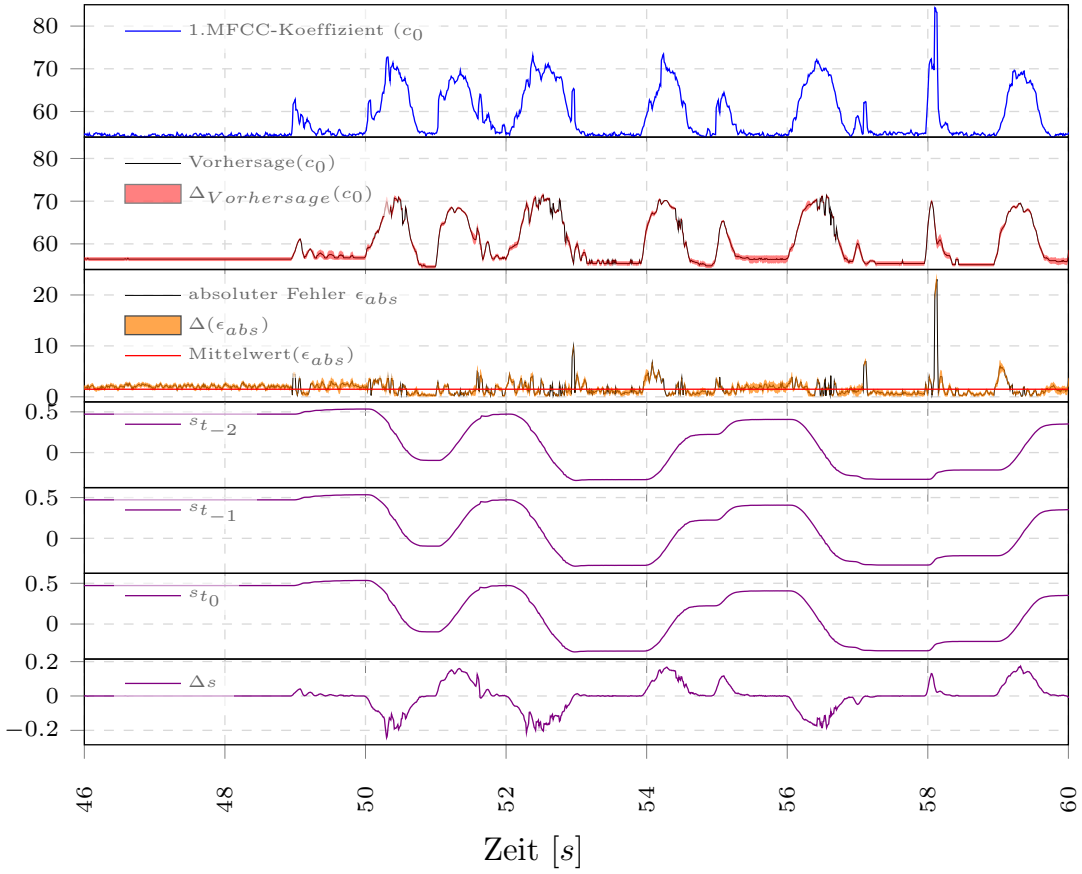


Abbildung 4.15: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 2.5. Man sieht deutlich, dass die Vorhersage weniger variiert als in Abbildung 4.14. Die Belegung der Input-Neuronen entspricht der aus Experiment 1.8, unter Hinzunahme von 2 Gelenkwinkeln vorheriger Zeitpunkte.*
Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert) und in den unteren vier Grafiken sind die Gelenkwinkel der Zeitpunkte t_{-2}, t_{-1}, t_0 und die Änderung des Gelenkwinkels Δs abgebildet.

Diese Hyper-Parameterkonfiguration entspricht der Konfiguration aus Experiment 1.8, unter Hinzunahme von 2 Gelenkwinkeln vorheriger Zeitpunkte. Dass mit dieser Konfiguration der erste MFCC-Koeffizient vorhergesagt werden kann, ist aufgrund des Ergebnisses von Experiment 1.8 zu erwarten. Von Interesse ist bei diesem Experiment, ob es ein genaueres oder stabileres Vorhersageverhalten hat. Ein Abschätzung dazu wird in der später folgenden Auswertung vorgenommen.

Experiment 2.6

Für Experiment 2.6 werden wieder vier Input-Neuronen verwendet. Die vier Input-Neuronen sind mit den Änderung des Gelenkwinkels Δs der Zeitpunkte t_0, t_{-1}, t_{-2} und t_{-3} belegt.

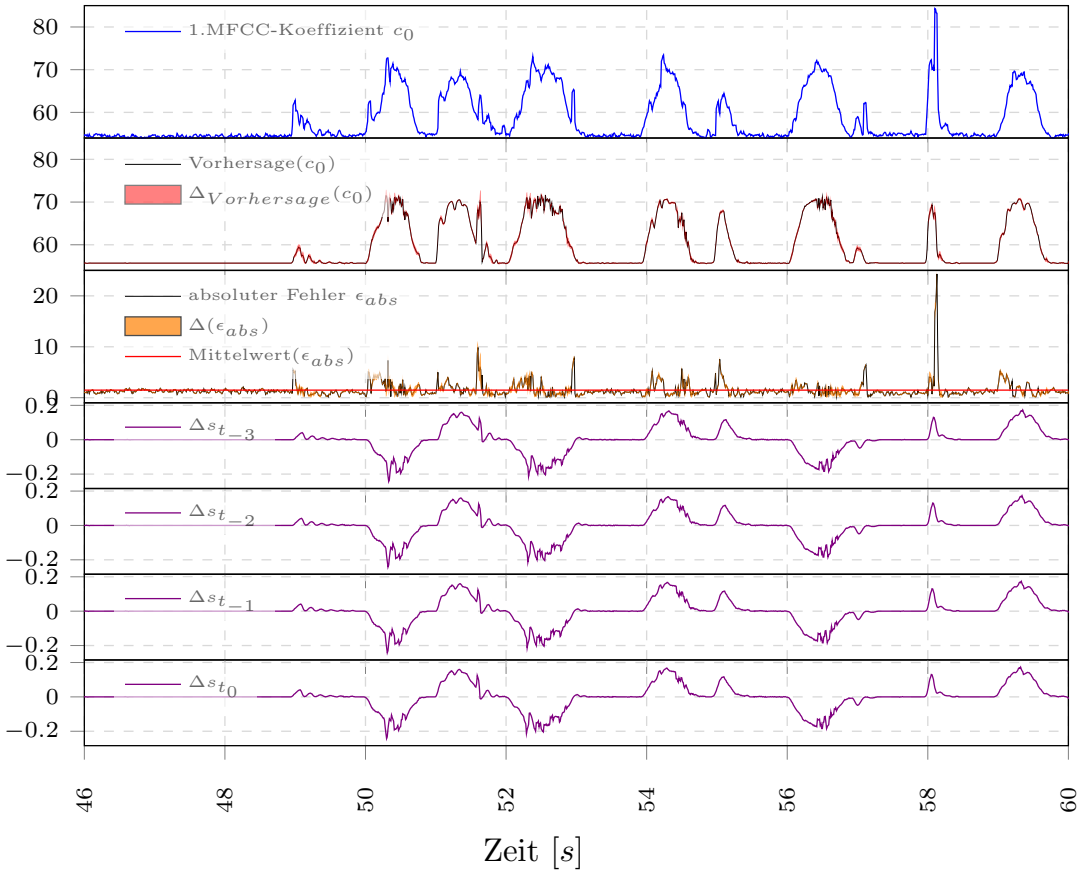


Abbildung 4.16: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 2.6. Die oberste Grafik zeigt den zu approximierenden ersten MFCC-Koeffizienten c_0, direkt darunter ist dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert), darunter den absoluten Fehler (orange = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert) und in den unteren vier Grafiken ist die Änderung des Gelenkwinkels Δs zu den Zeitpunkten t_{-2}, t_{-1}, t_0 abgebildet.*

Auch bei diesem Experiment ist zu erwarten, dass der erste MFCC-Koeffizient aus den Gelenkwinkeländerungen abgebildet werden kann. Hier ist erneut von Interesse wie und ob sich das Vorhersageverhalten des Netzes im Vergleich zum Netz aus Experiment 2.1 ändert, wenn mehr Informationen aus vorherigen Zeitpunkten mit einfließt.

Auswertung der Experimentalreihe 2

Ziel der zweiten Experimentalreihe ist es zu ermitteln, welche Konfiguration an Eingangsdaten für ein Multi-Layer-Perzeptron die im Vergleich beste Vorhersage des ersten MFCC-Koeffizienten liefern kann. Wie in der ersten Experimentalreihe, ist eine einzelne Durchführung eines Experiments nicht genügend aussagekräftig, um die Vorhersagegüte abzuschätzen. Jedes Experiment wurde deshalb zehn mal ausgeführt. Dazu wurde bei jeder Durchführung eines Experiments das MLP randomisiert, neu trainiert und der mittlere Fehler über den gesamten Testdatensatz berechnet. Abbildung 4.17 illustriert, wie sich der mittlere quadratische Fehler über die zehn Durchläufe je Experiments verteilt.

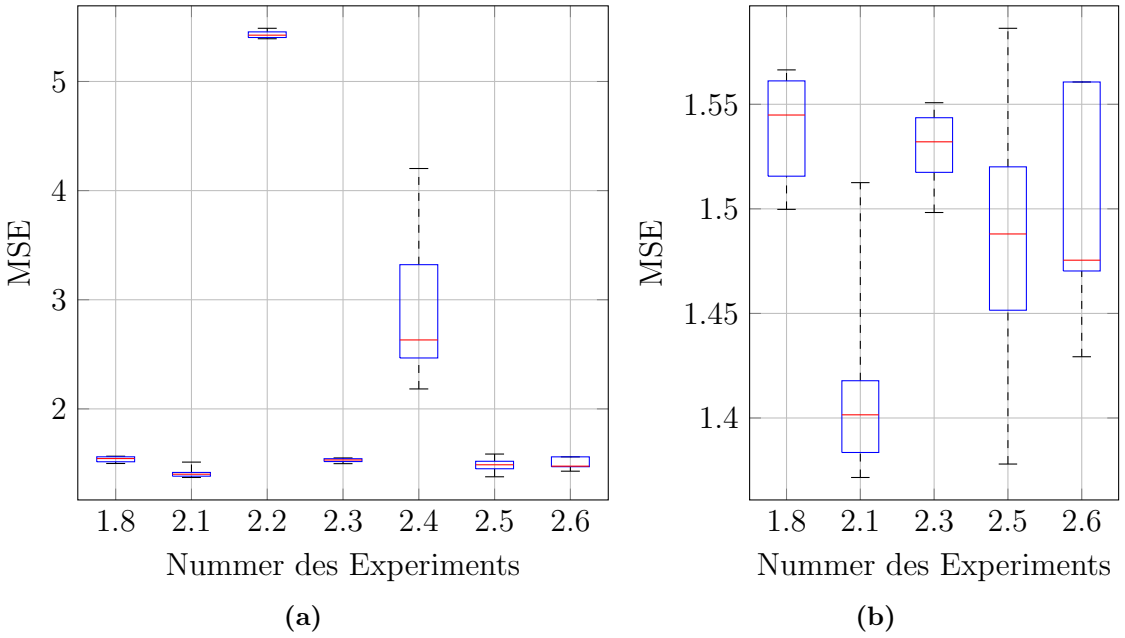


Abbildung 4.17: *Zu sehen sind die Verteilungen der mittleren quadratischen Fehler über die zehn Durchläufe jedes Experiments.*
(a) Vergleich der Verteilung der mittleren quadratischen Fehler für die Experimente 1.8 und 2.1 bis 2.5.
(b) Vergleich der Verteilung der mittleren quadratischen Fehler, ohne die zwei am schlechtesten abschneidenden Experimente.
Rote Linien markieren den Median, blaue Boxen den Bereich zwischen dem 25 %-Quartil und dem 75 %-Quartil und die Markierungen am Ende der gestrichelten Linien den kleinsten und den größten Fehlerwert.

In Abbildung 4.17a ist gut zu sehen, dass die zwei MLP mit der schlechtesten Vorhersage das MLP aus Experiment 2.2 und das MLP aus Experiment 2.4 sind. In beiden Experimente wurden ausschließlich Gelenkwinkelpositionen als Eingangsdaten verwendet. Die Ergebnisse des Experiments 2.4 zeigen, dass die Hinzunahme von Gelenkwinkeln aus vorheriger Zeitpunkten eine Verbesserung der Vorhersage im Vergleich zum Experiment 2.1 bewirkt, während die Streuung des mittleren quadratischen Fehlers stark ansteigt. Den ersten MFCC-Koeffizienten nur auf Basis aufeinanderfol-

gender Gelenkwinkel als Eingangsdaten eines MLP zu verwenden, ist offensichtlich nicht die beste Idee. Abbildung 4.17b zeigt, für eine bessere Vergleichbarkeit der anderen Experimente, die Fehlerverteilungen aller Experimente exklusive Experiment 2.2 und 2.4. Die geringste Varianz weisen die Verteilungen der mittleren quadratischen Fehler zu Experiment 1.8, 2.1 und 2.3 auf. Den kleinsten Median hat die Fehlerverteilung zu Experiment 2.1. Die Mediane der Verteilungen zu Experiment 1.8 und 2.3 sind auch im Vergleich zu denen von Experiment 2.5 und 2.6 signifikant größer.

Man kann aus den Ergebnissen der bisherigen Experimente das Folgende entnehmen:

- Nur die Änderung des Gelenkwinkels Δs als Eingangswert zu verwenden, ist ausreichend, um den ersten MFCC-Koeffizienten vorherzusagen. Es wird dabei auch noch die in Schnitt genaueste Vorhersage berechnet, im Vergleich zu den MLP mit anderen Belegungen der Input-Neuronen.
- Dass die Änderung des Gelenkwinkels Δs als Eingangswert zu verwenden so gut abschneidet weist darauf hin, dass die Vorhersage des ersten MFCC-Koeffizienten nur vom aktuellen Zustand abhängig ist, zumindest wenn die Audio- und Sensordaten zuvor synchronisiert wurden.
- Gelenkwinkel und Gelenkwinkeländerungen vorheriger Zeitpunkte miteinzubeziehen, kann ein Verbesserung der Vorhersage bedeuten. Die große Streuung und Varianz der mittleren quadratischen Fehler zu den Experimenten 2.5 und 2.6 deutet aber auf einen instabilen Lernerfolg solcher MLP hin.
- Den Betrag der Gelenkwinkeländerung $|\Delta s|$ als Belegung für die Input-Neuronen zu verwenden, schneidet schlechter ab, als nur Δs zu verwenden, obwohl der Verlauf von $|\Delta s|$ augenscheinlich dem des ersten MFCC-Koeffizienten ähnlich ist. Dies könnte bedeuten, dass Informationen über die Bewegungsrichtung des Roboterarms sogar schon im ersten MFCC-Koeffizienten enthalten ist.

Das MLP aus Experiment 2.1 weist den geringsten mittleren quadratischen Fehler und eine geringe Varianz dieses Fehlers bei Durchführung mehrerer Lern- und Testläufe auf. Es ist auch eines der MLP, deren Vorhersage des ersten MFCC-Koeffizienten am wenigsten variiert. Somit ist die bisher beste Konfiguration eines MLP, um den ersten MFCC-Koeffizienten vorherzusagen: ein einschichtiges Multi-Layer-Perzeptron mit 100 Hidden-Neuronen und einem Input-Neuron, das mit Δs belegt ist.

4.3 Experimentalreihe 3: Untersuchung der Vorhersage verschiedener Anzahlen von MFCCs

In den Experimenten wurde bisher untersucht, mit welcher Konfiguration der erste MFCC-Koeffizient möglichst gut aus den Sensordaten vorhergesagt werden kann. Der erste MFCC-Koeffizient repräsentiert hauptsächlich das mittlere Niveau der spektralen Koeffizienten des zugehörigen transformierten Spektrums. Die Elektromotoren des Nao erzeugen Geräusche die ein breites Frequenzspektrum überdecken und in unterschiedlichen Frequenzbereichen auch unterschiedlich stark repräsentiert sind. Es ist deshalb notwendig möglichst viele der relevanten MFCC-Koeffizienten vorhersagen zu können, wenn zum Beispiel die Vorhersage zur Unterdrückung bzw. Separation der Motorgeräusche verwendet werden soll. Aus diesem Grund wird in den folgenden Experimenten untersucht, wie gut das bisher beste MLP mehrere MFCC-Koeffizienten vorhersagen kann. Es wurde wieder auf einer Teilaufnahme des Trainingsdatensatzes trainiert und auf dem vollständigen Testdatensatz überprüft. Als Transferfunktion kommt der Tangens-Hyperbolicus zu Einsatz.

Das Training und anschließende Testen wird je Experiment zehn mal wiederholt, um ein Abschätzung über die Stabilität des Lernerfolgs treffen zu können. Es wird wieder je Experiment und Durchlauf ein MLP auf einer ca. 80 Sekunden langen Teilaufnahme des Trainingsdatensatzes über 10000 Epochen mit der Resilient Propagation Methode trainiert und auf dem vollständigen Testdatensatz überprüft. Die beiden Datensätze enthalten zufällige Auf- und Abbewegungen des linken Roboterarms bei einer eingestellten maximalen Bewegungsgeschwindigkeit von 50 % der maximal möglichen Geschwindigkeit. Als Transferfunktion kommt erneut der Tangens-Hyperbolicus zu Einsatz. Die Anzahl der MFCC-Koeffizienten wird sukzessive von zwei auf 26 in unregelmäßig großen Schritten gesteigert.

Bei den Experimenten liegt das Augenmerk auch darauf, ob und wie sich die absoluten Fehler jedes einzelnen MFCC-Koeffizienten verändern, wenn die Anzahl der vorherzusagenden MFCC-Koeffizienten ansteigt, da Distanzen in Räumen unterschiedlicher Dimension nicht direkt miteinander vergleichbar sind. Für alle folgenden Experimente wurde ein MLP mit 100 Hidden-Neuronen auf einer ca. 80 Sekunden langem Teilaufnahme des Trainingsdatensatzes über 10000 Epochen mit der Resilient Propagation Methode trainiert. Jedes trainierte MLP hat nur ein Input-Neuron, das mit der Gelenkwinkeländerung Δs belegt ist.

Experiment 3.1

Das MLP zu Experiment 3.1 erweitert die Vorhersage von einem auf zwei MFCC-Koeffizienten, den ersten und den zweiten MFCC-Koeffizienten. Abbildung 4.18 zeigt den ersten und zweiten MFCC-Koeffizienten und deren Vorhersage auf dem gleichen Teilabschnitt des Testdatensatzes, der auch in allen vorherigen Experimenten gezeigt wurde.

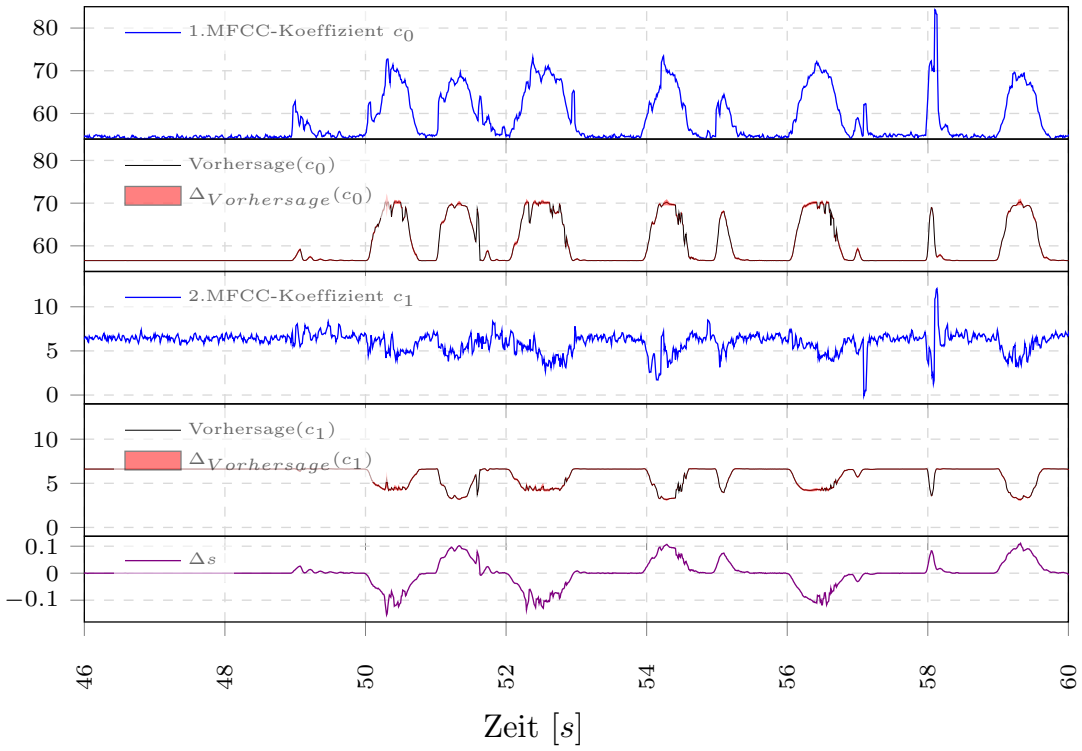


Abbildung 4.18: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.1.*
Im Vergleich zum Lernergebnis von Experiment 2.1 (siehe Abbildung 4.11) ist keine wesentliche Verschlechterung auszumachen, wenn ein weiterer MFCC-Koeffizient vorhergesagt werden soll.
Die oberen vier Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 0, ..., 1$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Vergleich man das Lernergebnis der MLP dieses Experiments mit den Ergebnissen aus Experiment 2.1 (vgl. Abbildung 4.11), ist keine wesentliche Verschlechterung auszumachen, wenn die ersten zwei MFCC-Koeffizienten vorhergesagt werden.

Experiment 3.2

Die MLPs in Experiment 3.2 berechnen die Vorhersage der ersten vier MFCC-Koeffizienten auf Basis der Gelenkwinkeländerung Δs.

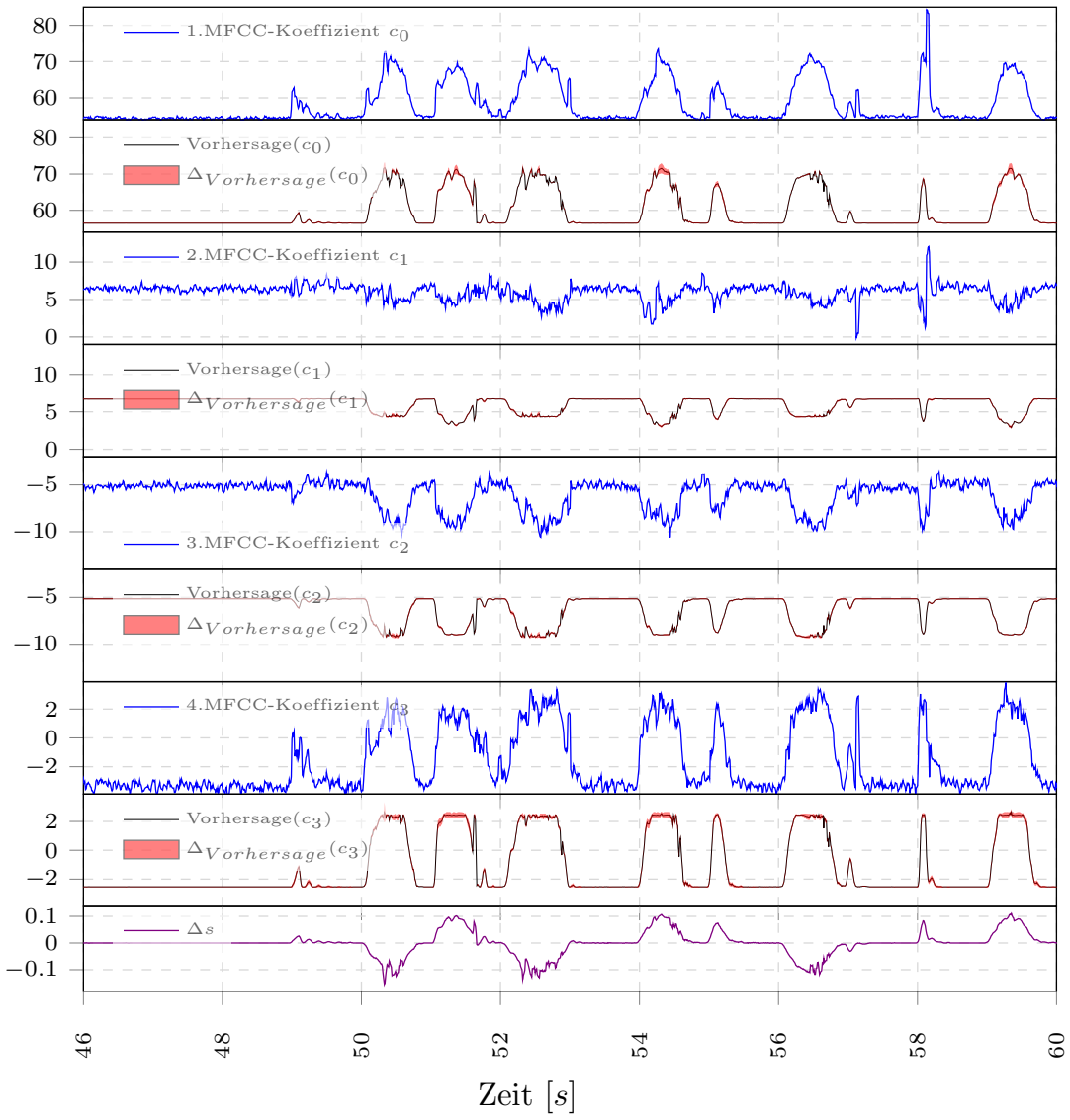


Abbildung 4.19: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.2.*
Man sieht die stellenweise geringfügig stärker variierende Vorhersage des ersten MFCC-Koeffizienten im Vergleich zu Experiment 3.1.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 0, ..., 3$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Stellenweise ist eine etwas stärker variierende Vorhersage des ersten MFCC-Koeffizienten im Vergleich zu Experiment 3.1 zu sehen.

Experiment 3.3

Für Experiment 3.3 werden die ersten sechs MFCC-Koeffizienten von den MLPs vorhergesagt.

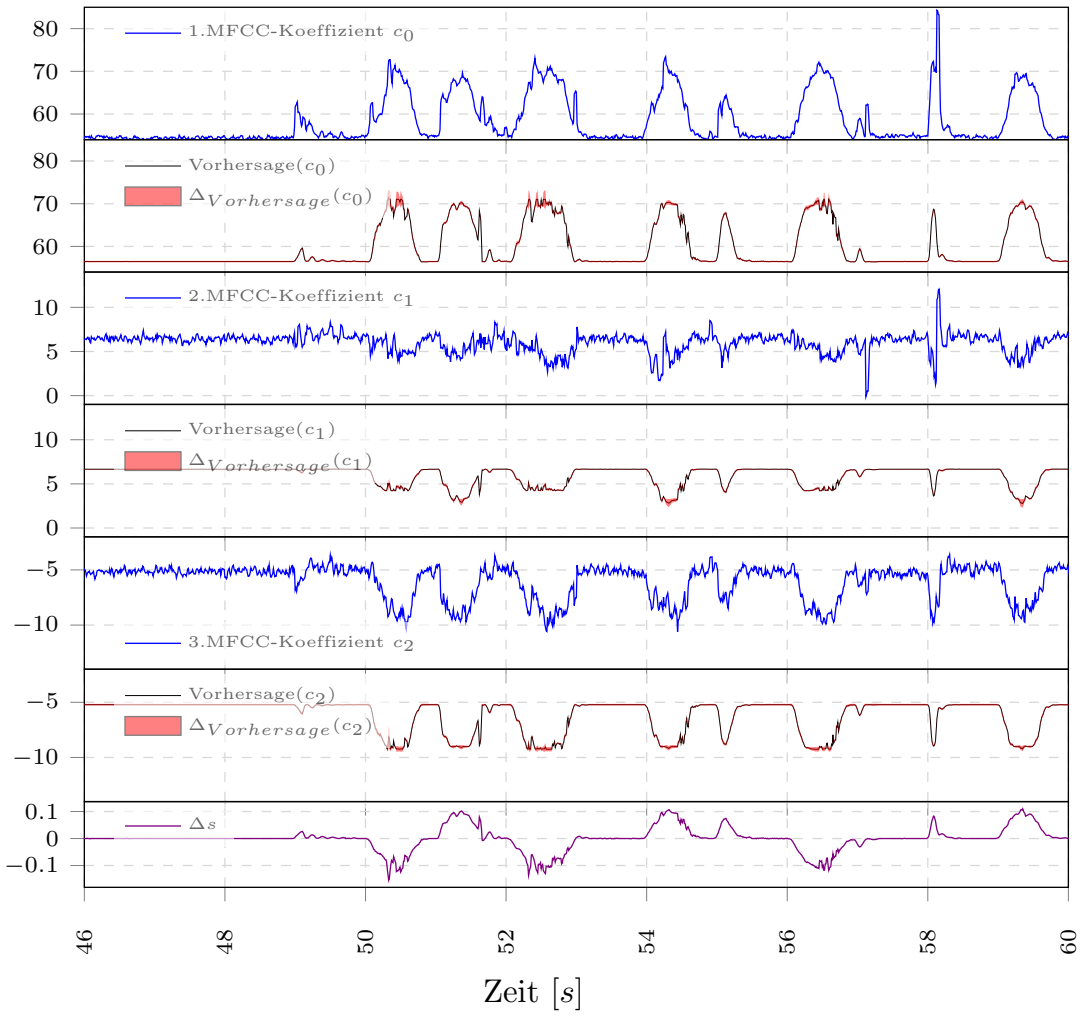


Abbildung 4.20: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.3 für die ersten drei MFCC-Koeffizienten.*
Die Vorhersage variiert stellenweise geringfügig stärker, im Vergleich zu Experiment 3.2. Es fällt auch auf, dass die Vorhersage ab dem zweiten MFCC-Koeffizienten gegenüber dem vorherzusagenden MFCC-Koeffizienten viel glatter ist.
Die oberen sechs Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 0, ..., 2$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Bei Betrachtung der Abbildungen 4.18 bis 4.21 kann eine geringfügig größer werdende Varianz der Vorhersage des ersten MFCC-Koeffizienten beobachtet werden, wenn die Anzahl der vorherzusagenden Koeffizienten zunimmt. Man sieht auch, dass ab dem zweiten MFCC-Koeffizienten der Verlauf der Vorhersagen viel glatter ist, im Vergleich

zum vorherzusagenden MFCC-Koeffizienten, als das beim ersten MFCC-Koeffizienten der Fall ist. Dies deutet darauf hin, dass die höheren Koeffizienten stärker verrauscht sind und die Vorhersagen der MLPs das Rauschen der höheren Koeffizienten nicht abbilden können.

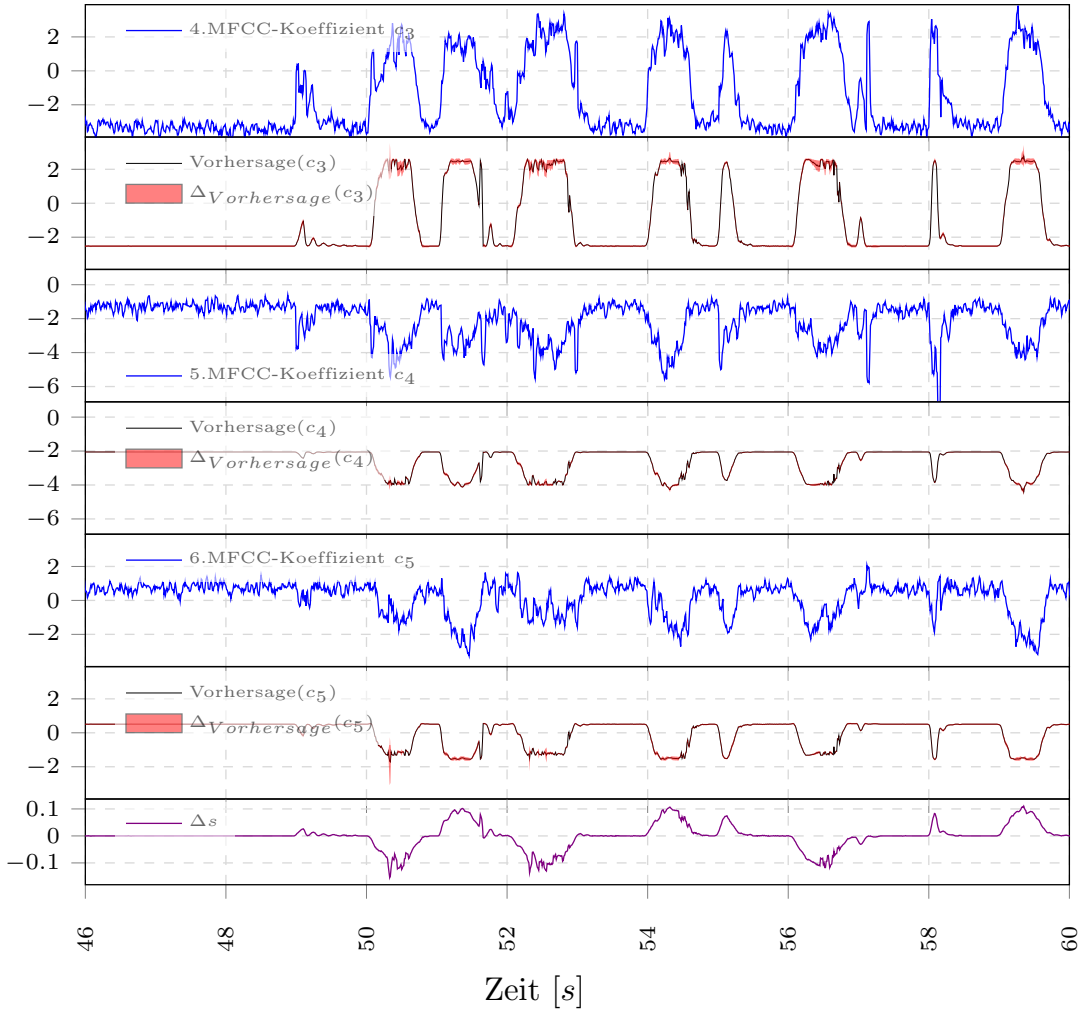


Abbildung 4.21: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.3 für den vierten bis sechsten MFCC-Koeffizienten.*
Man sieht auch hier, wie in Abbildung 4.20, den recht glatten Verlauf der Vorhersage im Vergleich zu dem vorherzusagenden MFCC-Koeffizienten. Es fällt auch ein recht ähnliches Aussehen des Verlaufes der Vorhersagen der zweiten, dritten, fünften und sechsten MFCC-Koeffizienten auf.
Die oberen sechs Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 3, ..., 5$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Experiment 3.4

Die Anzahl der vorherzusagenden MFCC-Koeffizienten des MLP steigt in Experiment 3.4 auf dreizehn.

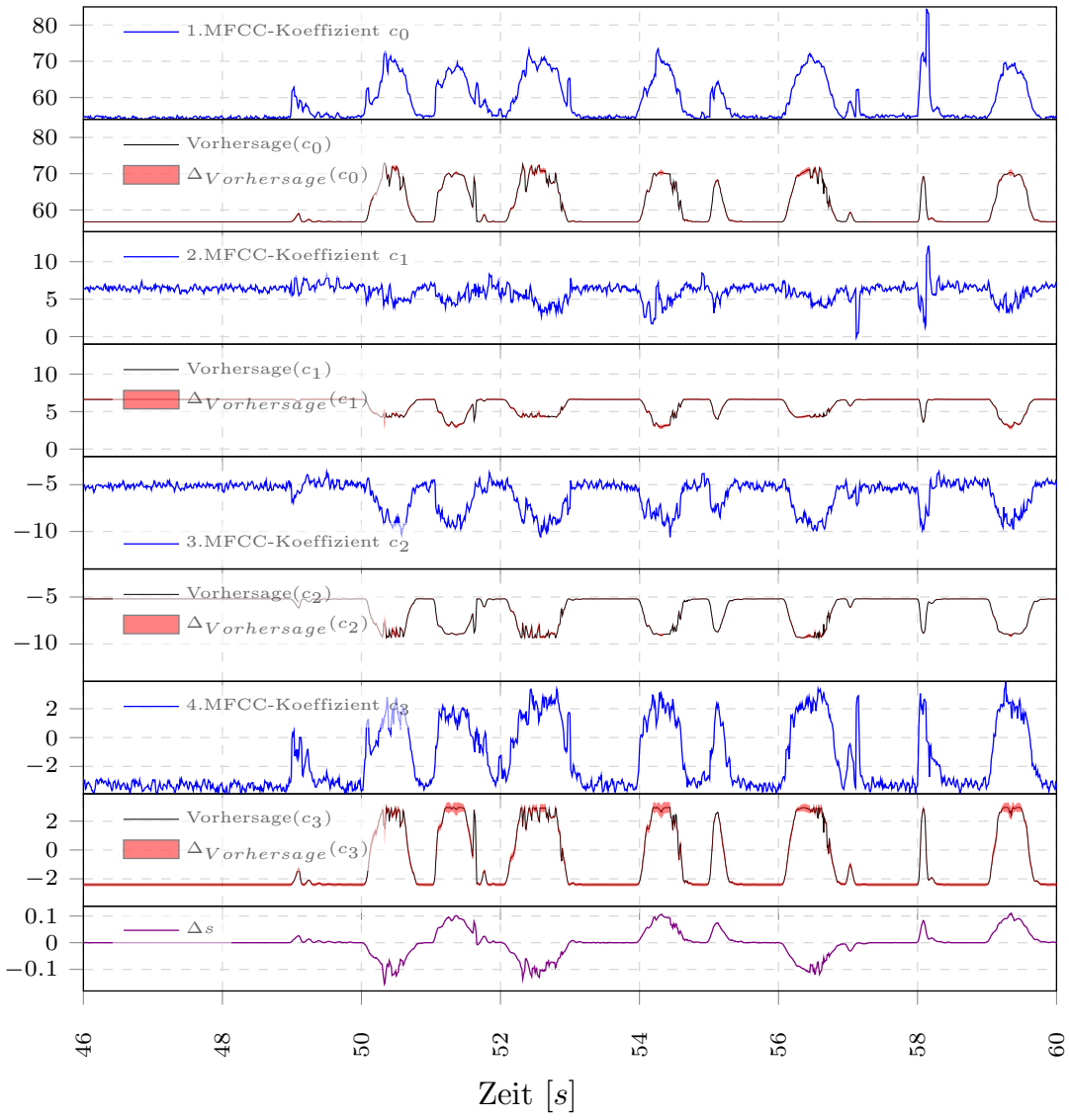


Abbildung 4.22: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.4 für die ersten vier MFCC-Koeffizienten.*
Wie man sieht, sind die Vorhersagen der ersten vier MFCC-Koeffizienten kaum von denen aus Experiment 3.3 zu unterscheiden.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k $(k = 0, ..., 3)$, direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Vergleicht man die Abbildungen 4.20 bis 4.24, dann sieht man, dass sich die Vorhersagen der ersten sechs MFCC-Koeffizienten kaum von denen aus Experiment 3.3 un-

terscheiden. Insbesondere fällt der recht ähnliche Verlauf der Vorhersage der MFCC-Koeffizienten zwei, drei, fünf, sechs, sieben und neun auf.

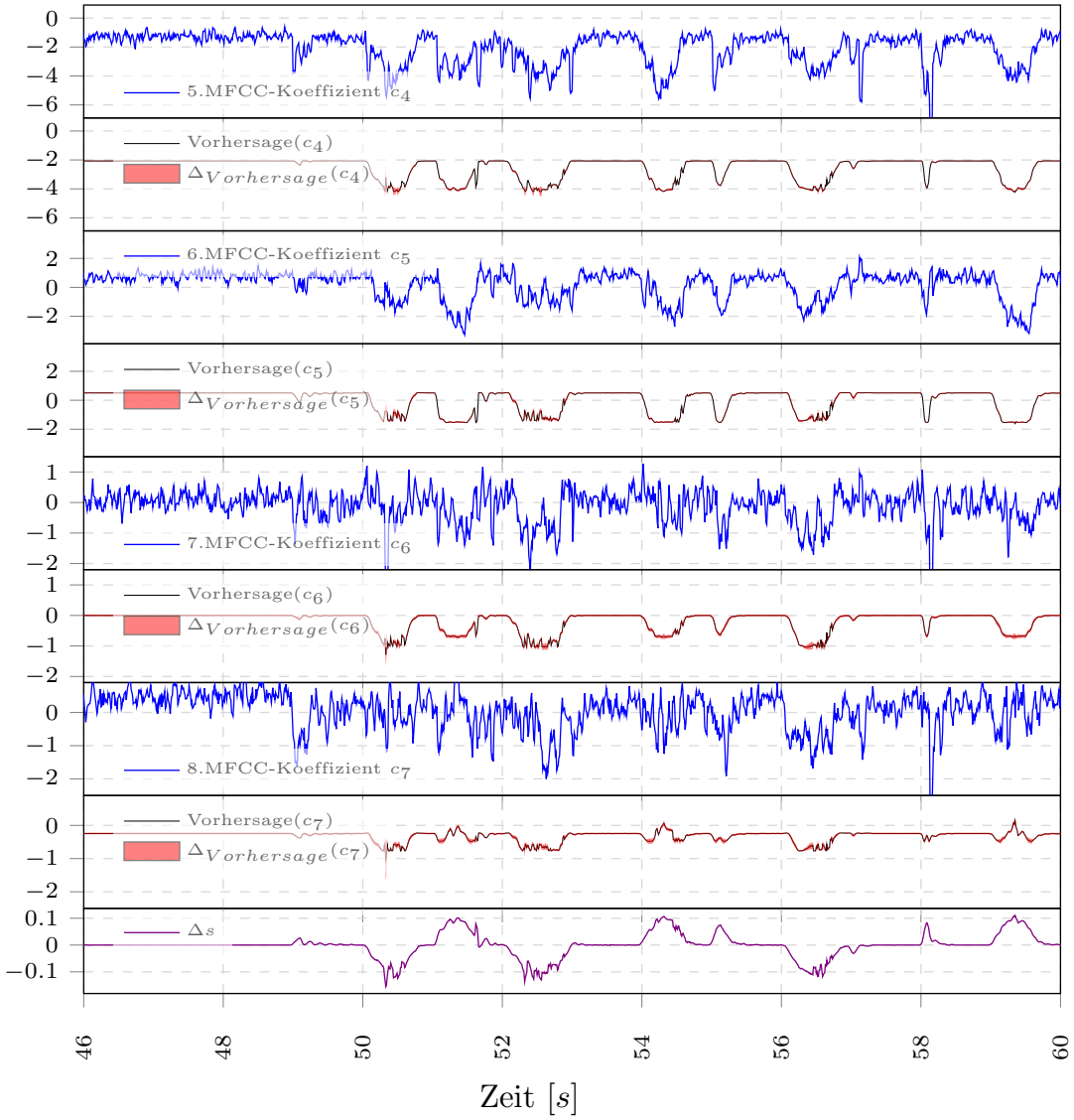


Abbildung 4.23: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.4 für den fünften bis achten MFCC-Koeffizienten.*
Insbesondere fällt der recht ähnliche Verlauf der Vorhersage der MFCC-Koeffizienten fünf, sechs und sieben auf.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 4, ..., 7$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

An den Vorhersage der höheren MFCC-Koeffizienten, wie dem zehnten, elften und zwölften MFCC-Koeffizienten, ist besonders deutlich zu sehen, dass die MLPs den

Verlauf der Koeffizienten und das Rauschen in den höheren MFCC-Koeffizienten nur schlecht abbilden können.

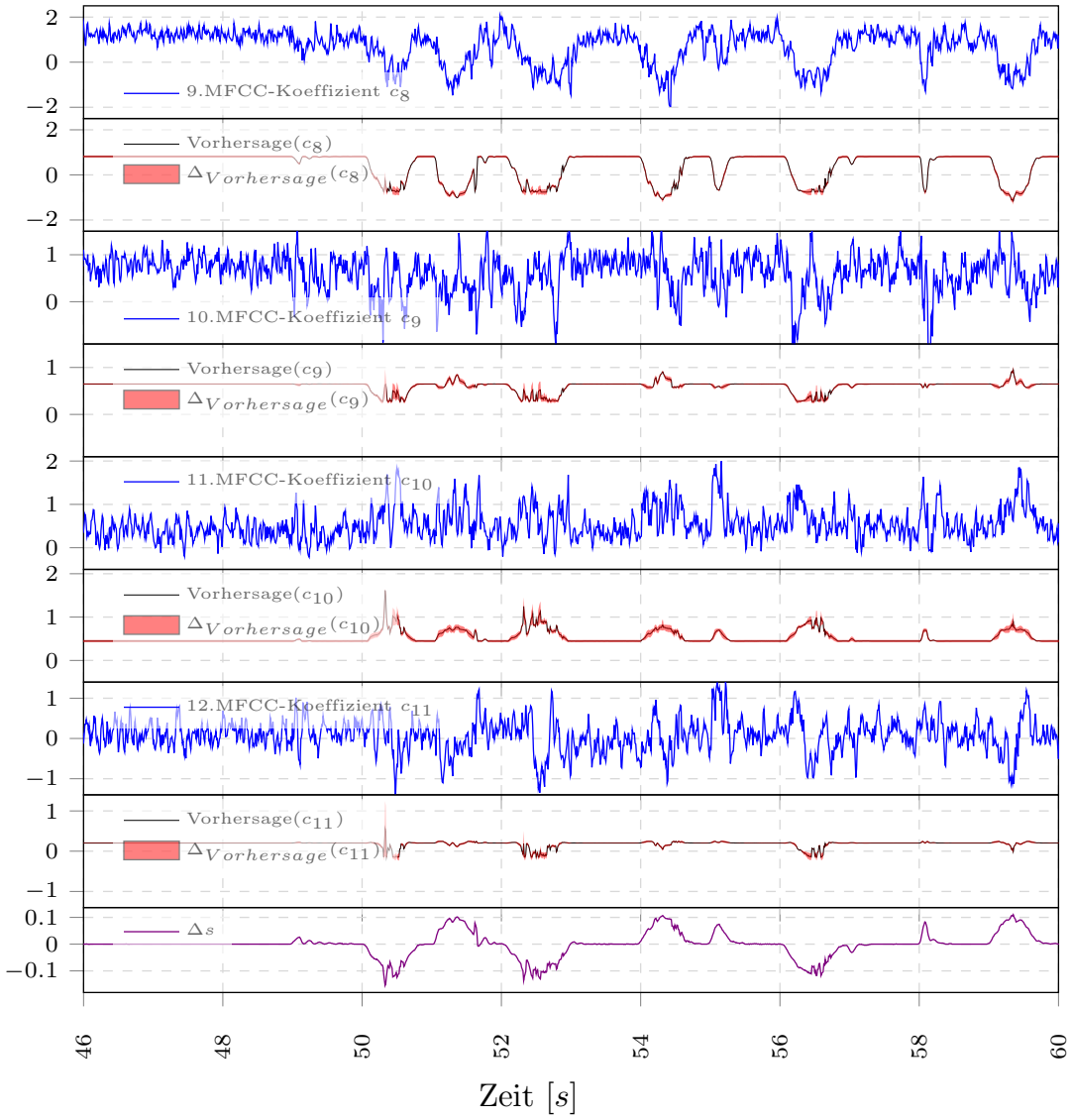


Abbildung 4.24: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.4 für den neunten bis zwölften MFCC-Koeffizienten.*
Bei der Vorhersage des zehnten, elften und zwölften MFCC-Koeffizienten ist besonders deutlich zu sehen, dass die MLPs den Verlauf der Koeffizienten und das Rauschen in den höheren MFCC-Koeffizienten nur schlecht abbilden können.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 8, ..., 11$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Experiment 3.5

In Experiment 3.5 werden die zwanzig ersten MFCC-Koeffizienten von dem MLP vorhergesagt.

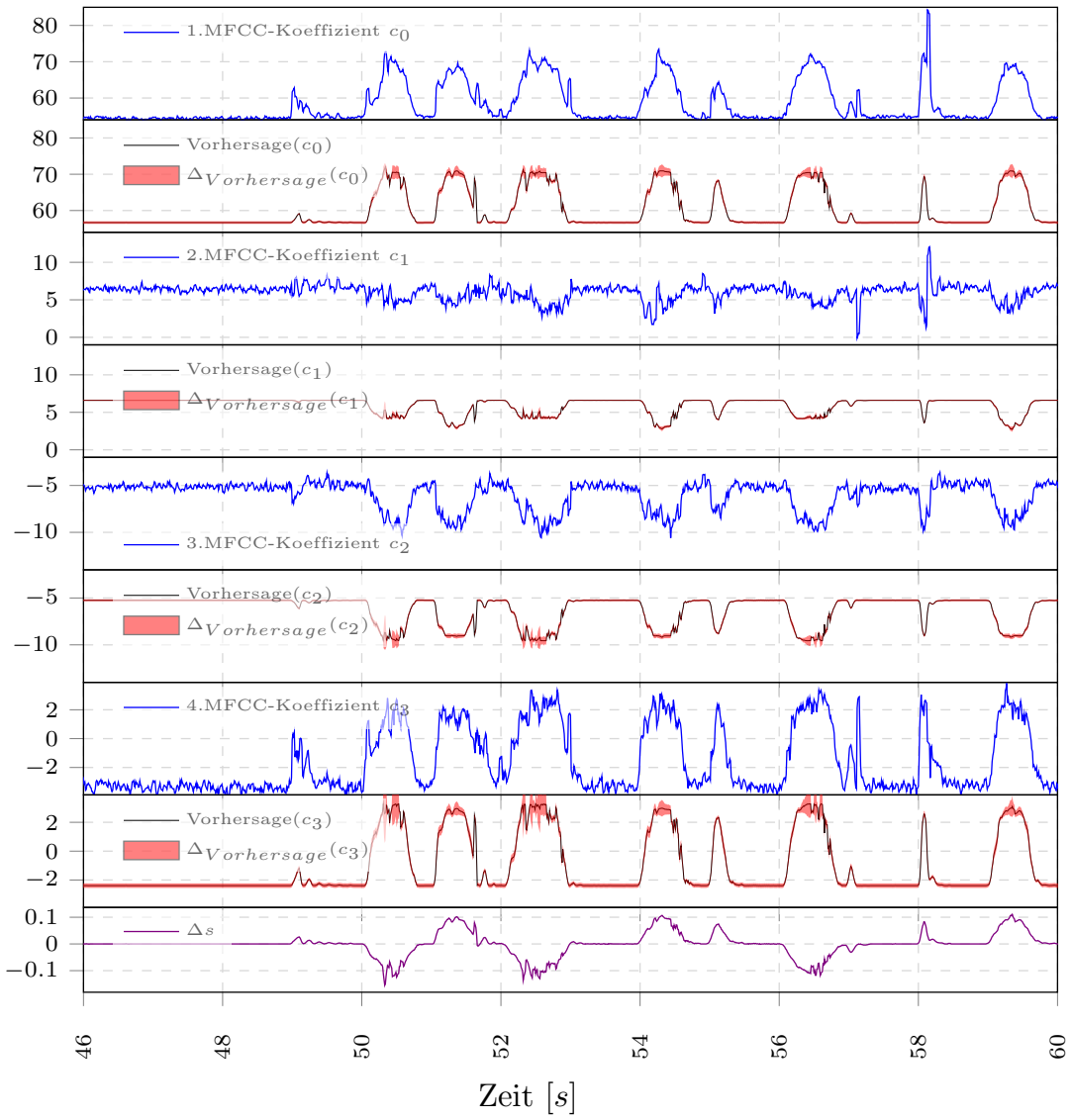


Abbildung 4.25: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.5 für die ersten vier MFCC-Koeffizienten.*
Bis auf eine leicht stärkeres Variieren der Vorhersagen des ersten, dritten und vierten MFCC-Koeffizienten, sind kaum Änderungen am Vorhersageverhalten im Vergleich zu den Experimenten 3.2 bis 3.4 zu erkennen.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 0, ..., 3$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Bis auf dass die Vorhersagen des ersten, dritten und vierten MFCC-Koeffizienten

etwas stärker variieren, sind kaum Unterschiede im Vergleich zu den Experimenten 3.2 bis 3.4 (vgl. Abbildung 4.19 bis 4.22) zu erkennen.

Auch für die Vorhersagen des fünften bis achten MFCC-Koeffizienten sind beim Vergleich der Abbildungen 4.21 bis 4.26 nur marginale Unterschiede erkennbar.

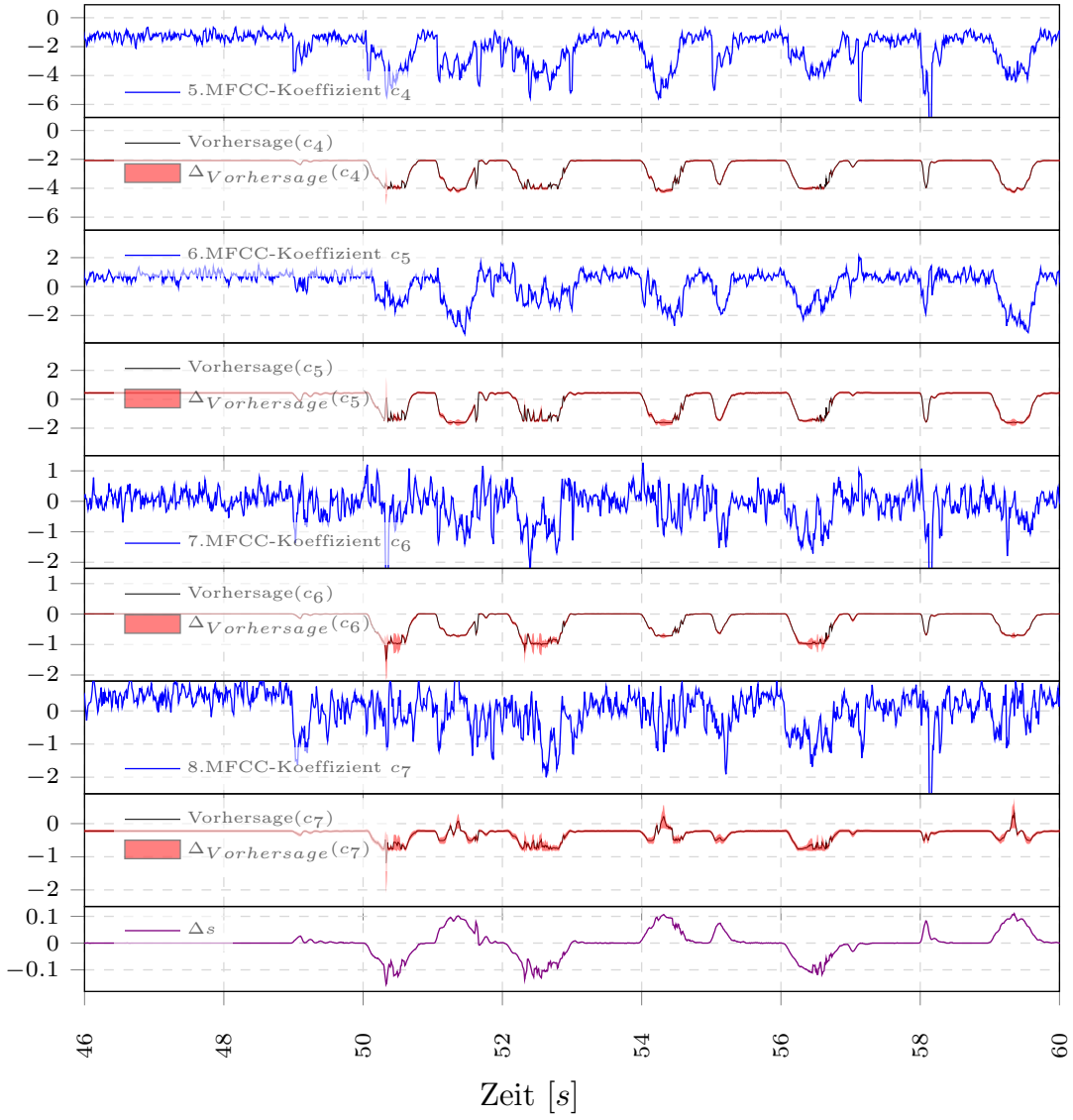


Abbildung 4.26: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.5 für den fünften bis achten MFCC-Koeffizienten.*
Man sieht, dass die Vorhersagen der MFCC-Koeffizienten fünf bis acht ähnlich denen von Experiment 3.4 sind.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 4, ..., 7$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Gleiches gilt für den neunten bis zwölften MFCC-Koeffizienten, wenn man die Abbil-

dungen 4.24 und 4.27 miteinander vergleicht. Eine steigende Anzahl von vorherzusagenden MFCC-Koeffizienten scheint nur wenig Einfluss auf die Vorhersagequalität zu haben.

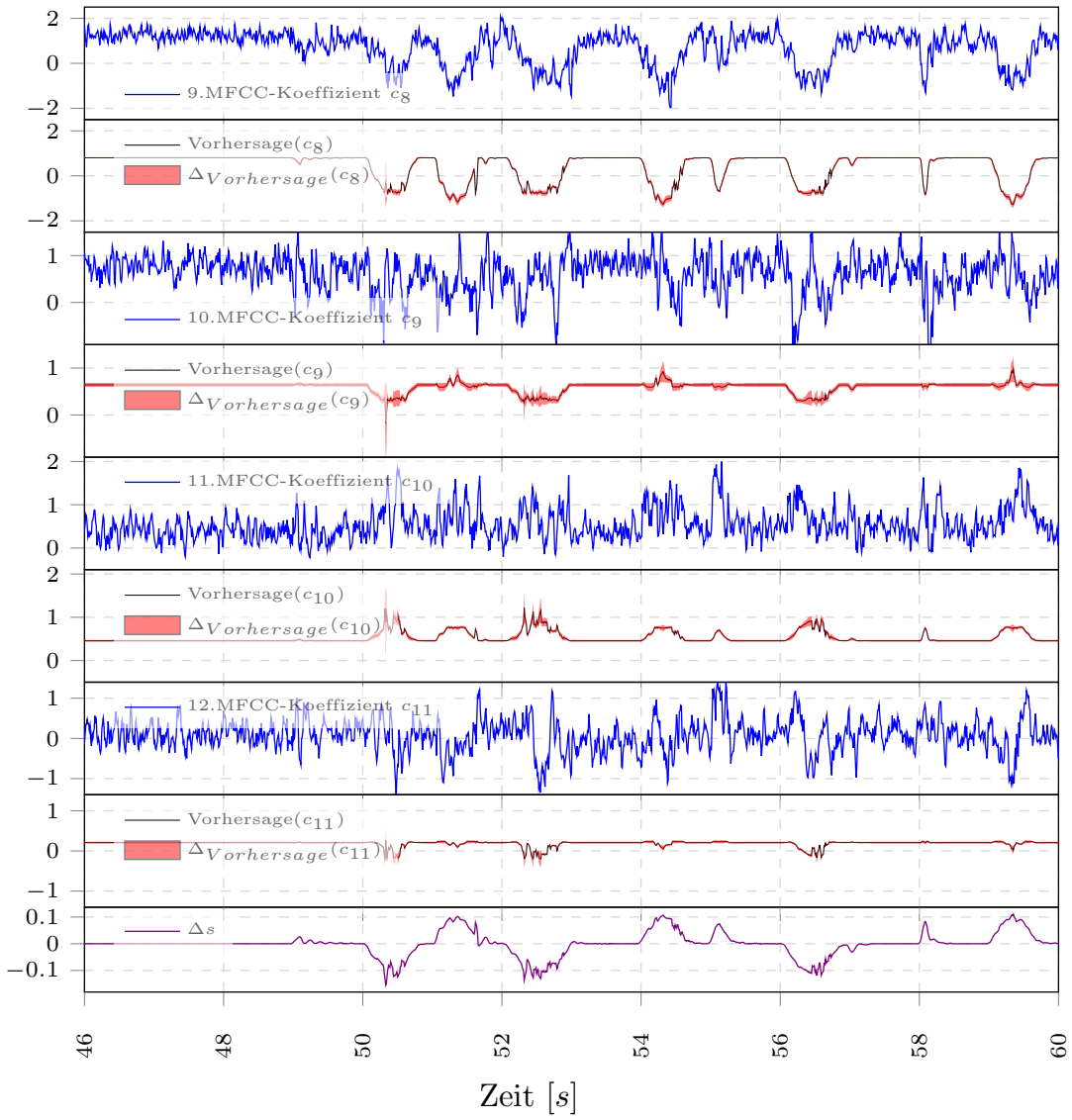


Abbildung 4.27: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.5 für den neunten bis zwölften MFCC-Koeffizienten.*
Es sind kaum Unterschiede zur den Vorhersagen der MFCC-Koeffizienten neun bis zwölf im Vergleich zu denen aus Experiment 3.4 zu sehen.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 8, ..., 11$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Experiment 3.6

Im letzten Experiment der dritten Experimentalreihe wird die Vorhersage von sechsundzwanzig MFCC-Koeffizienten berechnet.

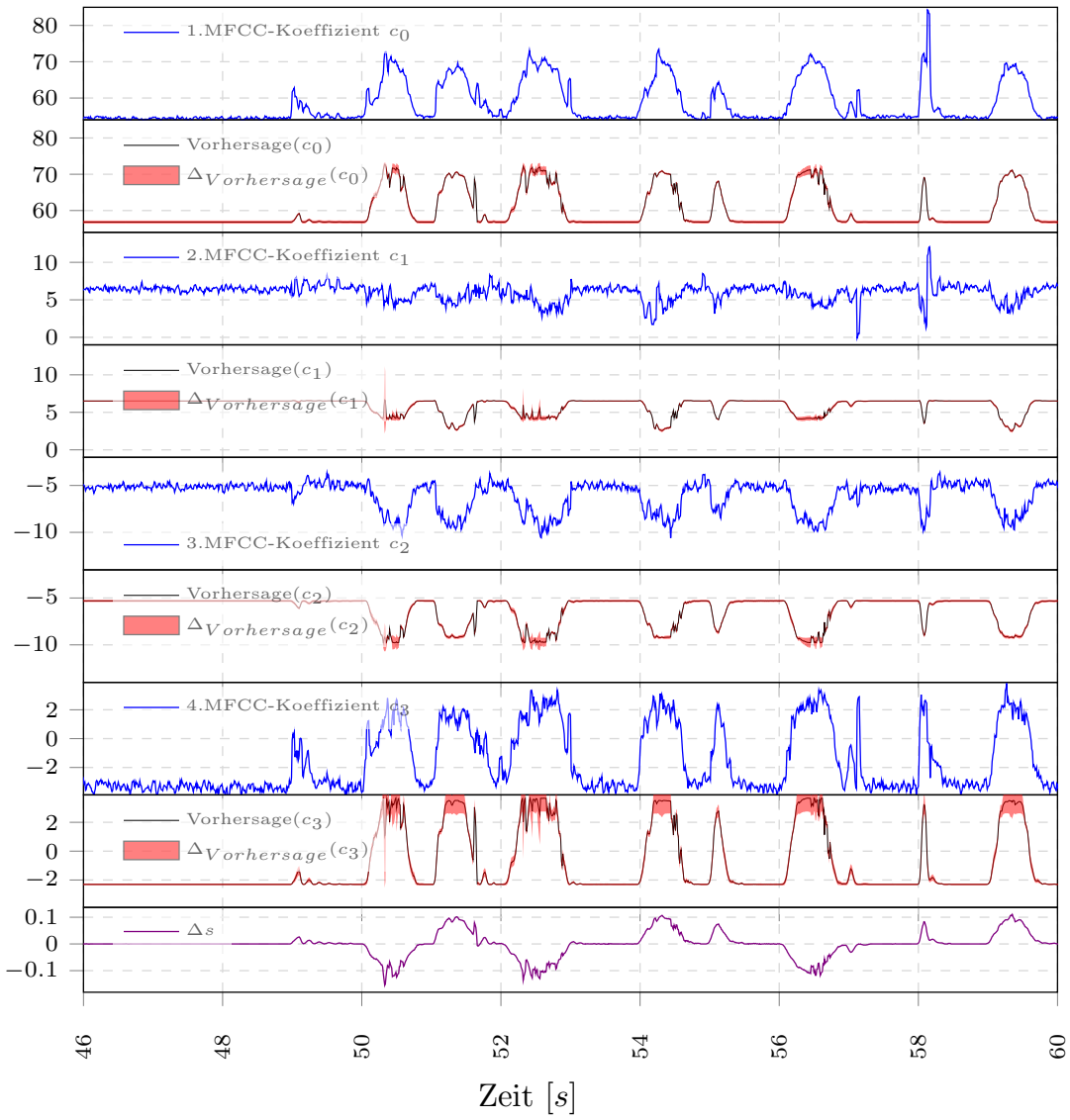


Abbildung 4.28: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.6 für die ersten vier MFCC-Koeffizienten.*
Man kann vor allem an der Vorhersage des vierten MFCC-Koeffizienten sehen, dass die Vorhersagen einzelner Koeffizienten instabiler werden, wenn 26 MFCC-Koeffizienten von den MLPs approximiert werden.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k $(k = 0, ..., 3)$, direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Vor allem an der Vorhersage des vierten MFCC-Koeffizienten in Abbildung 4.28 ist

zu sehen, dass die Vorhersagen einzelner Koeffizienten instabiler werden, wenn 26 MFCC-Koeffizienten von den MLPs approximiert werden.

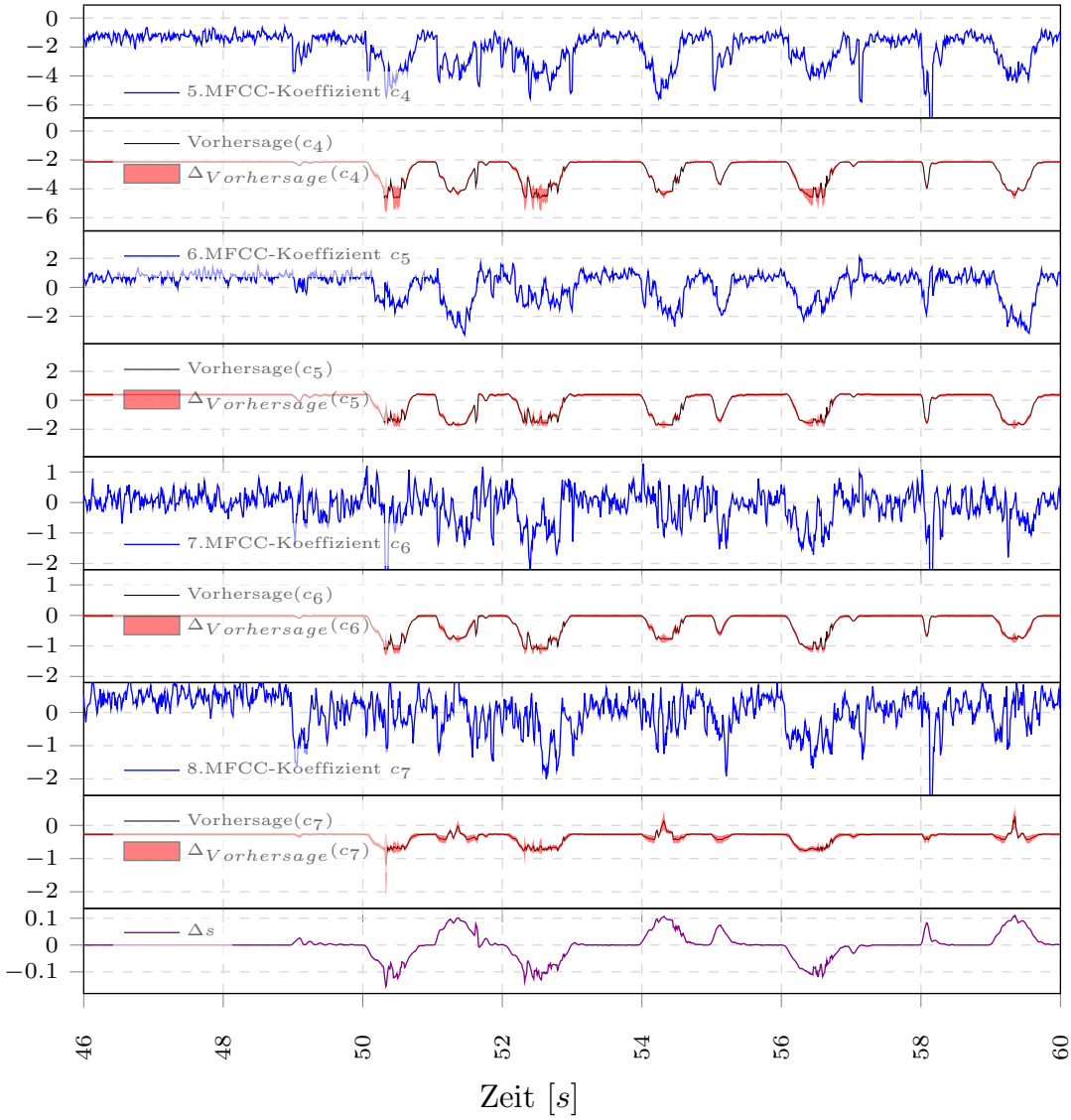


Abbildung 4.29: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.6 für den fünften bis achten MFCC-Koeffizienten.*
Die Vorhersagen des fünften bis achten MFCC-Koeffizienten gelingen ähnlich gut wie in den Experimenten 3.4 und 3.5, der Lernerfolg ist teilweise aber geringfügig instabiler.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 4, ..., 7$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Obwohl 26 MFCC-Koeffizienten vorhergesagt werden, gelingt die Vorhersage der höheren Koeffizienten, ab dem Fünften, ähnlich gut wie bei der Vorhersage von weni-

ger MFCC-Koeffizienten, der Lernerfolg ist zum Teil aber geringfügig instabiler (siehe Abbildung 4.29 und 4.30).

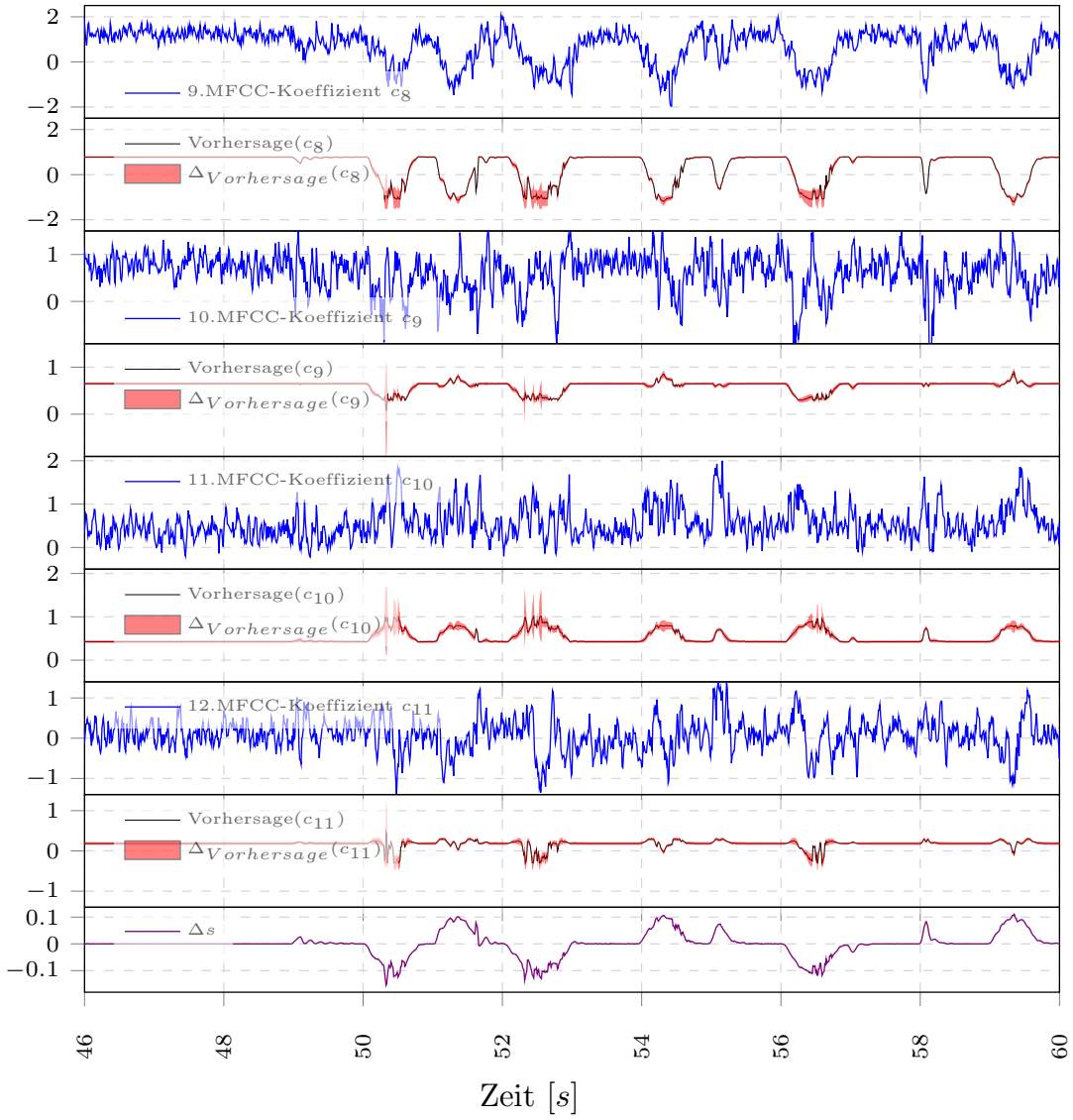


Abbildung 4.30: *Zu sehen ist der Lernerfolg der MLPs zu Experiment 3.6 für den neunten bis zwölften MFCC-Koeffizienten.*
Obwohl 26 MFCC-Koeffizienten vorhergesagt werden, gelingt die Vorhersage der höheren Koeffizienten ähnlich gut wie bei der Vorhersage von weniger MFCC-Koeffizienten.
Die oberen acht Grafiken zeigen, von oben nach unten, jeweils den zu approximierenden MFCC-Koeffizienten c_k ($k = 8, ..., 11$), direkt darunter dessen Vorhersage (rot = Bereich zwischen Minimum und Maximum, schwarz = Mittelwert). Zum Vergleich ist in der untersten Grafik die Änderung des Gelenkwinkels Δs abgebildet.

Auswertung der Experimentalreihe 3

Es wurden für jedes Experiment der quadratische Fehler und die absoluten Fehler je MFCC-Koeffizient über den gesamten Testdatensatz gemittelt und die statistischen Kennzahlen, wie Median und Standardabweichung, über diese Mittelwerte der zehn Experimente berechnet.

In den Abbildungen 4.18 bis 4.28 ist zu sehen, dass die ersten zwei bzw. vier MFCC-Koeffizienten von allen MLPs annähernd vorhergesagt werden können. In einigen der Grafiken, wie in Abbildung 4.28, ist am Beispiel des vierten MFCC-Koeffizienten zwischen Sekunde 51 und 52 zu sehen, dass es vereinzelt zu großen Fehlern in der Vorhersage der MFCC-Koeffizienten kommen kann. Den Abbildungen kann auch entnommen werden, dass sich die Vorhersagen der ersten drei MFCC-Koeffizienten, mit jeder Steigerung der Anzahl vorherzusagender Koeffizienten, nur leicht verändern. Abbildung 4.32 zeigt die Verteilung des gemittelten quadratischen Fehler je Experiment nach zehn Durchläufen. Mit steigender Anzahl der vorherzusagenden MFCC-Koeffizienten, steigt der mittlere quadratische Fehler nahezu linear.

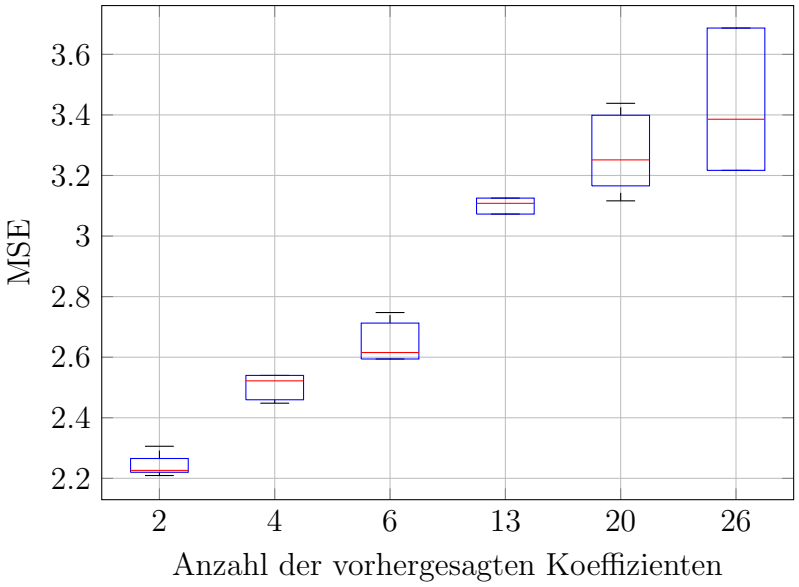
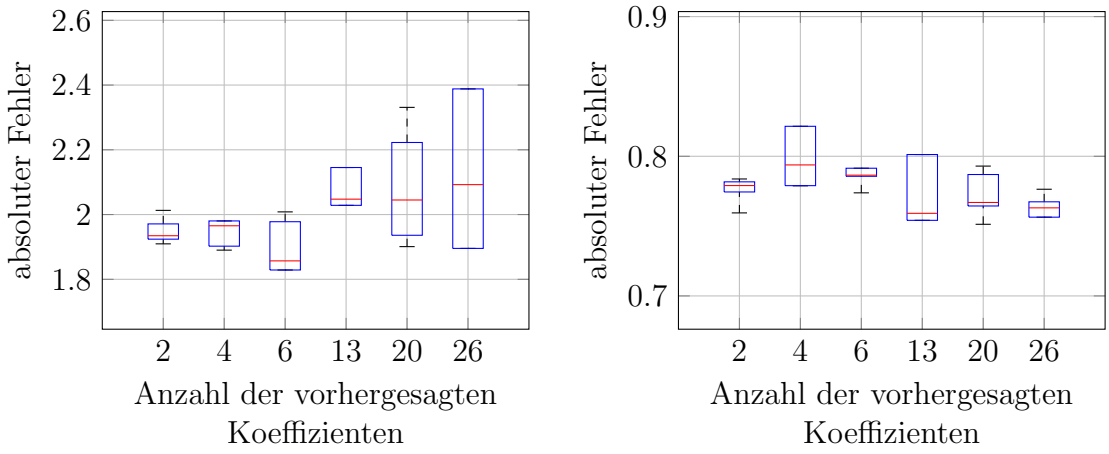


Abbildung 4.31: *Dargestellt ist die Verteilung der mittleren quadratischen Fehler aus zehn Durchläufen je Experiment, in Abhängigkeit von der Anzahl der vorhergesagten MFCC-Koeffizienten. Nicht vorhergesagte MFCC-Koeffizienten wurden bei der Berechnung der mittleren quadratischen Fehler (MSE) nicht berücksichtigt.*
Rote Linien markieren den Median, blaue Boxen den Bereich zwischen dem 25 %-Quartil und dem 75 %-Quartil und die Markierungen am Ende der gestrichelten Linien den kleinsten und den größten mittleren Fehler.

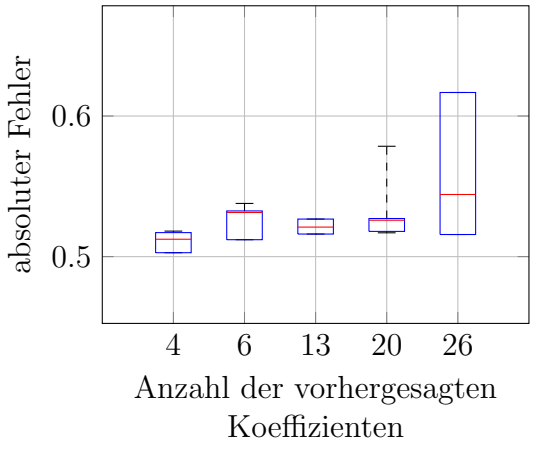
Ebenso werden Streuung und Abweichung größer, je mehr MFCC-Koeffizienten vorhergesagt werden sollen. Da der mittlere quadratische Fehler dem gemittelten Abstand der Vorhersage in einem mehrdimensionalen Raum entspricht, sagt dieser über die erreichte mittlere Genauigkeit der Vorhersage eines einzelnen MFCC-Koeffizienten nicht sehr viel aus. Die Aussage des mittleren quadratischen Fehler bezieht sich auf die

mittlere Güte der gesamten Vorhersage aller MFCC-Koeffizienten. Ein Vergleich der mittleren absoluten Fehler je MFCC-Koeffizient ermöglicht ein bessere Abschätzung, wie genau die Vorhersage des einzelnen MFCC-Koeffizienten ist. Die folgenden Abbildungen 4.32 bis 4.34 illustrieren die Änderungen des mittleren absoluten Fehlers der Vorhersage eines bestimmten MFCC-Koeffizienten, während die Anzahl der MFCC-Koeffizienten ansteigt.

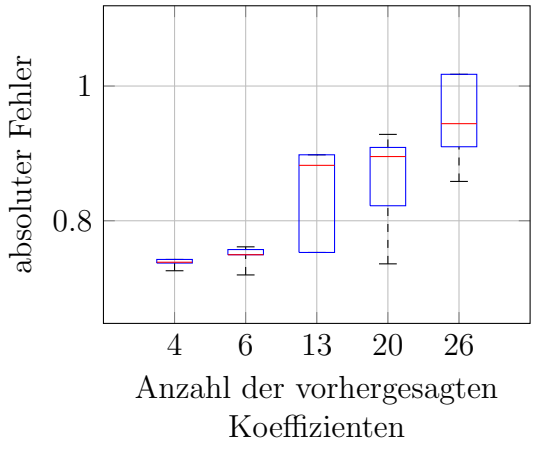


(a) *1. MFCC-Koeffizient*

(b) *2. MFCC-Koeffizient*



(c) *3. MFCC-Koeffizient*

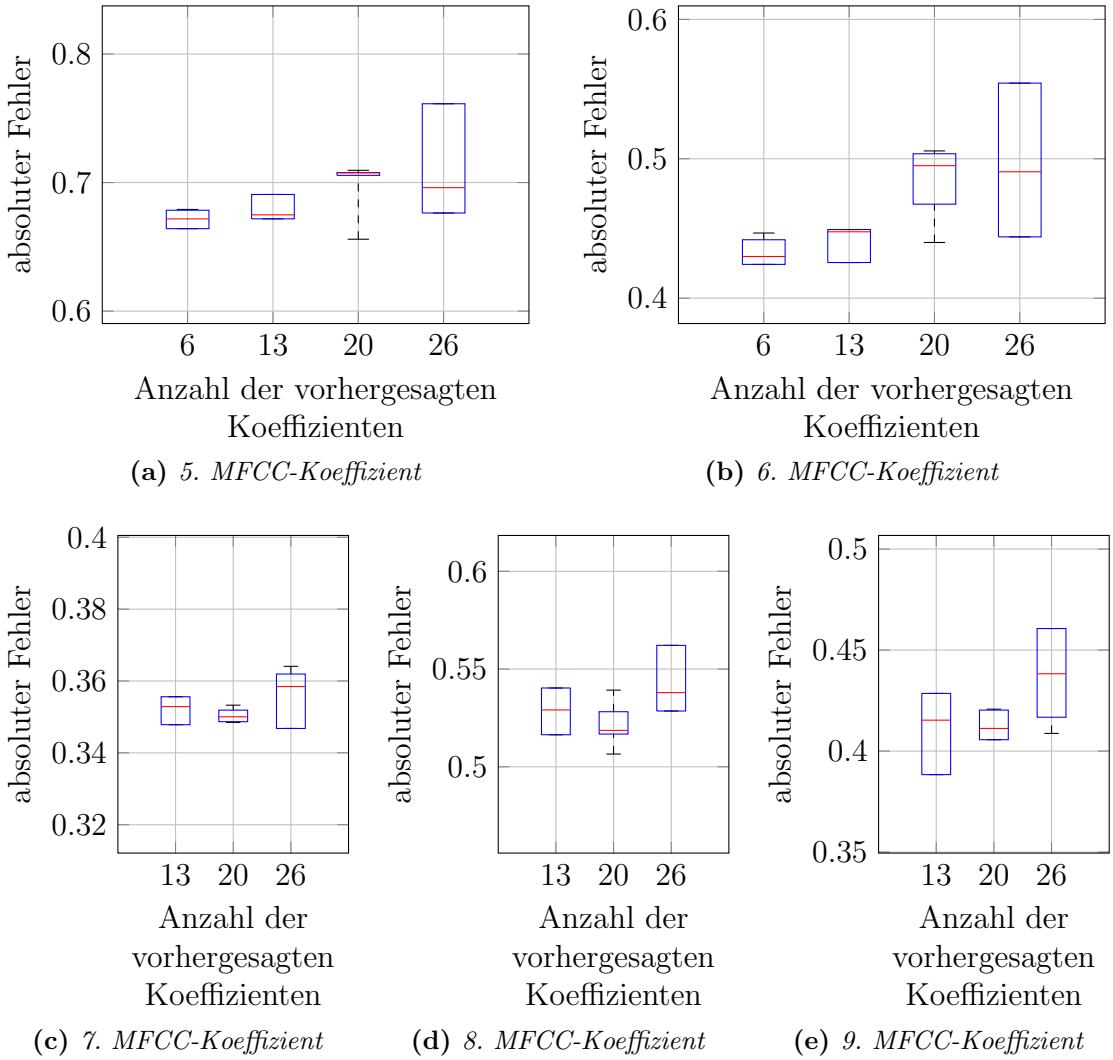


(d) *4. MFCC-Koeffizient*

Abbildung 4.32: *Dargestellt ist die Verteilung der mittleren absoluten Fehler des ersten (a), zweiten (b), dritten (c) und vierten MFCC-Koeffizienten (d) aus zehn Durchläufen je Experiment, in Abhängigkeit von der Anzahl der vorhergesagten MFCC-Koeffizienten. Rote Linien markieren den Median, blaue Boxen den Bereich zwischen dem 25 %-Quartil und dem 75 %-Quartil und die Markierungen am Ende der gestrichelten Linien den kleinsten und den größten mittleren Fehler.*

Wie man sehen kann, steigt der Fehler der Approximation des ersten MFCC-Koeffizienten bei Vorhersage von insgesamt dreizehn MFCC-Koeffizienten leicht an, während sich die Streuung des Fehlers bei der Vorhersage von insgesamt 20 und 26 MFCC-Koeffizienten zusätzlich mehr als verdoppelt. Der zweite MFCC-Koeffizient zeigt ein anderes Verhalten bei steigender Anzahl der vorherzusagenden MFCC-Koeffizienten.

Für den zweiten MFCC-Koeffizienten ergibt sich eine leichte Verbesserung der Vorhersage und zudem eine etwas geringere Abweichung, je mehr MFCC-Koeffizienten vorhergesagt werden. Der Fehler vom MFCC-Koeffizient drei steigt nur sehr wenig an, dessen Abweichung wird bei einer Anzahl von 26 vorherzusagenden MFCC-Koeffizienten aber signifikant schlechter. Der Fehler des vierten MFCC-Koeffizienten steigt schon bei dreizehn vorherzusagenden MFCC-Koeffizient stärker an, während sich die Abweichung sichtbar vergrößert.



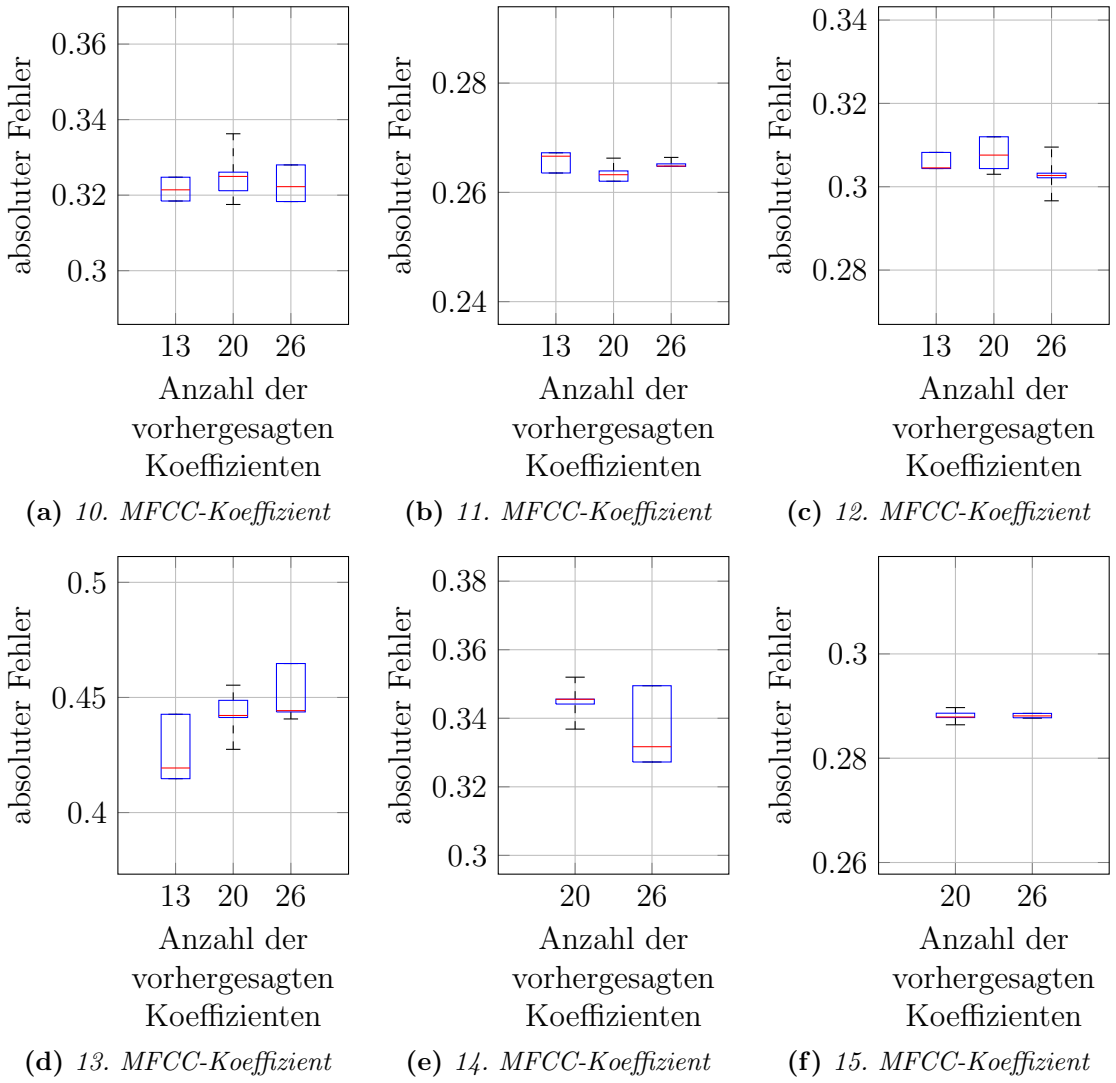
(a) *5. MFCC-Koeffizient* **(b)** *6. MFCC-Koeffizient*

(c) *7. MFCC-Koeffizient* **(d)** *8. MFCC-Koeffizient* **(e)** *9. MFCC-Koeffizient*

Abbildung 4.33: *Dargestellt ist die Verteilung der mittleren absoluten Fehler des fünften (a), sechsten (b), siebten (c), achten (c) und neunten MFCC-Koeffizienten (e) aus zehn Durchläufen je Experiment, in Abhängigkeit von der Anzahl der vorhergesagten MFCC-Koeffizienten.*
Rote Linien markieren den Median, blaue Boxen den Bereich zwischen dem 25 %-Quartil und dem 75 %-Quartil und die Markierungen am Ende der gestrichelten Linien den kleinsten und den größten mittleren Fehler.

Auch Koeffizient fünf und sechs zeigen einen leichten Anstieg des Fehlers bei wachsender Anzahl vorherzusagender MFCC-Koeffizienten mit größer werdender Streuung

oder Varianz. Bei den Koeffizienten sieben, acht und neun wiederum sind, bei Vorhersage von zwanzig MFCC-Koeffizienten, der absolute Fehler und die Varianz am kleinsten, während die Vorhersage von dreizehn und sechsundzwanzig MFCC-Koeffizienten schlechter abschneidet. Bei dreizehnten MFCC-Koeffizienten zeigt sich ein ähnliches Bild. Mit Ausnahme des dreizehnten und vierzehnten MFCC-Koeffizienten ändert sich der Fehler ab dem zehnten MFCC-Koeffizienten kaum, wenn die Anzahl der vorhergesagten MFCC-Koeffizienten ansteigt.



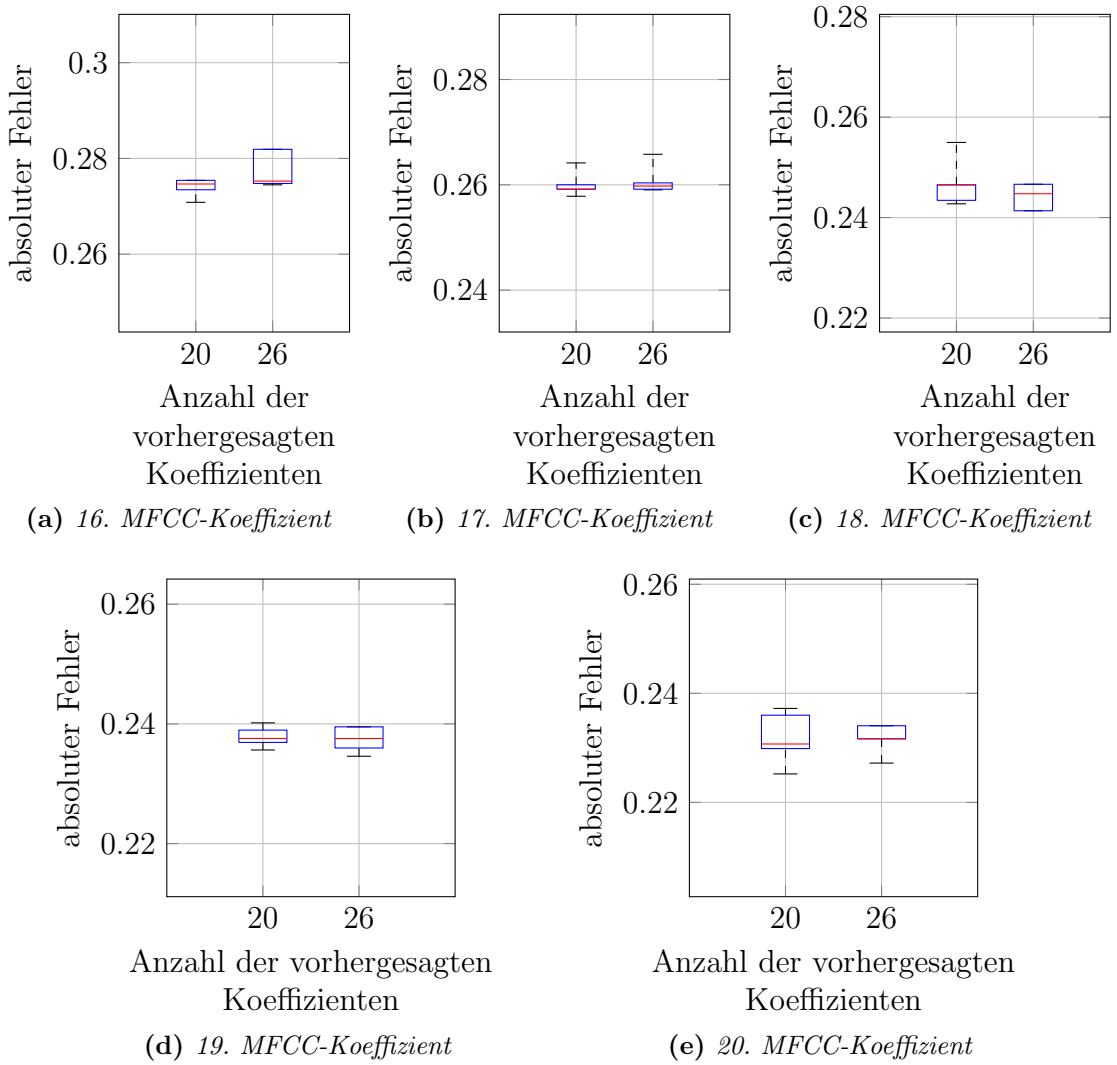
(a) *10. MFCC-Koeffizient* **(b)** *11. MFCC-Koeffizient* **(c)** *12. MFCC-Koeffizient*

(d) *13. MFCC-Koeffizient* **(e)** *14. MFCC-Koeffizient* **(f)** *15. MFCC-Koeffizient*

Abbildung 4.34: *Entwicklung des mittleren absoluten Fehlers des 10., 11. und 12. MFCC-Koeffizienten nach jeweils 10 Durchläufen je Experiment. Rote Linien markieren den Median, blaue Boxen den Bereich zwischen dem 25 %-Quartil und dem 75 %-Quartil und die Markierungen am Ende der gestrichelten Linien den kleinsten und den größten mittleren Fehler.*

Zusammengefasst lässt sich sagen, dass der Fehler eines einzelnen MFCC-Koeffizienten bis zum zehnten Koeffizienten meist ansteigt und bei den höheren MFCC-Koeffizienten mit kleinen Ausnahmen nahezu unverändert bleibt, wenn die Anzahl der vorhergesag-

ten MFCC-Koeffizienten größer wird. Da MFCC-Koeffizienten eine zueinander verschiedene Skalierung haben, reicht die Betrachtung des absoluten Fehlers je MFCC-Koeffizient für eine Abschätzung der Güte der Vorhersage allein nicht ganz aus. Die Skalierung muss in die Betrachtung miteinbezogen werden. Eine Möglichkeit ist, den prozentualen Anteil zu betrachten, den der absolute Fehler bezogen auf den vorherzusagenden Wert des jeweiligen MFCC-Koeffizienten einnimmt. Je kleiner der prozentuale Fehler der Vorhersage eines MFCC-Koeffizienten ist, desto genauer dann ist natürlich auch die Vorhersage des jeweiligen MFCC-Koeffizienten.



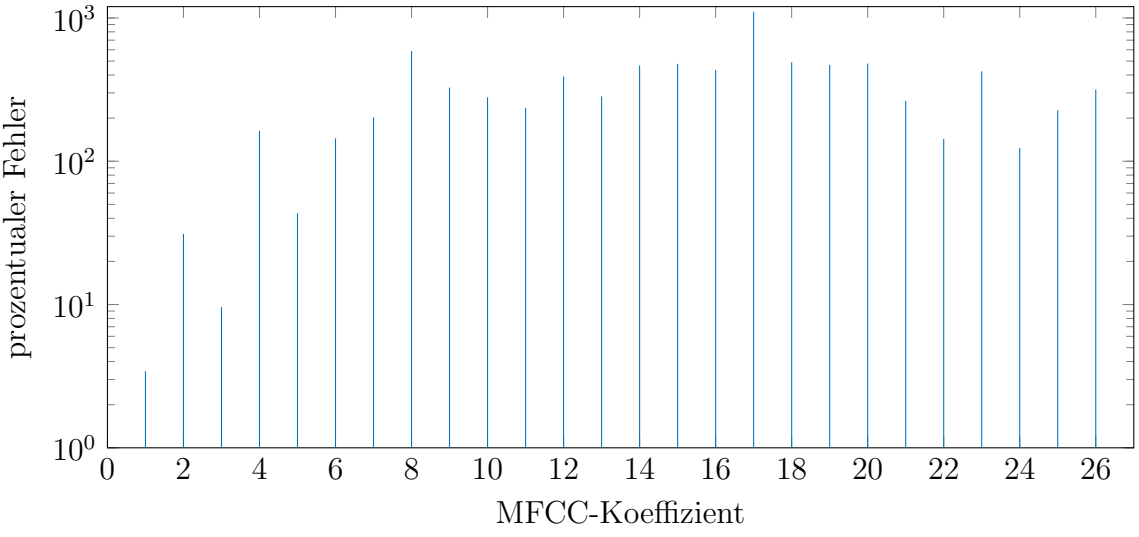
(a) *16. MFCC-Koeffizient* **(b)** *17. MFCC-Koeffizient* **(c)** *18. MFCC-Koeffizient*

(d) *19. MFCC-Koeffizient* **(e)** *20. MFCC-Koeffizient*

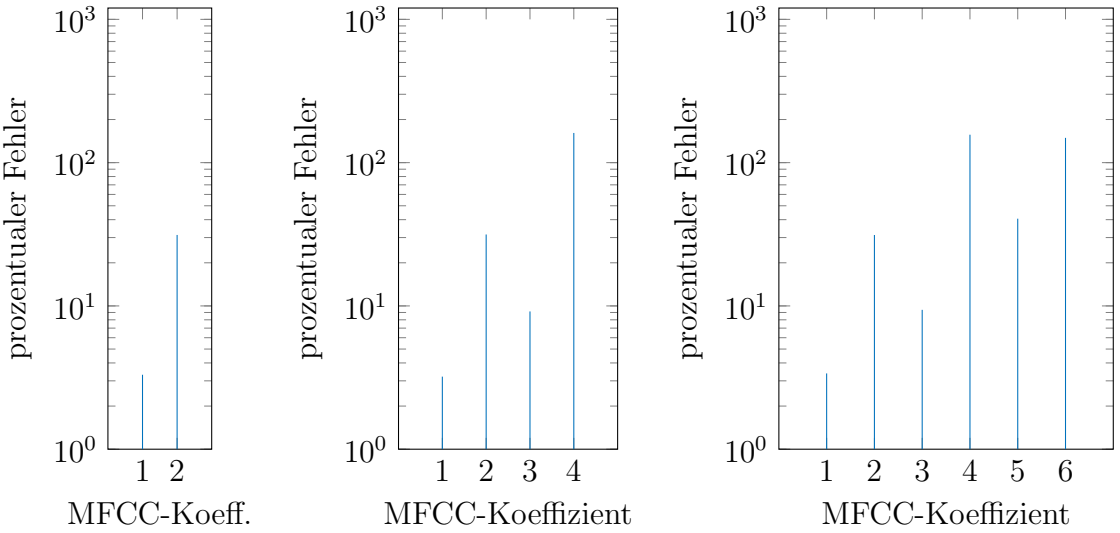
Abbildung 4.35: *Entwicklung des mittleren absoluten Fehlers des 16., 17. und 18. MFCC-Koeffizienten nach jeweils 10 Durchläufen je Experiment. Rote Linien markieren den Median, blaue Boxen den Bereich zwischen dem 25 %-Quartil und dem 75 %-Quartil und die Markierungen am Ende der gestrichelten Linien den kleinsten und den größten mittleren Fehler.*

In Abbildung 4.36a bis 4.36d ist deutlich zu sehen, dass die Vorhersagen der MFCC-Koeffizienten, mit Ausnahme der Vorhersage des ersten, zweiten und dritten MFCC-

Koeffizienten, im Grunde viel zu schlecht sind, um brauchbar zu erscheinen. Man kann auch gut sehen, dass durch Reduktion der Anzahl der vorherzusagenden MFCC-Koeffizienten keine Verbesserung eintritt. Das heißt aber, dass mit dem untersuchten einschichtigen Multi-Layer-Perzeptron maximal die ersten drei MFCC-Koeffizienten mit guter Genauigkeit vorhersagbar sind, wobei die Genauigkeit der Vorhersage des zweiten MFCC-Koeffizienten durchschnittlich nur ca. 70 Prozent beträgt.



(a) *Experiment 3.1*



(b) *Experiment 3.1* **(c)** *Experiment 3.2* **(d)** *Experiment 3.3*

Abbildung 4.36: *Die Grafik zeigt den gemittelten absoluten Fehler in Prozent vom vorherzusagenden MFCC-Koeffizienten für Experiment 3.6 (a), 3.1 (b), 3.2 (c) und 3.3 (d). Man kann sehen, dass schon ab dem zweiten MFCC-Koeffizienten die Vorhersage sehr schlecht wird und ab dem vierten MFCC-Koeffizienten weit vom eigentlichen Wert entfernt liegt.*

Es scheint also wenig sinnvoll, mehr als den ersten und dritten MFCC-Koeffizienten

mit Hilfe eines 100-Hidden-Neuronen-MLP aus den Gelenkwinkeldaten vorhersagen zu wollen. Da ein mittlerer Fehler von ca. drei bis vier Prozent bei der Vorhersage des ersten MFCC-Koeffizienten am ehesten vertretbar ist, noch vor den fast zehn Prozent des dritten MFCC-Koeffizienten, ist die genauste Vorhersage nur für den ersten MFCC-Koeffizienten zu erreichen.

Es stellt sich nun die Frage, welchen Anteil des Störgeräusches, verursacht durch die Elektromotoren während der Armbewegung, von der Vorhersage nur der ersten drei MFCC-Koeffizienten abgedeckt werden kann. Ist das Motorgeräusch durch die ersten drei MFCC-Koeffizienten ausreichend gut repräsentiert, um zum Beispiel eine Reduktion dieses Störgeräuschanteil vorzunehmen?

Bei der Beantwortung dieser Frage ist zu bedenken, dass MFCC-Koeffizienten die Koeffizienten eines Cepstrums sind. Diese beschreiben den Frequenzgehalt des Kurvenverlaufes eines Spektrums. Der erste MFCC-Koeffizient wird zum Beispiel das durchschnittliche Niveau der Koeffizienten des Spektrums beschreiben. Die Frage, ob die ersten drei MFCC-Koeffizienten hinreichend sind, wird in der abschließenden Diskussion in Kapitel 5 untersucht werden.

5 Diskussion und Ausblick

Zum Abschluss der Arbeit werden in diesem Kapitel die Ergebnisse der Experimente diskutiert und ein anschließend Ausblick auf offene Fragen und mögliche weiterführende Arbeiten gegeben.

5.1 Diskussion der Ergebnisse

Diese Arbeit beschäftigt sich mit der Modellierung der Eigengeräusche mit internen Modellen. Die Untersuchung interner Modelle wurde in diesem Fall auf die Modellierung von MFCC-Koeffizienten als auditorische Konsequenz, der durch Armbewegungen erzeugten Motorgeräusche, mit Vorwärtsmodellen beschränkt. Dazu wurden die MFCC-Koeffizienten über einem Zeitfenster von 40 Millisekunden Länge und einer Überlappung der Zeitfenster von 30 Millisekunden berechnet und eine Untersuchung der Vorhersage dieser MFCC-Koeffizienten aus damit synchronisierten Gelenkdaten mit Hilfe von verschieden konfigurierten MLPs vorgenommen. Dabei zeigte sich, dass zur Vorhersage des ersten MFCC-Koeffizienten ein Multi-Layer-Perzeptron mit einem 100-Neuronen-Hidden-Layer und die Verwendung der Gelenkwinkeländerung die genaueste und stabilste Vorhersage des ersten MFCC-Koeffizienten berechnet. Es wurde ferner untersucht, wie gut eine Vorhersage von mehr als einem MFCC-Koeffizienten mit einem solchen MLP vorgenommen werden kann.

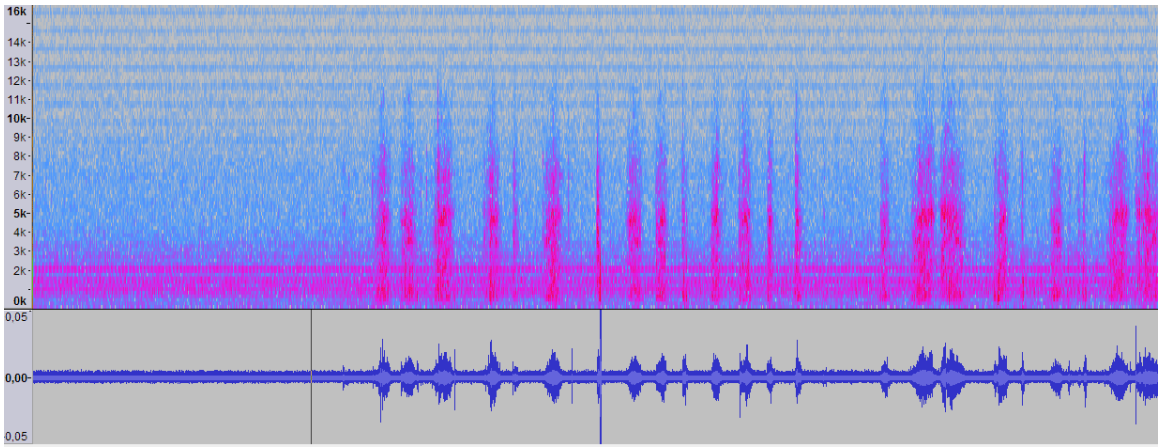


Abbildung 5.1: *Spektrum und Wellenform des ersten Audiokanals eines Teilabschnitts des Testdatensatzes. Der Teilabschnitt enthält auch den kleineren Teilabschnitt dessen Verläufe der MFCC-Koeffizienten und Gelenkdaten in den Abbildungen 4.2 bis 4.30 abgebildet sind.*

Das Ergebnis dieser Untersuchung ist auf den ersten Blick ernüchternd. Nur die ers-

ten drei MFCC-Koeffizienten konnten mit einem akzeptablen prozentualen Fehler vorhergesagt werden. Die Vorhersagen aller anderen MFCC-Koeffizienten postulieren Werte, die prozentual weit oberhalb des eigentlichen Wertes des MFCC-Koeffizienten liegen. Es scheint wenig sinnvoll mehr als die Vorhersage der drei ersten MFCC-Koeffizienten von einem 100-Hidden-Neuronen-MLP lernen zu lassen. Allerdings sind MFCC-Koeffizienten die Koeffizienten eines Cepstrums und zudem ab dem vierten Koeffizienten sehr klein.

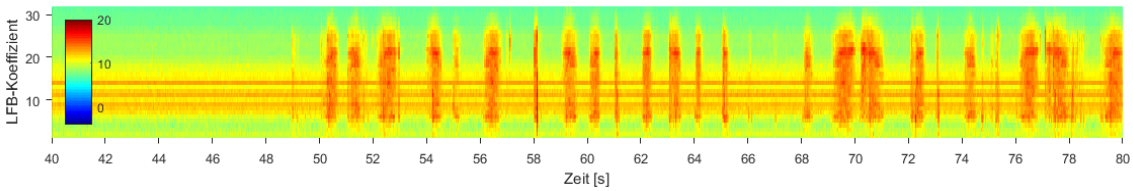


Abbildung 5.2: *Spektrale Darstellung der 32 LFB-Koeffizienten berechnet aus dem aufgenommenen Audiosignal des gleichen Teilabschnittes wie in Abbildung 5.1. Das zugrundeliegende Audiosignal enthält Lüfter-, Motor- und Belastungsgeräusche der mechanisch beweglichen Teile.*

Zur Berechnung der MFCC-Koeffizienten wird aus dem Leistungsspektrum eines Audiosignals zunächst durch Anwendung einer Mel-Filterbank (Abbildung 5.3) und anschließender Logarithmierung eine komprimierte Version des Leistungsspektrums erzeugt. Diese wird im Folgenden als Log-Filterbank-Spektrum (LFB-Spektrum) und deren Koeffizienten werden im Folgenden als Log-Filterbank-Koeffizienten (LFB-Koeffizienten) bezeichnet. Abbildung 5.2 zeigt das LFB-Spektrum des originalen Audiosignals (Abb. 5.1). Nachdem das LFB-Spektrum berechnet ist, werden durch Anwendung der DCT die MFCC-Koeffizienten berechnet.

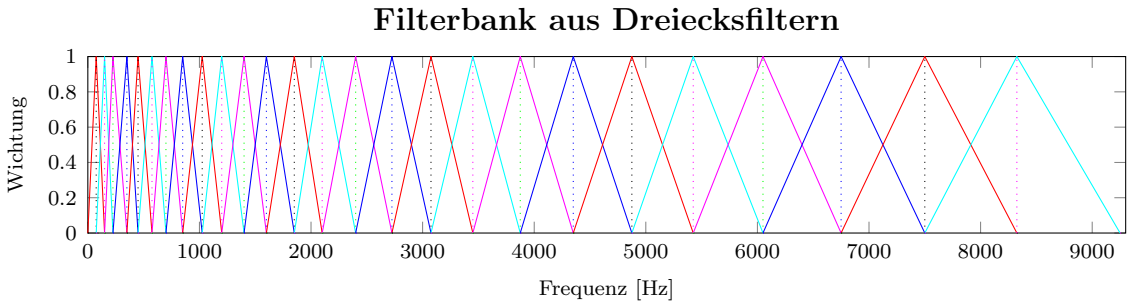


Abbildung 5.3: *Die Grafik zeigt den Teil der Filterbank zur Berechnung von 26 MFCC-Koeffizienten. Die gestrichelten Linien markieren die Position der Mittenfrequenz eines Filters und die durchgezogenen Linien den Verlauf der Wichtungsfunktion eines einzelnen Filters der Filterbank.*

Möchte man nun den Anteil an einem Spektrum betrachten, den die MFCC-Koeffizienten repräsentieren, müssen die MFCC-Koeffizienten erst wieder in dieses Spektrum zurückgeführt werden. Welchen Anteil die Vorhersage der MFCC-Koeffizienten an dem Spektrum des zu entstörenden Audiosignals hat, kann an dem LFB-Spektrum

abgeschätzt werden. Wendet man die inverse diskrete Kosinus-Transformation auf die MFCC-Koeffizienten an, erhält man die entsprechenden LFB-Koeffizienten.

Die vorhergesagten MFCC-Koeffizienten, nur bis zu deren Anteil am LFB-Spektrum, zurückzurechnen hat den Vorteil, dass keine weiteren Annahmen vorgenommen werden müssen, um aus den LFB-Koeffizienten ein unkomprimiertes Spektrum zu berechnen. Ein weiterer Vorteil ist, dass die LFB-Koeffizienten, die aus den vorhergesagten MFCC-Koeffizienten berechnet wurden, von den Koeffizienten des LFB-Spektrums des Audiosignals einfach subtrahiert werden können. Hierbei ist allerdings die Annahme zu treffen, dass sich das vorhergesagte Störgeräusch und das zu entstörende Nutzsignal, zum Beispiel Sprache, im Spektralbereich multiplikativ überlagern.

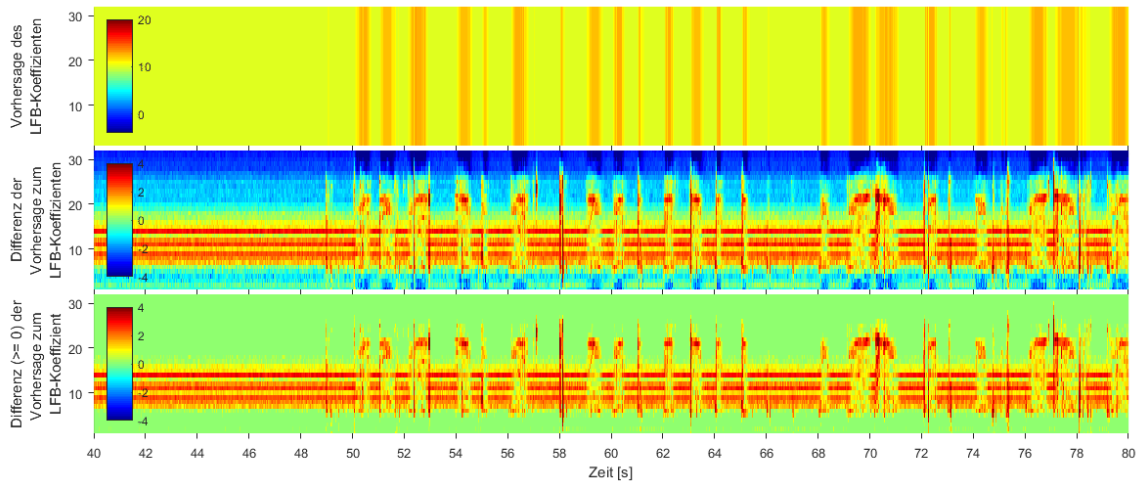


Abbildung 5.4: *Spektrale Darstellung der LFB-Koeffizienten berechnet aus der Vorhersage des ersten MFCC-Koeffizienten (Oben), der Differenz der LFB-Koeffizienten (Mitte) mit den LFB-Koeffizienten, dargestellt in Abbildung 5.2 und der gleichen Differenz, wobei Werte kleiner null bei null abgeschnitten wurden (Unten).*

Die Abbildungen 5.4 bis 5.6 zeigen, welchem Anteil des LFB-Spektrums (Abb. 5.2) der erste, die ersten zwei und die ersten drei vorhergesagten MFCC-Koeffizienten entsprechen. Die Grafiken zeigen das LFB-Spektrum, berechnet aus der Vorhersage der MFCC-Koeffizienten und darunter zwei Darstellungen der Differenz dieses LFB-Spektrums mit dem LFB-Spektrum, das aus den Audiodaten berechnet wurde (siehe Abb. 5.2).

In der untersten Darstellung der Differenz sind alle Werte kleiner null auf den Wert null gesetzt, da ein LFB-Spektrum keine negativen Koeffizienten haben kann. Die blauen Bereiche der jeweils mittleren Grafik machen unter anderem deutlich, wie sehr die resultierenden LFB-Koeffizienten der Vorhersage zu groß sind, die Vorhersage sozusagen daneben liegt.

Man kann sehen, dass schon die Vorhersage der drei ersten MFCC-Koeffizienten ein recht gutes Resultat liefern kann, nachdem die LFB-Koeffizienten davon berechnet und von LFB-Spektrum des Audiosignals abgezogen wurden. Interessant ist nun, welchen Effekt die Vorhersage der MFCC-Koeffizienten hat, die bei der Vorhersage

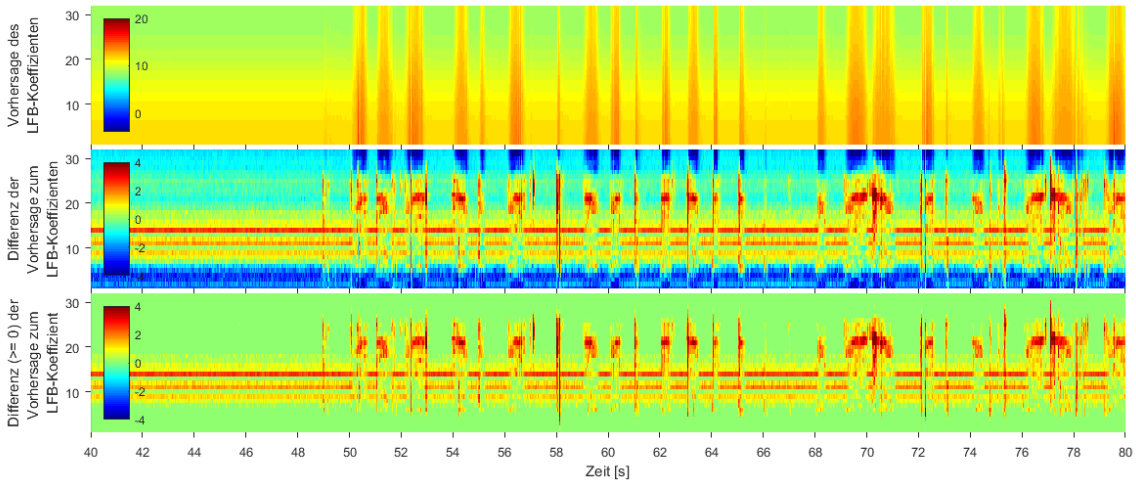


Abbildung 5.5: *Spektrale Darstellung der LFB-Koeffizienten berechnet aus der Vorhersage der ersten zwei MFCC-Koeffizienten (Oben), der Differenz der LFB-Koeffizienten (Mitte) mit den LFB-Koeffizienten, dargestellt in Abbildung 5.2 und der gleichen Differenz, wobei Werte kleiner null auf null gesetzt wurden (Unten).*

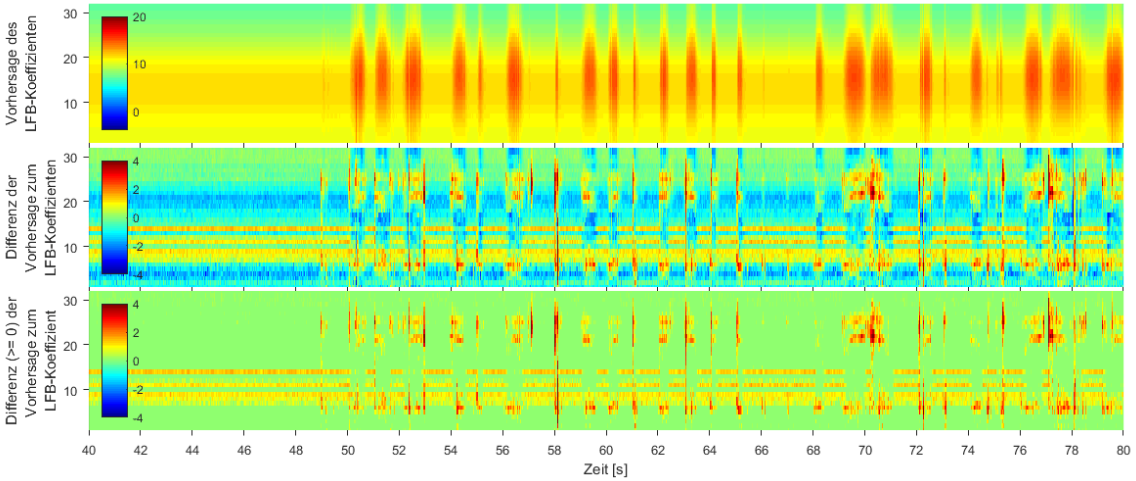


Abbildung 5.6: *Spektrale Darstellung der LFB-Koeffizienten berechnet aus der Vorhersage der ersten drei MFCC-Koeffizienten (Oben), der Differenz der LFB-Koeffizienten (Mitte) mit den LFB-Koeffizienten, dargestellt in Abbildung 5.2 und der gleichen Differenz, wobei Werte kleiner null auf null gesetzt wurden (Unten).*

einen so hohen prozentualen Fehler aufweisen, dass deren Verwendung nicht sinnvoll zu sein scheint.

Vergleicht man die jeweils mittlere Darstellung der Abbildungen 5.4 bis 5.8, dann ist zu sehen, dass die Vorhersage von mehr MFCC-Koeffizienten auch weniger große negative Anteile in der Differenz des aus der Vorhersage berechneten LFB-Spektrums vom LFB-Spektrum des Audiosignals enthalten sind. Das aus der Vorhersage berechnete LFB-Spektrum wird also genauer.

Schon Abbildung 5.7 zeigt, dass es durchaus sinnvoll ist, die höheren MFCC-Koeffizi-

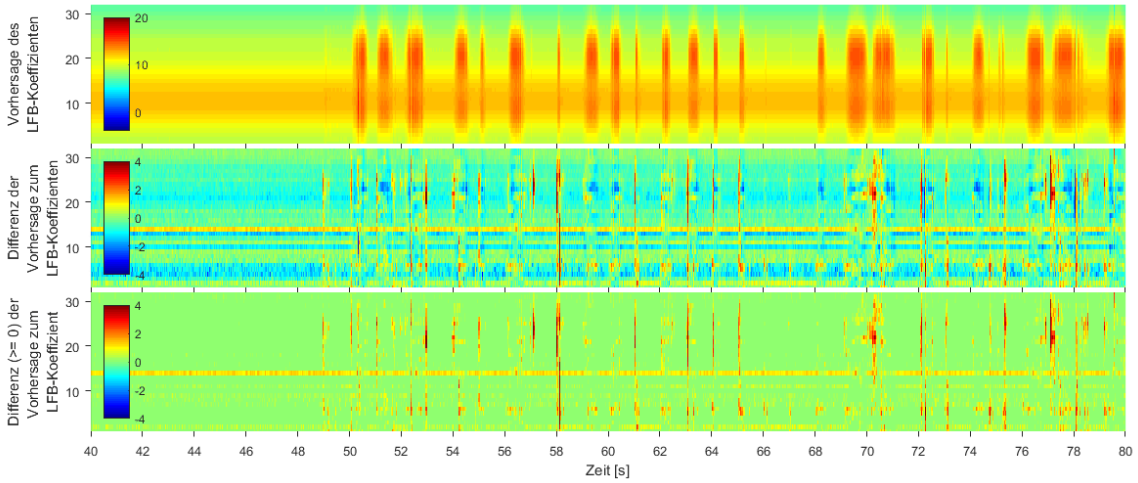


Abbildung 5.7: *Spektrale Darstellung der LFB-Koeffizienten berechnet aus der Vorhersage der ersten sechs MFCC-Koeffizienten (Oben), der Differenz der LFB-Koeffizienten (Mitte) mit den LFB-Koeffizienten, dargestellt in Abbildung 5.2 und der gleichen Differenz, wobei Werte kleiner null auf null gesetzt wurden (Unten).*

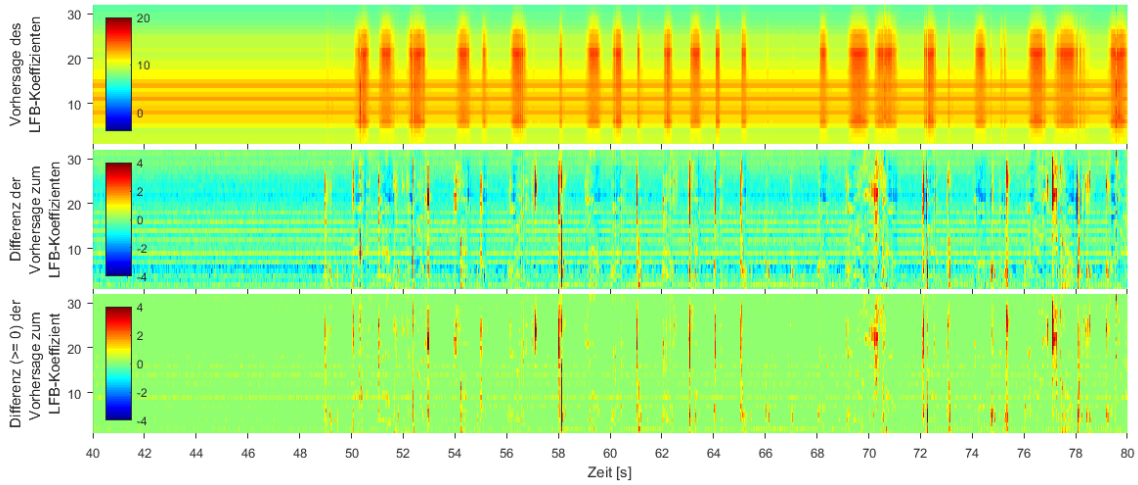


Abbildung 5.8: *Spektrale Darstellung der LFB-Koeffizienten berechnet aus der Vorhersage aller 26 MFCC-Koeffizienten (Oben), der Differenz der LFB-Koeffizienten (Mitte) mit den LFB-Koeffizienten, dargestellt in Abbildung 5.2 und der gleichen Differenz, wobei Werte kleiner null auf null gesetzt wurden (Unten).*

enten miteinzubeziehen. Werden die ersten sechs vorhergesagten MFCC-Koeffizienten bei der Berechnung des zugehörigen LFB-Spektrums genutzt, können fast alle akustischen Effekte entfernt werden, die durch die Ansteuerung der Motoren entstehen. Sogar dem Lüftergeräusch entsprechende Anteile an den LFB-Koeffizienten können unterdrückt werden. Werden alle 26 MFCC-Koeffizienten verwendet, kann das Lüftergeräusch nahezu vollständig mitmodelliert werden. Dies ist dadurch erklärbar, dass der Lüfter, wenn keine Bewegung ausgeführt wird, die stärkste verbleibende Quelle für Störgeräusche ist und das neuronale Netz das Lüftergeräusch mit der motorischen Aktion *keine Bewegung* assoziieren lernt.

5.2 Ausblick

In dieser Arbeit wurde untersucht, inwiefern ein einschichtiges Multi-Layer-Perzeptron geeignet ist, ein Vorwärtsmodell der akustischen Effekte der Bewegung des linken Arms des Roboters Nao aus den Motor- bzw. Sensordaten zu lernen. Anhand von Testdaten, die in den Audiodaten eines Audiokanals hauptsächlich das Lüftergeräusch und Bewegungsgeräusche des Armgelenkmotors enthalten, wurde abgeschätzt, wie gut die akustischen Effekte der Störquellen, wie Lüfter und Armgelenkmotor, aus den Gelenkdaten vorhergesagt werden können.

Die Audiodaten enthielten keine anderen akustischen Informationen, abgesehen von einigen Geräuschen mechanischer Belastung. So ist aufgrund eines fehlenden Nutzsignals in den untersuchten Testdaten eine Abschätzung der Güte einer Störgeräuschunterdrückung, zum Beispiel durch Ermittlung des Signal-Rausch-Abstand (SNR), nicht möglich. Eine weiterführende Untersuchung, wie gut das MLP geeignet ist, um eine Störgeräuschunterdrückung zu realisieren, sollte mit entsprechenden Daten überprüft werden, die neben den Störgeräuschen auch ein akustisches Nutzsignal enthalten. Mit einem solchen Testdatensatz kann dann analysiert werden, ob mit einem der untersuchten MLPs das Nutzsignal mit guter Qualität von den überlagerten Störgeräuschen befreit werden kann.

Die Berechnung eines Vektors von MFCC-Koeffizienten erfolgte durch Anwendung von Rechteckfenstern auf einem 40 ms langen Ausschnitt der Audiodaten, mit einer zeitlichen Überlappung von 30 ms. Die Verwendung anderer Fensterfunktionen, wie etwa ein Hannfenster oder ein Gaborfenster können bei der Berechnung des Kurzzeitspektrums ein Spektrum mit einem geringeren Anteil an Artefakten ergeben und damit eventuell auch eine bessere Basis zu einer genaueren Vorhersage von höheren MFCC-Koeffizienten sein. Es wäre interessant dies zu untersuchen. Welchen Einfluss die Größe des Zeitfensters und die Überlappung aufeinanderfolgender Fenster auf die Vorhersagequalität von MFCC-Koeffizienten haben, könnte weiterführend im Hinblick auf ein Entstörung des Audiosignals auch untersucht werden.

Die Menge an Trainingsbeispielen und die Anzahl der Epochen beim Training waren in allen Experimenten nicht verändert worden. Beide Hyper-Parameter haben jedoch signifikanten Einfluss auf die Dauer und Qualität des Lernprozesses. Ein Optimierung dieser zwei Hyper-Parameter könnte das Lernen der Vorhersage von MFCC-Koeffizienten verbessern und auch beschleunigen.

Es wurden die Untersuchungen in dieser Arbeit auf die Modellierung der akustischen Effekte eines einzelnen Gelenkes beschränkt. Eine weiterführend zu beantwortende Fragestellung ist, ob die Effekte von mehr als einem Gelenk auf ähnliche Weise modellierbar sind. Zusätzlich dazu steht die Frage im Raum, ob die Nutzung zusätzlicher Audiokanäle für die Realisierung der Vorhersage der akustischen Effekte mehrerer Gelenke von Vorteil sein kann.

Eine andere interessante Fragestellung ist, ob sich auch ein Rückwärtsmodell mit einem MLP realisieren lässt, mit dem aus den Audiodaten die Bewegungen eines oder mehrerer Gelenke des Roboters abgebildet werden können. Auch hierbei könnte die Nutzung mehrerer Audiokanäle vorteilhaft sein, was zusätzlich zu untersuchen wäre.

Literaturverzeichnis

[Bar] Lawrence W. Barsalou. Grounded cognition. volume 59, pages 617–645.

[BCPK05] Simone Bosbach, Jonathan Cole, Wolfgang Prinz, and Günther Knoblich. Inferring another's expectation from action: the role of peripheral sensation. *Nature neuroscience*, 8(10):1295–1297, 2005.

[DD05] Anthony Dearden and Yiannis Demiris. Learning forward models for robots. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1440–1445, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc. URL: `\url{http://dl.acm.org/citation.cfm?id=1642293.1642521}`.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[IKSM10] Akinori Ito, Takashi Kanayama, Motoyuki Suzuki, and Shozo Makino. Internal noise suppression for speech recognition by small robots. In Moonis Ali, José Manuel Benítez Sánchez, Colin Fyfe, Nicolás García-Pedrajas, and Francisco Herrera, editors, *Trends in applied intelligent systems*, volume 6096 of *Lecture notes in computer science Lecture notes in artificial intelligence*, pages 2685–2688, Berlin [u.a.], 2010. Springer. URL: `\url{http://dblp.uni-trier.de/db/conf/interspeech/interspeech2005.html#ItoKSM05}`.

[INR+11] Gokhan Ince, Kazuhiro Nakadai, Tobias Rodemann, Jun-ichi Imura, Keisuke Nakamura, and Hirofumi Nakajima. Incremental learning for ego noise estimation of a robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 131–136, Piscataway, NJ, 2011. IEEE.

[JFa] Steven Johnson and Matteo Frigo. Fft accuracy benchmark results [online]. URL: `\url{http://www.fftw.org/accuracy/}` [cited 17.11.2015].

[JFb] Steven Johnson and Matteo Frigo. Fft accuracy benchmark results: 1.6 ghz pentium m (banias), intel compilers [online]. URL: `\url{http://www.fftw.org/accuracy/PentiumM-1.6GHz-icc/}` [cited 17.11.2015].

[JFc] Steven Johnson and Matteo Frigo. Fft benchmark results [online]. URL: `\url{http://www.fftw.org/speed/}` [cited 17.11.2015].

[JFd] Steven Johnson and Matteo Frigo. Fft benchmark results: 1.6 ghz pentium m (banias), gnu compilers [online]. URL: `\url{http://www.fftw.org/speed/PentiumM-1.6GHz-gcc/}` [cited 17.11.2015].

[MH04] Beate Meffert and Olaf Hochmuth. *Werkzeuge der Signalverarbeitung: Grundlagen, Anwendungsbeispiele, Übungsaufgaben.* Informatik. Pearson Studium, München and Boston [u.a.], 2004.

[Nie83] Heinrich Niemann. *Klassifikation von Mustern.* Springer, Berlin, Heidelberg [usw.], 1983. URL: \url{http://dx.doi.org/10.1007/978-3-642-47517-7}.

[NNN+06] Yoshitaka Nishimura, Mikio Nakano, Kazuhiro Nakadai, Hiroshi Tsujino, and Mitsuru Ishizuka. Speech recognition for a robot under its motor noises by selective application of missing feature theory and mllr. In *ISCA tutorial and research workshop on Statistical and perceptual audition*, 2006.

[Oou06] Takuya Ooura. Ooura's mathematical software packages [online]. 2006. URL: \url{http://www.kurims.kyoto-u.ac.jp/~ooura/} [cited 08.12.2015].

[RB93] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, Piscataway NJ, 1993. IEEE.

[RN95] Stuart J. Russell and Peter Norvig. *Artificial intelligence: A modern approach.* Prentice Hall series in artificial intelligence. Prentice Hall, Englewood Cliffs, N.J., 1995.

[SMD10] Bob L. Sturm, Marcela Morvidone, and Laurent Daudet. Musical instrument identification using multiscale mel-frequency cepstral coefficients. 2010.

[WGJ95] D. Wolpert, Z. Ghahramani, and M. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

[WMK98] Daniel M. Wolpert, R.Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347, 1998.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 26. November 2018 .....