DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Authentifizierung von Benutzern mittels Sprechererkennung

Studienarbeit T3200

über die Theoriephasen des dritten Studienjahrs

an der Fakultät für Technik
im Studiengang Informatik

an der DHBW Ravensburg
Campus Friedrichshafen

von
Johannes Brandenburger, Lukas Braun, Henry Schuler

17. Juli 2023

Name, Matrikelnummer:	Johannes Brandenburger, 7255587
	Lukas Braun, 1688304
	Henry Schuler, 5220542
Kurs:	TIT20
Bearbeitungszeitraum:	01.10.2022 - 17.07.2023
Betreuer der Hochschule:	Prof. Dr. Jürgen Schneider

Gender Erklärung

Aus Gründen der besseren Lesbarkeit wird in dieser Bachelorarbeit auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers (m/w/d) verzichtet. Sämtliche Formulierungen gelten gleichermaßen für alle Geschlechter.

Selbstständigkeitserklärung

gemäß Ziffer 1.1.13 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2017.

Wir versichern hiermit, dass wir unsere Bachelorarbeit (bzw. Projektarbeit oder Studienarbeit bzw. Hausarbeit) mit dem Thema:

Authentifizierung von Benutzern mittels Sprechererkennung

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Friedrichshafen, 17. Juli 2023	
Ort, Datum	Johannes Brandenburger
Friedrichshafen, 17. Juli 2023	
Ort, Datum	Lukas Braun
Friedrichshafen, 17. Juli 2023	
Ort, Datum	Henry Schuler

Zusammenfassung

verfasst von: Henry Schuler

In dieser Studienarbeit wird die Authentifizierung von Benutzern mittels Stimmmerkmalen erarbeitet und untersucht. Dazu werden zunächst basierend auf einer ausführlichen Literaturrecherche verschiedene Verfahren, Ansätze und Merkmale verglichen, um anschließend eine genauere Eingrenzung der Studienarbeit vorzunehmen. Folgend ergibt sich als Ziel dieser Arbeit das Finden der optimalen Kombination der Sprechermerkmale LPC, LPCC, MFCC und Delta MFCC für die Sprecherauthentifizierung unter Verwendung eines neuronalen Netzes.

Für die Evaluierung der besten Kombination wird ein eigenständiges Versuchssystem konzipiert und implementiert. Dieses vergleicht dabei verschiedene Werte für die Parameter Anzahl der Frames, Länge der Frames und Anzahl der LPC, LPCC, MFCC und Delta MFCC Koeffizienten in über 500 verschiedenen Kombinationen.

Parallel zum Versuchssystem wird ein Demosystem konzipiert und implementiert, welches einen Authentifizierungsprozess unter Verwendung der ermittelten optimalen Konfiguration abbildet. Dieses dient zur Präsentation des Ergebnisses dieser Arbeit.

Die Auswertungen des Versuchssystem führen zu der Erkenntnis, dass für eine optimale Authentifizierung hauptsächlich MFCC und Delta MFCC Koeffizienten relevant sind. LPC und LPCC Koeffizienten unterstützen den Authentifizierungsprozess nur gering und erreichen alleinstehend keine ausreichenden Ergebnisse.

Die Arbeit kommt zu dem Schluss, dass das optimale Authentifizierungssystem 15000 Frames mit einer Länge von je 800 Samples verwendet und dabei 27 MFCC und 13 Delta MFCC Koeffizienten pro Frame berechnet. Dabei wird zu 90 % die korrekte Person, zu 10 % keine Person und zu 0 % die falsche Person authentifiziert.

Abschließend werden verschiedene Sicherheitsaspekte in Zusammenhang mit der Sprecherauthentifizierung beleuchtet und bewertet.

Die Arbeit ist besonders für Softwareentwickler interessant, die im Bereich der Python- und Webentwicklung in Kombination mit biometrischer Authentifizierung tätig sind.

Schlüsselwörter Sprecherauthentifizierung, LPC, LPCC, MFCC, Delta MFCC, Neuronales Netz

Abstract

verfasst von: Henry Schuler

This study deals with authenticating users based on features of their voice. Based on thorough research, different features and approaches are compared to specify the goal of the study. As a result, the goal of the study is defined as to evaluate the optimal combination of the voice features LPC, LPCC, MFCC, and delta MFCC for speaker authentication using a neural network.

To evaluate this combination, an evaluation system (Versuchssystem) is designed and implemented. Within over 500 combinations, the system compares different values for the parameters amount of frames, frame length, and amount of LPC, LPCC, MFCC, and delta MFCC coefficients.

In addition to the evaluation system, a demo system is created, which implements an authentication process using the evaluated optimal combination of parameters. The purpose of this system is to present the achievements of this study.

The evaluation of the evaluation system concludes, that to achieve the best authentication results, mainly MFCC and delta MFCC features are required. Adding LPC and LPCC features to the authentication process shows little improvement, which is why they are not used.

In summary, the optimal authentication system uses 15000 frames with a length of 800 samples per frame. For each frame, a set of 27 MFCC and 13 delta MFCC coefficients are calculated and used within the neural network. Using this feature set, the correct user is authenticated in 90 % of all test cases, while no user is authenticated in 10 % and the wrong user in 0 % of all test cases.

Following the evaluation, the study is supplemented by a consideration of security aspects concerning speaker authentication. Therefore different aspects are presented and evaluated.

This study is of particular interest to software developers working in the field of Python and web development in combination with biometric authentication.

Keywords speaker authentication, LPC, LPCC, MFCC, delta MFCC, neuronal network

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

1.1 Problemstellung und Relevanz

verfasst von: Lukas Braun

In der heutigen digitalen Welt, in der Sicherheit und der Schutz personenbezogener Daten von entscheidender Bedeutung sind und viele Prozesse online und digital abgewickelt werden, gewinnt die Benutzerauthentifizierung als zentraler Faktor zunehmend an Bedeutung. Traditionelle Authentifizierungsmethoden, wie Passwörter oder PINs haben ihre Grenzen und können durch technologische Fortschritte und kreative Angriffsstrategien oft überwunden werden. Um den wachsenden Anforderungen der Authentifizierung gerecht zu werden, haben Forscher und Entwickler alternative Ansätze erforscht.

Ein vielversprechender entstandener Ansatz ist die Authentifizierung eines Benutzers durch seine Stimmmerkmale. Die Stimme ist wie zum Beispiel der Fingerabdruck ein eindeutiges biometrisches Merkmal. Jeder Mensch besitzt eine einzigartige Kombination von Stimmmerkmalen, die es ermöglichen, einen Nutzer eindeutig zu identifizieren. Dieser Ansatz bietet die Möglichkeit einer natürlichen und bequemen Authentifizierung, da die meisten Menschen über ein funktionierendes Sprachorgan verfügen.

1.2 Zielsetzung

verfasst von: Lukas Braun

Im Rahmen dieser Studienarbeit wird untersucht wie zuverlässig und wie sicher die Authentifizierung eines Benutzers über seine Sprachmerkmale realisierbar ist.

Dafür sollen die zugrunde liegenden Technologien und Verfahren zur Stimmerkennung erörtert werden und verschiedene Ansätze zur Benutzerauthentifizierung anhand der Stimmmerkmale beleuchtet und verglichen werden. Auf Grundlage dieser Recherche soll ein Sprecherauthentifikationssystem entwickelt werden. Dieses Systems soll in eine einfache Web-Anwendung integriert werden, dabei soll der Authentifizierungsprozess des Sprecherkennungssystems serverseitig angebunden werden. Abschließend sollen die potenziellen Sicherheitsrisiken des Systems betrachtet und bewertet werden.

Die Erkenntnisse dieser Arbeit zeigen den aktuellen Stand der Forschung und Entwicklung im Bereich der Benutzerauthentifizierung mittels Stimmmerkmalen auf. Durch das bessere Verständnis dieser Technologie können potenzielle Anwendungsfälle und mögliche Innovationen identifiziert werden.

1.3 Aufbau der Arbeit

verfasst von: Lukas Braun

Der Aufbau der vorliegenden Studienarbeit gliedert sich in mehrere Abschnitte. Zu Beginn werden die Grundlagen in Bezug zur Benutzerauthentifizierung mittels Stimmmerkmalen erläutert, dabei werden die grundlegenden Algorithmen und Verfahren zur Extraktion von Stimmmerkmalen dargestellt. In einem Folgeschritt wird der Stand der Technik beleuchtet, hierzu werden mehrere wissenschaftliche Arbeiten aus anderen Forschungen verglichen. Darauf basierend findet eine Technologieentscheidung, sowie die Konzeption und Umsetzung des Versuchssystems und des Demosystems statt. Anschließend werden die Ergebnisse des Versuchssystems evaluiert. In einem letzten Schritt werden Sicherheitsrisiken von Sprecherauthentifikationssystemen analysiert und bewertet. Die Arbeit endet mit einer kritischen Reflexion, einem Fazit und einem Ausblick.

2 Grundlagen

verfasst von: Johannes Brandenburger, Lukas Braun, Henry Schuler

Um die Hintergründe und Herangehensweisen der Studienarbeit nachzuvollziehen müssen zunächst verschiedene Grundlagen vorgestellt werden. Dabei findet eine Definition des Begriffs Sprecherauthentifikation statt. Anschließend wird ein Basisverständnis für die technische Umsetzung mittels eines Exkurses über den Aufbau der menschlichen Stimme hergestellt. Die konkret verwendeten Algorithmen werden in dem Kapitel Audioanalyse beschrieben. Außerdem findet eine kurze Einführung in den Bereich der künstlichen Intelligenz, sowie der Neuronalen Netze (NN) statt. Abschließend werden zwei Ansätze zur Analyse von Sicherheitsrisiken vorgestellt.

2.1 Sprecherauthentifikation

verfasst von: Henry Schuler

In der Kryptographie beschreibt der Begriff Authentifizierung bzw. Authentifikation „eine Identität durch eine identitätsgebundene Information zu überprüfen“ (Tsolkas und Schmidt 2017, S. 129). Zu den identitätsgebundenen Informationen zählt unter anderem die biometrische Information der Stimme für die Sprecherauthentifikation.

Zu den heutigen Einsatzgebieten der Sprecherauthentifikation zählen hauptsächlich Telefon-Hotlines, welche anstelle oder in Ergänzung zu wissensbasierten Sicherheitsfragen auch die Informationen der Stimme auswerten, um diese für die Authentifizierung zu verwenden. So kommt die Sprecherauthentifikation beispielsweise bereits bei den Hotlines der schweizer Banken Postfinance, Migros und Cler, aber auch der deutschen Telekom zum Einsatz, wodurch eine Authentifizierung mittels Kundendaten unterstützt wird oder gänzlich entfällt (vgl. Anz 2023) (vgl. *Authentifizierung mit Stimmerkennung* o. D.) (vgl. *Meine Stimme – mein Passwort | Telekom Hilfe* o. D.).

Im konkreten Anwendungsfall der Sprecherauthentifikation wird zwischen textunabhängigen und textabhängigen Authentifizierungssystemen mit open-set oder closed-set Datensätzen unterschieden.

2.1.1 Textunabhängig und Textabhängig

verfasst von: Henry Schuler

Textunabhängige Systeme beziehen sich ausschließlich auf die Charakteristiken der Stimme, ohne dabei den Inhalt des Gesprochenen auszuwerten. Für die Authentifizierung kann somit

eine beliebige Wortkombination eingesprochen werden.

Textabhängige Systeme hingegen implementieren zusätzlich eine Texterkennung, wodurch neben der Stimmcharakteristik auch der gesprochene Satz dem vorgegebenen Satz entsprechen muss (vgl. Thullier, Bouchard und Menelas 2017, S. 2). Dadurch kann das Authentifizierungssystem mit einem bestimmten Satz trainiert werden, wodurch die Authentifizierung verbessert werden kann (vgl. Zulfiqar u. a. 2010, S.). Im Gegensatz zu textunabhängigen Systemen steigen jedoch die Anforderungen an die Trainingsdaten stark an, da der verwendete Satz von jedem Sprecher einzeln eingesprochen werden muss und somit keine bereits existierenden Aufzeichnungen verwendet werden können.

2.1.2 Open- und closed-set

Open- und closed-set Sprecherauthentifikation unterscheidet sich in der Zusammensetzung und Klassifizierung des Test- und Anwendungsdatensatzes. Während bei open-set Anwendungen sowohl Sprachaufzeichnungen von bekannten, als auch unbekannten Sprechern ausgewertet werden, ist die Anwendung bei closed-set ausschließlich auf die bekannten Sprecher begrenzt.

Somit ordnet eine closed-set Anwendung eine Sprachaufzeichnung immer einem der bekannten Sprecher zu, während in einer open-set Anwendung auch eine Klassifikation als „unbekannt“ stattfindet.

2.2 Menschliche Stimme

Um die grundlegenden Konzepte hinter den in dieser Arbeit verwendeten Verfahren zur Parameterextraktion zu verstehen, werden in diesem Kapitel die wesentlichen Grundlagen zu der menschlichen Stimme vorgestellt. Dabei wird sowohl auf die Stimmerzeugung, als auch die Stimmwahrnehmung eingegangen.

2.2.1 Stimmerzeugung

Die Stimmerzeugung unterteilt sich in zwei Phasen: die Phonation (Stimmbildung) und die Resonanz. Bei der Phonation werden die Stimmbänder im Kehlkopf durch den Luftstrom in Schwingung versetzt, wodurch ein Ton (bestehend aus dem Grundton, sowie Obertönen) erzeugt wird. Abhängig von der Spannung und Länge der Stimmbänder kann dabei die Tonhöhe des erzeugten Grundtons variiert werden. Da sich der Kehlkopf bei Männern während des Stimmbruchs stärker vergrößert, liegt ihre Grundfrequenz tiefer (vgl. Clauss und Clauss 2018, S. 278-279).

Der erzeugte Grundton gelangt anschließend in den Vokaltrakt, bestehend aus dem Rachen-,

Mund- und Nasenraum. Dieser besitzt spezifische Resonanzeigenschaften, wodurch die Frequenzkomponenten des Grundtons verändert (verstärkt oder abgeschwächt) werden. Durch die Artikulatoren Zunge, Lippen, Kiefer und Gaumensegel kann der Vokaltrakt und damit dessen Resonanzeigenschaften verändert werden. Somit können die verschiedenen Laute, aus denen die Sprache aufgebaut ist, erzeugt werden (vgl. Pfister und Kaufmann 2017, S. 13-14).

In der deutschen und englischen Sprache werden die Buchstaben des Alphabets grundlegend in Vokale und Konsonanten unterteilt. Im Folgenden werden die beiden Kategorien kurz erläutert.

Vokale Vokale zählen zu den stimmhaften Lauten. Das bedeutet, dass hier die Stimmbänder im Kehlkopf einen Grundton erzeugen, welcher durch den Vokaltrakt verstärkt wird. Die Luft kann dabei ungehindert durch den Mund ausströmen. Die unterschiedlichen Vokale werden besonders durch die Position der Zunge und der Lippen gebildet (vgl. Pfister und Kaufmann 2017, S. 14).

Konsonanten Im Gegensatz zu den Vokalen zeichnen sich Konsonanten dadurch aus, dass der Luftstrom teilweise oder vollständig durch eine Verengung behindert wird. Die Verengung wird durch die Artikulatoren erzeugt. Dabei gibt es sowohl stimmhafte, als auch stimmlose Konsonanten (vgl. Pfister und Kaufmann 2017, S. 15).

2.2.2 Stimmwahrnehmung

Im Gegensatz zu der Stimmerzeugung, lässt sich die Stimmwahrnehmung nicht ausschließlich anhand der Anatomie des menschlichen Ohrs erklären. Deshalb werden in diesem Kapitel Eigenschaften der Stimme betrachtet, welche in Verbindung mit der Stimmwahrnehmung stehen. Dazu zählen die Konzepte der Formanten und der Tonheit.

Formanten Bei der Stimmerzeugung werden bestimmte Frequenzbereiche durch den Vokaltrakt verstärkt. Diese Frequenzbereiche werden als Formanten bezeichnet und durch die Attribute (Mitten-)Frequenz, Bandbreite und Amplitude beschrieben. Die Formanten werden in aufsteigender Frequenz angeordnet und mit $F_1, F_2, F_3 \ldots$ bezeichnet. Da die Formanten durch die Resonanzeigenschaften des Vokaltraktes erzeugt werden, sind diese besonders für die Erkennung von Vokalen relevant (vgl. Pfister und Kaufmann 2017, S. 19-20). Dabei ist die Position der Formanten sowohl von dem jeweiligen Vokal, als auch dem jeweiligen Sprecher abhängig.

Tonheit In der Psychoakustik unterscheidet sich die wahrgenommene Tonhöhe (auch Tonheit genannt) von der physikalischen Tonhöhe. Die physikalische Tonhöhe wird durch die Grundfre-

quenz eines Tons bestimmt. Eine Verdopplung der Grundfrequenz führt dabei zu einer Verdopplung der Tonhöhe. Der Mensch nimmt jedoch diese Frequenzdifferenzen unterschiedlich wahr. Während für geringe Frequenzen bis zu 500 Hz ein Ton mit doppelter Frequenz auch doppelt so hoch wahrgenommen wird, ist für Töne höherer Frequenz ein größerer Abstand notwendig. Aus diesem Grund wurde im Jahr 1937 die Mel-Skala durch Stevens, Volkman und Newmann vorgeschlagen (vgl. Maschke und Jakob 2018, S. 11). Diese verläuft logarithmisch zur Frequenz, was auf den Aufbau der Basilarmembran im Innenohr zurückzuführen ist (vgl. Kröger 2018, S. 54). Als Bezug zwischen der Frequenz und der Mel-Skala wurde dabei die Frequenz 1000 Hz gewählt, welche 1000 mel entspricht. Für die Umrechnung zwischen den beiden Skalen wird folgende Formel verwendet (vgl. Pfister und Kaufmann 2017, S. 94-95):

$$h(f) = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{1}$$

2.3 Audioanalyse

verfasst von: Johannes Brandenburger, Henry Schuler

Basierend auf den biologischen Grundlagen der menschlichen Stimme, welche im vorangehenden Kapitel vorgestellt wurden, beschäftigt sich dieses Kapitel nun mit der Generierung von Parametern aus Stimmaufzeichnungen, die einen Rückschluss auf den Sprecher erlauben. Dazu wird zunächst auf die benötigten Vorverarbeitungsschritte eingegangen, die zur Weiterverarbeitung der aufgezeichneten Audiodatei benötigt werden. Anschließend wird das grundlegende Konzept der Fourier Analyse vorgestellt, welches zusammen mit dem vorangehenden Kapitel der Stimmwahrnehmung die Basis für die Mel Frequency Cepstral Coefficients (MFCC) bildet. Abschließend werden basierend auf den Konzepten der Stimmerzeugung die Linear Predicitve Coding (LPC) Koeffizienten eingeführt.

2.3.1 Signalvorverarbeitung

verfasst von: Johannes Brandenburger, Henry Schuler

Um ein gegebenes Audiosignal einheitlich verarbeiten zu können, muss dieses zunächst mittels verschiedener Verfahren vorbereitet werden. Ziel dieser Vorverarbeitung ist es, die Effizienz und Effektivität des anschließenden Verarbeitungsprozesses zu erhöhen und somit ein verbessertes Ergebnis zu erzielen (vgl. Lokesh und Devi 2019, S. 11672). Die Vorverarbeitung im Rahmen dieser Arbeit beinhaltet die vier Schritte Rauschreduzierung, Pausen entfernen, Framing und Windowing, welche im Folgenden genauer erläutert werden.

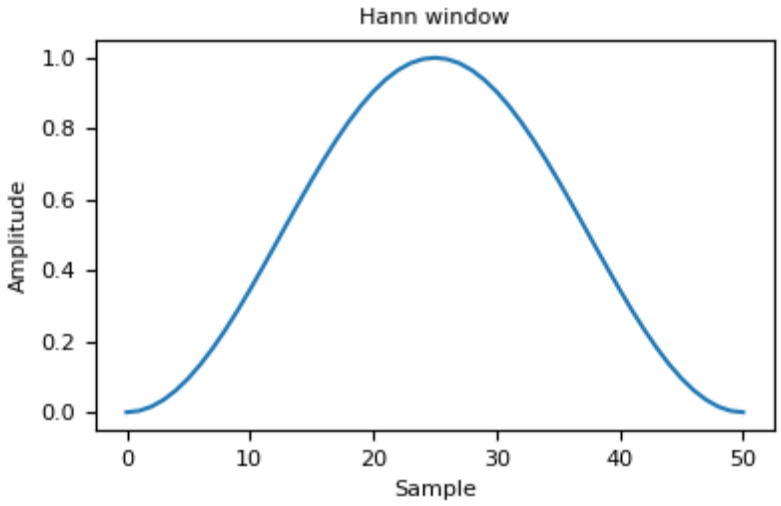


Abb. 2.1: Von Hann Fensterfunktion (*numpy.hanning — NumPy v1.24 Manual* 2022)

Rauschreduzierung Um störende Frequenzen aus dem Audiosignal zu entfernen, wird eine Rauschreduzierungsfunktion verwendet. Die in dieser Arbeit verwendete Funktion nutzt den sogenannten Spectral Noise Gate Algorithmus. Dabei wird zunächst die Signatur des Rauschens ermittelt. Basierend darauf kann das Rauschen anschließend verringert werden (vgl. Kiapuchinski, Lima und Kaestner 2012, S. 25).

Pausen entfernen Die für die Sprecherauthentifikation relevanten Daten stecken in dem aufgezeichneten Signal der Stimme. Sprechpausen innerhalb des Audiosignals enthalten somit keine brauchbaren Informationen, weshalb diese herausgefiltert werden müssen. Durch den vorangehenden Schritt der Rauschreduzierung kann hier ein stark vereinfachtes Verfahren gewählt werden. Liegt das Signal für einen definierten Zeitraum unterhalb einer definierten Lautstärke, werden die entsprechenden Signalwerte aus dem Gesamtsignal entfernt.

Framing Für eine detaillierte Analyse des Audiosignals muss dieses in kleinere Blöcke unterteilt werden. Dieser Prozess wird als Framing bezeichnet. Dabei muss zunächst eine einheitliche Blockgröße festgelegt werden. Da Stimmsignale aufgrund der Eigenschaften des Vokaltrakts über eine Periode von 10-30 ms stationär sind, wird eine Blockgröße in dieser Zeitordnung verwendet. Zusätzlich wird eine Überlagerungszeit definiert, welche eine Überlappung der einzelnen Blöcke verursacht. Durch die Überlappung wird ein Zusammenhang zwischen zwei benachbarten Frames und damit auch den anschließend berechneten Koeffizienten hergestellt (vgl. Richter u. a. 2022, S. 457).

Windowing Um die bei der Unterteilung des Audiosignals entstandenen Diskontinuitäten aufzulösen, wird eine Fensterfunktion auf die einzelnen Blöcke angewendet. Abbildung 2.1 zeigt die von Hann Fensterfunktion, welche neben dem Hamming Fenster zu den typischen Fensterfunktionen in der Audiosignalverarbeitung zählt. Durch den Nulldurchgang am Anfang und Ende der Fensterfunktion werden die Amplituden des Blocksignals nach Anwenden der Funktion an den Grenzen auf Null gezogen, wodurch sich ein kontinuierlicher, periodischer Signalverlauf ergibt (vgl. Richter u. a. 2022, S. 462).

Wird der Schritt des Windowing nicht durchgeführt, führt dies zu einem Phänomen namens „spectral leakage“. Bei der Transformation des Signals von dem Zeitbereich in den Frequenzbereich resultiert der Amplitudensprung an den Blockenden in der Registrierung einer Vielzahl von Frequenzen. Wie der Name bereits beschreibt, wird aus einer eindeutigen Frequenz, ein Spektrum aus Frequenzen, die nicht Teil des Signals sind (vgl. Wu, Chen und Chen 2012, S. 1296).

2.3.2 Fourier Analyse

verfasst von: Johannes Brandenburger

Die Grundlage für die Fourier Analyse stellt die von Jean-Baptiste Joseph Fourier entwickelte Fourier Reihe dar. In diesem Unterkapitel wird dabei zunächst die Fourier Reihe vorgestellt und anschließend die in dieser Arbeit verwendete Fast Fourier Transform (FFT) daraus schrittweise abgeleitet.

Fourier-Reihe Die Fourier-Reihe ist ein Verfahren, mit dem eine periodische Funktion in eine unendliche Summe von Sinus- und Cosinusfunktionen zerlegt werden kann. Wenn ein Signal $s(t)$ folgende Eigenschaften aufweist:

- $s(t)$ ist periodisch mit der Periode T (d.h. $s(t+T) = s(t)$)
- $s(t)$ ist stetig mit endlich vielen Sprungstellen im Intervall $[0, T]$
- $\int_0^\infty |s(t)|dt < \infty$

dann kann $s(t)$ in

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cdot cos(n\omega_0 t) + \sum_{n=1}^{\infty} b_n \cdot sin(n\omega_0 t) \tag{2}$$

zerlegt werden, wobei

$$a_0 = \frac{2}{T} \int_0^T s(t)dt \tag{3}$$

$$a_n = \frac{2}{T} \int_0^T s(t) \cdot cos(n\omega_0 t)dt \tag{4}$$

$$b_n = \frac{2}{T} \int_0^T s(t) \cdot sin(n\omega_0 t)dt \tag{5}$$

$$\omega_0 = \frac{2\pi}{T} \tag{6}$$

sind (vgl. Rieß und Wallraff 2018, S. 19f).

Die Fourier-Koeffizienten a_n und b_n sagen also aus, wie stark einzelne Frequenzen in $s(t)$ vorkommen. Diese Frequenzanteile sind als Output der Fourier-Reihe zu verstehen und können als Merkmale für periodische Signale verwendet werden. a_0 ist der Mittelwert der Funktion $s(t)$.

Kontinuierliche Fourier-Transformation Anders als bei der Fourier Reihe, können mit der verallgemeinerten kontinuierlichen Fourier-Transformation auch aperiodische Signale zerlegt werden, die folgende Eigenschaften aufweisen:

- $s(t)$ ist stückweise stetig
- $\int_{-\infty}^{\infty} |s(t)|dt < \infty$

Die Fourier-Transformation liefert eine transformierte Funktion $\underline{S}(f)$, die als Spektrum des Signals $s(t)$ bezeichnet wird. Es wird also die Dimension Zeit t in die Dimension Frequenz f transformiert. Die Formel für die Fourier-Transformation lautet:

$$\underline{S}(f) = \int_{-\infty}^{\infty} s(t) \cdot e^{-j2\pi ft}dt \tag{7}$$

Wie bereits an der Formel zu erkennen ist, ist das Ergebnis der Fourier-Transformation eine komplexe Funktion mit einem Realteil $\Re(\underline{S}(f))$ und einem Imaginärteil $\Im(\underline{S}(f))$.

Da bei der Weiterverarbeitung in der Regel nur der Normalteil des Spektrums benötigt wird, wird die Fourier-Transformation in der Praxis meistens in der Form

$$\Re(\underline{S}(f)) = \int_{-\infty}^{\infty} s(t) \cdot cos(2\pi ft)dt \tag{8}$$

durchgeführt (vgl. Picard 1996, S. 350f).

Diskrete Fourier-Transformation Bei den bisherigen Analyseverfahren wurden immer kontinuierliche Signale betrachtet. Das heißt, dass für jeden Zeitpunkt t ein Wert $s(t)$ existiert.

Bei der Discrete Fourier Transform (DFT) können diskrete Signale verarbeitet werden, die aus einer Folge von N Werten $(s_0, s_1, \ldots, s_{N-1})$ bestehen, weshalb diese Methode besonders interessant für elektronisch aufgezeichnete Signale ist.

Da keine kontinuierliche Funktion vorliegt, besteht keine Möglichkeit mehr, ein Integral zu bilden, weshalb die Formel mit einer Summe aufgestellt ist:

$$S_k = \frac{1}{N} \sum_{n=0}^{N-1} s_n \cdot e^{-j2\pi \frac{kn}{N}} \tag{9}$$

Der Output der DFT ist wieder eine komplexe Wertereihe, deren Realteil $\Re(S_k)$ und Imaginärteil $\Im(S_k)$ die Amplitude und Phase der k-ten Frequenzkomponente des Signals s_n beschreiben. Der Realteil lässt sich durch die Umformung der Formel in die Form

$$\Re(S_k) = \frac{2}{N} \sum_{n=0}^{N-1} s_n \cdot cos(2\pi \frac{kn}{N}) \tag{10}$$

berechnen (vgl. Smith 1997, S. 567ff).

Um auf die jeweiligen Frequenzanteile des Signals zu schließen, können die Output-Bins S_k mit folgender Formel in die Frequenzen f_k umgerechnet werden:

$$f_k = \frac{k}{N} \cdot f_s \tag{11}$$

wobei f_s die Abtastrate des Signals ist.

Fast Fourier Transformation Die DFT würde sich nach der obigen Formel implementieren und für jedes Signal nutzen lassen. Allerdings ist die Komplexität der DFT mit $O(N^2)$ sehr hoch, weshalb die FFT entwickelt wurde, die den gleichen Output liefert, aber mit einer Komplexität von $O(Nlog(N))$ (vgl. Beucher 2011, S. 338).

Die FFT ist somit eine für Computer optimierte Implementierung der DFT. Eine genaue Erklärung, warum FFT signifikant weniger Berechnungen benötigt, würde für diese Seminararbeit zu sehr ins Detail gehen. Einfach gesagt, nutzt der Algorithmus die Eigenschaft von Sinus-

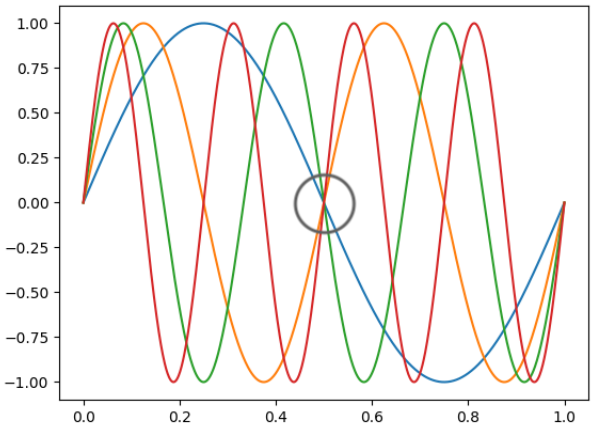


Abb. 2.2: Sinuswellen mit 1hz, 2hz, 3hz, 4hz überlagern sich bei $t = 0,5$
Quelle: Eigene Darstellung

und Kosinus-Wellen, dass Wellen mehrfacher Frequenz an bestimmten Stellen dieselben Werte annehmen und so nur einmal berechnet werden müssen.

Durch diese Eigenschaft und das rekursive Aufteilen des Signals in gerade und ungerade Datenpunkte spart der Algorithmus sehr viele Berechnungen ein (vgl. Oppenheim, Schafer und Buck 1999, S. 643).

2.3.3 Mel Frequency Cepstral Coefficients

verfasst von: Johannes Brandenburger

Die DFT ermöglicht es bereits wesentliche Merkmale wie die Stimmlage und -farbe einer aufgezeichneten Stimme zu bestimmen. Um die gewonnenen Merkmale der Fourier Analyse zu verbessern, werden weitere Berechnungsschritte durchgeführt, sodass die Merkmale in MFCC umgewandelt werden.

Der Vorteil von Merkmalen wie MFCC ist zunächst, dass Audio-Signale mit verschiedenen Längen und Abtastraten besser miteinander verglichen werden können. Das liegt daran, dass bei einer Analyse der MFCC eine fest definierbare Anzahl an Koeffizienten generiert werden. Die Anzahl hängt dabei, anders als bei einer Fouriertransformation, nicht von der Frame-Länge und Abtastrate ab. Zudem sind die MFCCs mehr auf die menschliche Wahrnehmung von Tönen abgestimmt und robuster gegenüber Hintergrundrauschen.

Um die MFCC zu berechnen sind die folgenden vier Schritte notwendig (vgl. Logan 2000, S. 2):

1. **DFT**
   Bestimmung der Frequenzkomponenten

2. **Logarithmierung**
   Näher an menschlicher Wahrnehmung
3. **Abbildung auf die Mel-Skala**
   Zusammenfassung von Frequenzen zu Mel-Bändern mithilfe von Dreiecksfiltern
4. **Diskrete Kosinustransformation**
   Ähnlich zur inversen Fourier-Transformation

Durch die Abbildung der ermittelten Frequenzen auf die Mel-Skala orientiert sich das Verfahren somit an der menschlichen Stimmwahrnehmung, welche bereits im vorangehenden Kapitel (2.2.2) erläutert wurde. Das macht MFCC zu einem guten Merkmal, wenn es darum geht, von einem Stimmsignal auf dessen Sprecher zu schließen.

2.3.4 Linear Predicitve Coding

verfasst von: Henry Schuler

Die Grundlage von LPC bildet das Autoregression (AR) Modell. Die AR basiert auf dem Konzept der multiplen Regression und wird auf zeitlich veränderliche Prozesse angewandt. Dabei wird eine Kriteriumsvariable unter Betrachtung von einer beliebigen Anzahl an Prädiktorvariablen vorhergesagt (vgl. Canela, Alegre und Ibarra 2019, S. 37-38). Im speziellen Fall der AR handelt es sich bei den Prädiktorvariablen um vorhergehende Werte des Prozesses. Ein AR Modell sagt somit den Wert zu einem Zeitpunkt n, basierend auf p Vorgängerwerten des Prozesses voraus. Es gilt somit der in Formel 12 dargestellte Zusammenhang, wobei $\hat{s}_n$ den vorausgesagten Wert, s_{n-k} die vorhergehenden Werte, a_k die Regressionsgewichte und p die Anzahl an verwendeten Vorgängerwerten darstellt (Atal 1974, S. 1304).

$$\hat{s}_n = \sum_{k=1}^{p} s_{n-k} a_k \tag{12}$$

Zur Bestimmung der Regressionsgewichte wurden verschiedene rekursive Verfahren entwickelt. Neben der Yule-Walker Methode stellt der Burg Algorithmus eine beliebte Alternative dar, welcher in Marple, S. 443 beschrieben ist.

Bei dem Verfahren LPC wird der Ansatz verfolgt, Rückschlüsse von dem akustischen Signal auf die Stimmerzeugung zu ziehen. Dazu wird ein AR-Filter verwendet, um ein vereinfachtes Modell des menschlichen Stimmtrakts zu erstellen. Das vereinfachte Modell enthält zunächst ein Rausch-Signal (z.B.: white noise), welches die Stimmbildung repräsentiert. Anschließend wird auf dieses Rauschen der AR-Filter angewendet, welcher die durch Resonanzen im Vokaltrakt entstehenden Frequenzänderungen abbildet. Die Regressionsgewichte a_k des AR-Filters entsprechen dabei den LPC-Koeffizienten.

Die LPC-Koeffizienten erfassen somit die Resonanzeigenschaften des Signals, wodurch Rückschlüsse auf die Formanten gezogen werden können. Da die Struktur der Formanten sprecherspezifisch ist, kann der Sprecher über die LPC-Koeffizienten identifiziert werden (vgl. Zulfiqar u. a. 2010, S. 117).

Zur Berechnung der LPC-Koeffizienten wird zunächst die selbe Annahme wie in Kapitel 2.3.1 getroffen, dass sich die Form des Vokaltrakts und das in den Stimmritzen erzeugte Signal über den betrachteten Zeitraum nicht verändert (vgl. Atal 1974, S. 1304). Somit lassen sich die Koeffizienten des AR-Filters mittels des Burg-Algorithmus berechnen.

2.4 Künstliche Intelligenz

verfasst von: Lukas Braun

Künstliche Intelligenz (KI) hat in den letzten Jahren stark an Aufmerksamkeit gewonnen und gilt als eine der wichtigsten Technologien des 21. Jahrhunderts. Spätestens seit der Veröffentlichung und dem Erfolg durch den Chatbot ChatGPT am 30. November 2022 sind die Möglichkeiten von KI den meisten Menschen bekannt. ChatGPT hat innerhalb von fünf Tagen die Schwelle von einer Million Nutzer erreicht. Spotify oder Facebook zum Vergleich haben hierfür 5 beziehungsweise 10 Monate benötigt (vgl. Janson 2023).

Die Statistik die im Jahr 2022 von der International Data Corperation erstellt wurde prognostiziert für die Anwendungsfelder Hardware, Software und IT-Services im Jahr 2024 einen weltweiten Umsatz von 554,3 Milliarden US-Dollar. Dies entspricht einer Steigerung von 171 Milliarden US-Dollar (44,6 %) im Vergleich zum Jahr 2021 (vgl. IDC 2022). Dies unterstreicht noch einmal die Relevanz von KI im aktuellen Zeitalter. Doch was ist KI und wie ist Sie für diese Arbeit von Relevanz, diese Fragen werden im Folgenden beantwortet.

Die Idee einer KI ist bereits seit Mitte des 20. Jahrhunderts existent. 1955 definierte John McCarthy, einer der Pioniere der KI, als Erster den Begriff der KI wie folgt:

> „Ziel der KI ist es, Maschinen zu entwickeln, die sich verhalten, als verfügten sie über Intelligenz.“ (Ertel 2016, S. 1)

Aus heutiger Sicht ist diese Definition nicht mehr ausreichend. Eine weit verbreitete Definition stammt von Elaine Rich welche sagt, dass sich KI mit der Entwicklung von Computeralgorithmen befasse, für Probleme, die Menschen im Moment noch besser lösen können (vgl. Rich 1983).

Aufgrund der Entwicklung seit Mitte des 20. Jahrhunderts gliedert sich das Thema KI in viele Teilgebiete. Die verschiedenen Teilgebiete sind in der Abbildung 2.3 dargestellt.

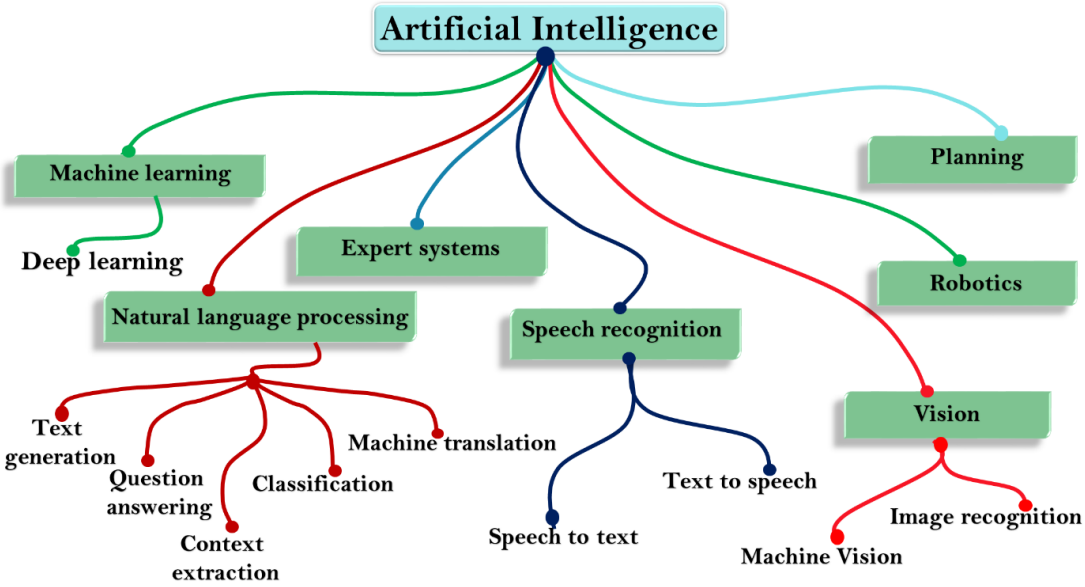


Abb. 2.3: Teilgebiete Künstliche Intelligenz (vgl. *Subsets of AI - Javatpoint* o. D.)

Insbesondere das Thema Machine Learning (ML) hat in den letzten zehn Jahren viel Beachtung gefunden. ML spielt für diese Studienarbeit eine entscheidende Rolle und wird deshalb im folgenden genauer betrachtet.

ML ist nicht neu, bereits in den 1950er Jahren wurden von Arthur Lee Samuels die ersten ML Programme entwickelt (vgl. Judith Hurwitz und Daniel Kirsch 2018, S. 5). Der Aufschwung von ML im letzten Jahrzehnt ist auf steigende und bezahlbare Rechenleistung bzw. Speichergrößen zurückzuführen. ML beschreibt den Prozess bei dem Computeralgorithmen trainiert werden, um ihre Leistung bei einer bestimmten Aufgabe zu verbessern, ohne dass dies explizit programmiert werden muss. Hierzu werden dem Computer Daten bereitgestellt aus welchen er lernt. Der Computer kann anschließend die aus den Daten gewonnenen Erkenntnisse nutzen, um Vorhersagen für neue Daten zu treffen (vgl. Judith Hurwitz und Daniel Kirsch 2018, S. 4). Die Vorhersagen stimmen nicht immer zu 100 %, die Vorhersagegenauigkeit hängt von verschiedenen Faktoren ab, welche je nach Anwendungsfall variieren.

ML lässt sich in die drei Kategorien überwachtes Lernen, unüberwachtes Lernen und verstärkendes Lernen gliedern. Beim überwachten Lernen besteht der Trainingsdatensatz aus Eingabewerten und gewünschten Ausgabewerten. Der Algorithmus generiert eine Abbildungsfunktion, die die Eingabewerte auf die Ausgabewerte abbildet. Der Computer wird sozusagen unterrichtet, Vorhersagen für zukünftige Daten auf Grundlage von Bestandsdaten zu erstellen (vgl Thakkar 2019, S. 11). Im Gegensatz dazu besteht der Trainingsdatensatz beim unüberwachten Lernen nur aus unmarkierten Eingabewerten. Das Ziel von unüberwachtem Lernen besteht darin Muster oder Strukturen zu erkennen. Der Computer unterrichtet sich sozusagen selbst (vgl. Thakkar 2019, S. 11-12; vgl. Etaati 2019, S. 6). Verstärkendes Lernen ist ein verhaltensbasiertes Lernmo-

dell. Hierzu interagiert der Lernalgorithmus mit der Umgebung und erhält Belohnungen, wenn er die gewünschten Ergebnisse erzielt. Der Computer erlernt so das ideale Verhalten innerhalb der Umgebung (vgl Thakkar 2019, S. 12).

2.5 Neuronale Netze

verfasst von: Lukas Braun

In engem Zusammenhang zu Machine Learning stehen die sogenannten Neuronalen Netze. Ein NN ist ein maschinelles Modell, das auf dem Vorbild der Abläufe im Gehirn von Mensch und Tier basiert. Durch geeignete mathematische Operationen wird versucht die Art und Weise abzubilden wie das menschliche Gehirn Probleme angeht beziehungsweise Zusammenhänge aus beobachteten Daten erkennt (vgl. Backhaus u. a. 2016, S. 604; vgl. Judith Hurwitz und Daniel Kirsch 2018, S. 31). Aus den vorhin genannten Lernalgorithmen (überwachtes, unüberwachtes, und verstärkendes Lernen) ergibt sich als Ergebnis ein neuronales Netz, welches das erlernte Wissen in seiner Struktur speichert (vgl. Guresen und Kayakutlu 2011, S. 427).

Ein neuronales Netz ist prinzipiell in Schichten aufgebaut. In jedem neuronalen Netz gibt es eine Eingabe- und Ausgabeschicht. Zwischen der Eingabe- und Ausgabeschicht liegen mehrere versteckte Schichten (hidden layers). Jede Schicht besteht aus mehreren Neuronen (Knoten), welche über gewichtete Kanten miteinander verbunden sind (vgl. Sonnet 2022, S. 26). Die Informationen werden durch die Neuronen in der Eingabeschicht aufgenommen und durch die Neuronen in der Ausgabeschicht ausgegeben. Die Neuronen in der versteckten Schicht verarbeiten die Informationen. Ein grundsätzlicher Aufbau eines neuronalen Netzes ist in Abb. 2.4 dargestellt.

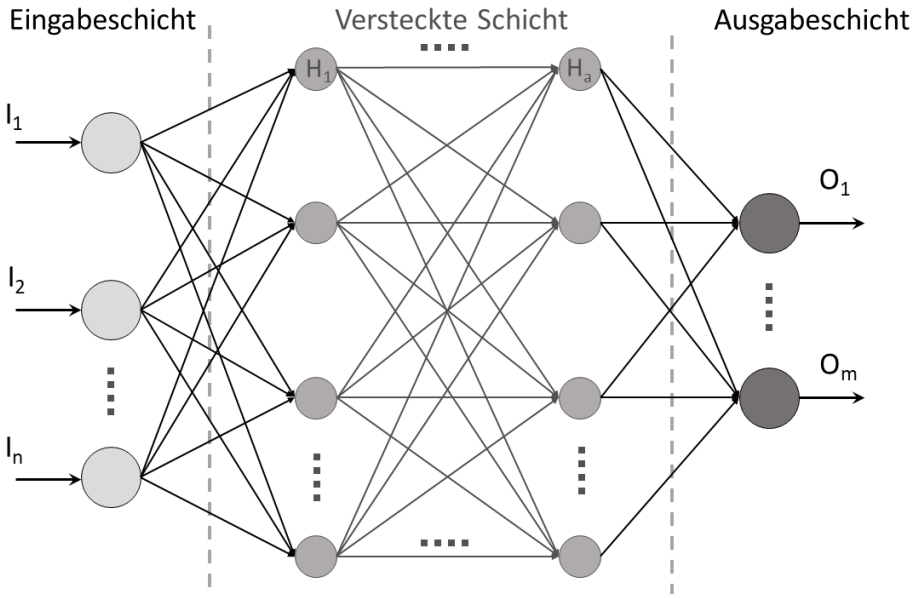


Abb. 2.4: Grundsätzlicher Aufbau eines neuronalen Netzes (Daniel John 2022)

Ein Neuron ermittelt seine Aktivierung aufgrund der Eingaben der vorherigen Neuronen und gibt daraus aufbauend einen Wert aus. Dieser Wert kann durch die Gewichte auf den Kanten beeinflusst werden. Eine Aktivierungsfunktion entscheidet dabei, ob das Neuron feuert, sprich seinen errechneten Wert weitergibt. Dies gibt Aufschluss über die Relevanz der vorangegangenen Werte. Somit können durch die Gewichte der Kanten und den Wert der Neuronen Informationen abgebildet werden (vgl. Sonnet 2022, S. 27).

Für das Training von neuronalen Netzen werden viele klassifizierte Daten benötigt, welche als Ausgangspunkt dienen. Das Training läuft iterativ in Schleifen, sogenannte Epochen ab. Nach jedem Durchlauf werden mithilfe des Feedbacks der Ausgabeschicht die Gewichte der Kanten und der Wert der Neuronen angepasst. Zu Beginn werden die Kanten-Gewichte und der Wert eines Neurons zufällig gesetzt. Dies wirkt sich auf das Ergebnis des gesamten Trainingsprozesses aus, da die zufällig gesetzten Werte so ungünstig sein können, dass das Netz während des Durchlaufs die Gewichte nicht ausreichend anpassen kann. Nach Abschluss des Trainings kann das Netz bewertet werden. Hierbei werden abhängig von der gewählten Technologie unterschiedlich Parameter ausgegeben. Grundsätzlich kann ein Netz immer beurteilt werden, indem bereits klassifizierte Daten durch das Netz klassifiziert werden und die Ergebnisse anschließend mit dem tatsächlichen Ergebnis verglichen werden. Im Regelfall werden hierbei Daten verwendet, welche zwar aus dem gleichen Datensatz stammen, jedoch nicht für das Training genutzt wurden (vgl. Marcel Mikl 2018).

2.6 Gefahrenanalyse

verfasst von: Lukas Braun

Eine Möglichkeit der Beurteilung von sicherheitskritischen Systemen stellt die Gefahrenanalyse (Threat Analyse) dar. Dabei wird durch verschiedene Herangehensweisen versucht, möglichst viele potenzielle Gefahren und Angriffsstellen des Systems zu ermitteln und anschließend zu bewerten. Als Resultat ergibt sich eine klare Einschätzung der Gefahrensituation, wodurch eine Aussage über die Sicherheit des entwickelten Systems getroffen werden kann.

In diesem Kapitel werden zwei Verfahren zur Ermittlung und Beurteilung von Gefahren vorgestellt. STRIDE bietet einen grundlegenden Ansatz um den Ermittlungsprozess zu unterstützen. Abschließend ermöglicht DREAD eine Bewertung der einzelnen Gefahren.

2.6.1 STRIDE

Das Akronym STRIDE steht für Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service und Elevation of Privilege. Es beschreiben somit verschiedene Kategorien von Gefahren. Im Ermittlungsprozess liefern diese Kategorien dabei eine Eingrenzung der Analyse auf eine bestimmte Art von Gefahren, wodurch ein gezielteres Vorgehen ermöglicht wird. Beispielsweise können verschiedene Gefahrensammlungen herangezogen werden, welche gängige Gefahren der jeweiligen Kategorien auflisten.

Die Zuteilung einer Gefahr zu exakt einer STRIDE Kategorie ist dabei nicht immer möglich und von STRIDE nicht vorhergesehen. Für die anschließende Evaluierung der Gefahren, sowie die Entwicklung möglicher Gegenmaßnahmen, wird durch die Zuteilung einer Gefahr zu mehreren Kategorien der Horizont gegenüber möglichen Lösungsansätzen erweitert, wodurch die Gefahr besser eingeschätzt und verhindert werden kann (vgl. Shostack 2014, S. 61-64).

2.6.2 DREAD

Die DREAD Methode ermöglicht eine feine Beurteilung der Gefahren. Dafür wird jede Gefahr in fünf verschiedenen Kategorien bewertet. Die Kategorien lauten Damage, Reproducibility, Exploitability, Affected users und Discoverability (DREAD).

Für jede der fünf Kategorien werden dabei zwischen einem und zehn Punkte vergeben. Die Bewertungen der Kategorien werden anschließend aufsummiert und der Mittelwert gebildet. Anhand dieses Mittelwertes kann nun eine priorisierte Liste erstellt werden, wobei ein hoher Wert eine hohe Gefahr bedeutet. Ausgehend von dieser Liste kann somit entschieden werden, welche Gefahren priorisiert behandelt werden müssen (vgl. DOMARS 2023).

3 Stand der Technik

verfasst von: Johannes Brandenburger

Im vorliegenden Kapitel wird der aktuelle Stand im Bereich der Sprechererkkenung detailliert untersucht und dargestellt. Dazu werden mehrere Arbeiten und Bücher aus anderen wissenschaftlichen Forschungen verglichen.

Für die Analyse von Sprachsignalen werden zunächst Merkmale (Features) des Signales benötigt, welche dann weiterverarbeitet werden können. Für die Weiterverarbeitung wird dann ein Klassifikations-Model benötigt, das aufgrund der Merkmale eine Aussage über den Sprecher treffen kann.

In der vorhandenen Literatur finden sich mehrere Bücher und Fachartikel, die sich mit diesem Thema bereits auseinandergesetzt haben. Dieses Kapitel zeigt, welche Merkmale von verschiedene vorangehende Arbeiten verwendet werden und wie effektiv diese im direkten Vergleich sind. Dabei wird ebenfalls auf die verwendeten Klassifikationsmodelle eingegangen.

Qi (Peter) Li, ein Elektrotechnik-Doktor und Experte für Sprecherauthentifikation, untersucht in seinem Buch „Speaker Authentification“, relevante Merkmale für die Sprechererkennung (Li 2012).

Er gelangte zu folgenden Erkenntnissen: MFCC und LPC sind sehr gute Merkmale zur Identifizierung eines Sprechers in ruhigen Umgebungen, aber ihre Leistung nimmt in Umgebungen mit Hintergrundrauschen ab (vgl. Li 2012, S. 136). Die ersten und zweiten Ableitungen von MFCC werden oft auch noch genutzt, tragen laut Qi (Peter) Li allerdings nicht zu einer Verbesserung des Authentifizierungssystems bei (vgl. Li 2012, S. 143).

Neben den Hauptmerkmalen MFCC und LPC werden zusätzlich Gammatone-Merkmale bewertet, die das menschliche Hören modellieren sollen. Diese können auch relevante Daten für eine Sprecherauthentifikation liefern (vgl. Li 2012, S. 111, 117).

Qi (Peter) Li hat im Zuge seiner Arbeit auch ein ganz neues Merkmal (CFCC[1]) entwickelt, das ebenfalls das menschliche Gehör nachahmen soll und in seinen Untersuchungen am besten abgeschnitten hat (vgl. Li 2012, S. 135).

Das Buch „Advanced Topics In Biometrics“ von Haizhou Li, Kar-ann Toh und Liyuan Li bezeichnet MFCC als das relevanteste Merkmal für die Sprecherauthentifikation, obwohl es ursprünglich für die Spracherkennung entwickelt wurde (vgl. Li, Toh und Li 2011, S. 7, 51).

[1] Cochlear filter cepstral coefficients

Aber auch Linear Prediction Cepstral Coefficient (LPCC), PLP[2] und FM[3] werden als effektive Merkmale genannt und können zusammen mit MFCC sehr gute Ergebnisse erzielen (vgl. Li, Toh und Li 2011, S. 6, 67).

Das Paper „Multilingual Speaker Recognition Using Neural Network“ protokolliert gute Ergebnisse durch die Extraktion von LPC und der Nutzung eines neuronalen Netzes (vgl. Kumar Rajeev u. a. 2009, S. 9).

Auch eine ältere Untersuchung von Reynolds and Rose ergab, dass LPC in einer ruhigen Umgebung effektiv ist, MFCC allerdings robuster ist und daher von den Autoren bevorzugt wurde (vgl. Reynolds und Rose 1995, S. 2f). Als Klassifikations-Model wurden hierbei Gaussian Mixture Models (GMM) genutzt und als effektiv erachtet. Auch neuronale Netze und Hidden Markov Modelle werden als mögliche Klassifikatoren erwähnt, allerdings nicht getestet (vgl. Reynolds und Rose 1995, S. 2f, 11).

Ein weiters Paper, welches sich mit Sprecherauthentifikation auf performance-kritischen Systemen beschäftigt, nennt LPCC als bestes Merkmal (vgl. Thullier, Bouchard und Menelas 2017, S. 7). Hierbei wurde ein Naive-Bayes-Klassifikator verwendet, welcher gute Ergebnisse erzielte (vgl. Thullier, Bouchard und Menelas 2017, S. 18f).

Fatma Zohra Chelali1 und Amar Djeradi geben in ihrem Artikel zur Sprechererkennung zunächst MFCC und PLP als relevante Merkmale an (vgl. Chelali und Djeradi 2017, S. 276). Allerdings vergleichen sie auch verschiedene Kombinationen von MFCC, den Ableitungen Delta Mel Frequency Cepstral Coefficients (dMFCC) und Delta Delta Mel Frequency Cepstral Coefficients (ddMFCC), DWT[4] und LPC. Bei diesen Untersuchungen hat die Kombination von MFCC, dMFCC, DWT und LPC am besten abgeschnitten. ddMFCC konnte das System nicht verbessern (vgl. Chelali und Djeradi 2017, S. 276, 739). Hier wurde ein neuronales Netz als Klassifikations-Model verwendet (vgl. Chelali und Djeradi 2017, S. 735).

Ein Paper der Tokyo Institute of Technology hatte ebenfalls mit MFCC und LPC sehr gute Ergebnisse. Durch das Hinzufügen von ZCR[5] konnten diese noch verbessert werden. Die Versuche wurden hier mit einem neuronalen Netz und einer Support-Vector-Machine durchgeführt. Die beiden Klassifizierer haben sich dabei in ihrer Effektivität kaum unterschieden, mit dem Neuralen Netz wurden allerdings die besten Ergebnisse erzielt (vgl. Neha Chauhan, Dongju Li und Tsuyoshi Isshiki 2019, S. 4).

[2] Perceptual Linear Prediction
[3] Frequency Modulation
[4] Discrete Wavelet Transform
[5] Zero-crossing Rate

Buch/Artikel	**MFCC**	**LPC**	**LPCC**	**ΔMFCC**	**ΔΔMFCC**	DWT (discrete wavelet transform)	GFCC (gammatone-frequency cepstral coefficients)	ZCR (zero-crossing-rate)	PLP (Perceptual linear predictive)	FM (Frequency modulation)
Speaker Authentification von Qi Li	✔	✔		✘	✘		✔			
Advanced Topics In Biometrics von Haizhou Li, …	✔		✔						✔	✔
FM features for automatic forensic speaker recognition von Thiruvaran, …	✔									✔
MULTILINGUAL SPEAKER RECOGNITION USING NEURAL NETWORK von Kumar Rajeev, …		✔								
Robust text-independent speaker identification using Gaussian mixture speaker models von Reynolds and Rose	✔	✔								
A Text-Independent Speaker Authentication System for Mobile Devices von Florentin Thullier, …			✔							
Text dependant speaker recognition using MFCC, LPC and DWT von Chelali und Amar Djeradi	✔	✔		✔	✘	✔				
Speaker Recognition Using LPC, MFCC, ZCR Features with ANN and SVM Classifier for Large Input Database Neha Chauhan, …	✔	✔						✔		

Abb. 3.1: Literaturbewertung zu Features

Die Abbildung 3.1 zeigt nochmals eine tabellarische Übersicht der verschiedenen Merkmale, die in den analysierten Büchern und Artikeln verwendet wurden und ihre bewertete Effektivität.

Die analysierten Bücher und Artikel legen nahe, dass eine Kombination von MFCC, LPC und LPCC sinnvoll für eine Sprecherauthentifikation ist. Zwar hatten einige Paper noch vereinzelt andere Features als effektiv erachtet, die grobe Schnittmenge besteht jedoch aus den oben genannten Features.

Als Klassifikationsmodelle wurden in den meisten Experimenten neuronale Netze verwendet, aber auch GMM, HMN und SVM wurden als effektiv erachtet.

4 Konzeption

verfasst von: Johannes Brandenburger, Lukas Braun, Henry Schuler

Das vorliegende Kapitel behandelt die Konzeption des Versuchssystems und des Demosystems. Dazu erfolgt zuvor ein grobes Konzept für ein generisches Sprecherauthentifikationssystem und die Festlegung der Anforderungen an das Demosystem.

4.1 Allgemeiner System-Aufbau

verfasst von: Johannes Brandenburger, Lukas Braun

Ein allgemeiner, generischer Aufbau für ein Sprecherauthentifikationssystem ist in Abbildung 4.1 dargestellt.

Das Konzept wurde weitgehend eigens entwickelt, orientiert sich aber an Ansätzen von Qi (Peter) Li (vgl. Li 2012, S. 7).

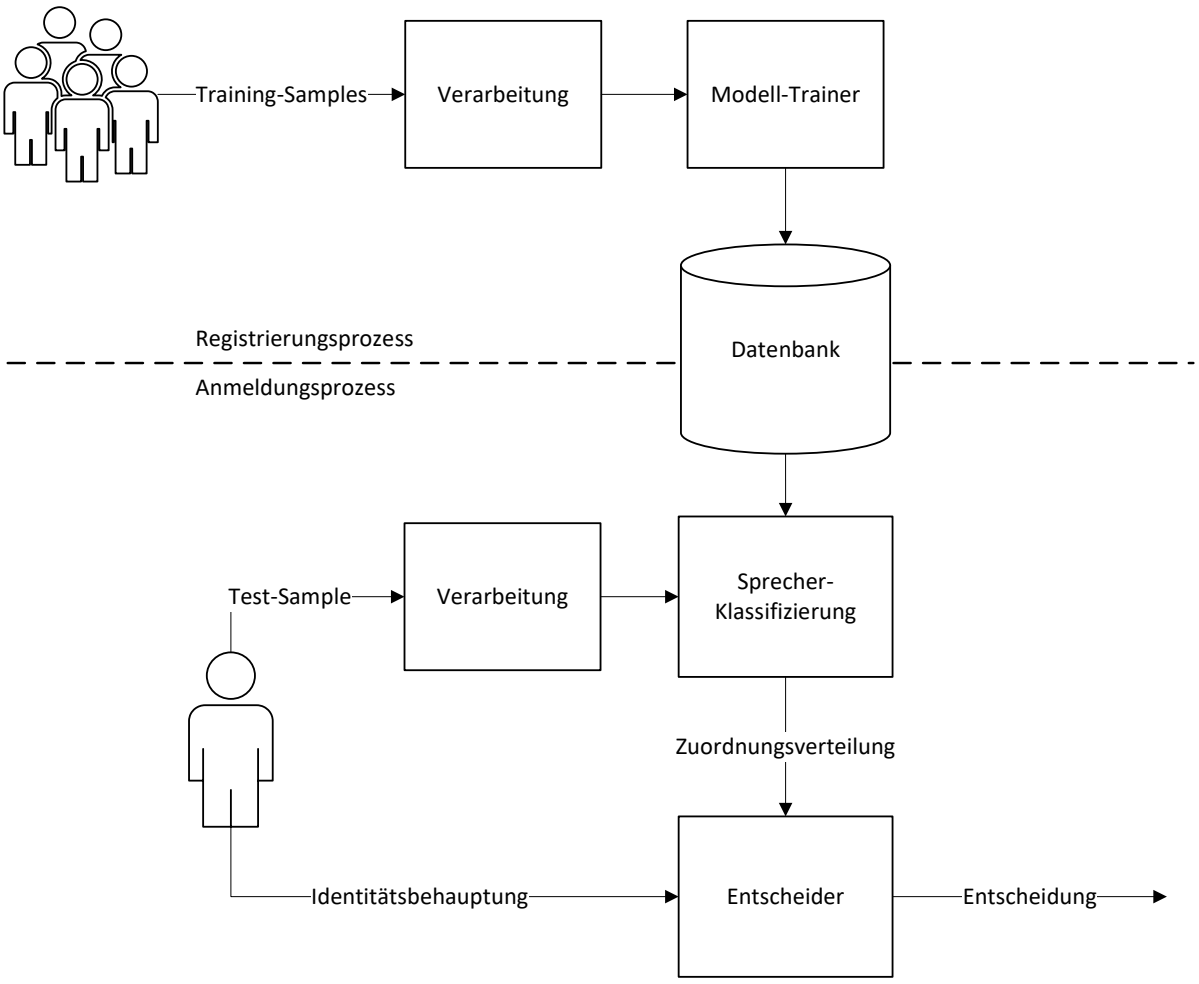


Abb. 4.1: Informelles Komponentenkonzept für ein Sprecherauthentifikationssystem

Das System lässt sich in zwei Prozesse aufteilen: dem Registierungs- oder Trainingsprozess und dem Anmeldungs- oder Testprozess. Die verschiedenen Komponenten der beiden Prozesse werden im folgenden Abschnitt kurz erklärt.

Am Anfang des Trainingsprozesses steht die Benutzergruppe. Diese beinhaltet alle Personen, die das System benutzen wollen, beziehungsweise sich authentifizieren möchten. Die Benutzer geben Trainings-Samples in eine Verarbeitungs-Komponente, welche die Audio-Daten so aufbereitet, dass sie für ein Modell verwendet werden können. Die Modell-Trainer-Komponente generiert dann über mathematische Methoden eine Art Datenbank (zum Beispiel in Form eines neuronalen Netzes). Dies ist die Schnittstelle zum Anmeldungsprozess. In diesem möchte sich ein einzelner Benutzer mit dem System authentifizieren. Dazu wird zunächst ein Test-Sample über eine Verarbeitungs-Komponente in die Sprecher-Klassifizierungs-Komponente gegeben, welche unter Verwendung der Datenbank-Komponente eine Zuordnungsverteilung erstellt. Außerdem gibt der Nutzer eine Identitätsbehauptung ab, welche zusammen mit der Zuordnungsverteilung von einer Entscheidungs-Komponente verarbeitet wird. Hieraus resultiert eine Entscheidung, ob die Identitätsbehauptung akzeptiert wird oder nicht.

Dieser Ablauf stellt die Grundlage für die Konzeption der nachfolgenden Systeme dar.

4.2 Festlegung der Systemanforderungen

verfasst von: Johannes Brandenburger

Die grundlegende praktische Anforderung an diese Studienarbeit ist es, ein Sprecherauthentifikationssystem zu entwickeln, welches über eine Webapplikation bedient werden kann. Dazu müssen geeignete Stimmmerkmale zur Benutzerauthentifizierung evaluiert werden. Da diese Anforderungsbeschreibung noch viel Spielraum über den Funktionsumfang dieses Systems lässt, werden in diesem Kapitel die Anforderungen an das System genauer spezifiziert.

- Das System soll lediglich ein Demosystem darstellen, welches die Machbarkeit eines Sprecherauthentifikationssystem aufzeigt. Es soll nicht in einem produktiven Umfeld eingesetzt werden können.
- Das System soll eine text-unabhängige Authentifizierung implementieren, da der Fokus des Systems auf der Identifikation von Stimmmerkmalen zur Authentifizierung von Benutzern liegt und dies unabhängig vom gesprochenen Text erfolgen soll.
- Der Datensatz für das System ist zu Beginn festgelegt. Das heißt, es können sich nur 20 bereits bekannte Sprecher authentifizieren. Zudem können keine neuen Sprecher registriert werden, wodurch das Sprecherauthentifikationssystem keine dynamische Erweiterung um Sprecher bereitstellen muss.

- Das System soll eine Open-Set-Implementierung verwenden. Dadurch wird sichergestellt, dass ein Benutzer erst mit einer fest definierten Sicherheit authentifiziert wird.
- Zur Authentifizierung eines Sprechers wird ein 15-sekündiger Audio-Clip an das System übermittelt. Der Audio-Clip ist dem System bisher unbekannt, aber in gleichen Bedingungen aufgenommen, wie die Clips des Trainingsprozesses.
- Das Sprecherauthentifikationssystem sollte Teil des Back-Ends sein. Die Bedienung erfolgt über eine Webapplikation, die mit dem System über eine Schnittstelle kommuniziert.
- In der Web-Oberfläche des Demosystems soll der zu authentifizierende Sprecher (einer der 20) und ein Verifikations-Clip ausgewählt werden können. Es soll für jeden Sprecher mindestens 5 Clips zur Auswahl geben.
- So soll der Nutzer des Demosystems testen können, was passiert, wenn ein zu dem zu authentifizierenden Sprecher passender bzw. unpassender Verifikations-Clip ins System gegeben wird.
- Das System soll dem Nutzer dann Informationen über die Identifikations-Verteilung (zu welchem Sprecher passt der Clip zu welchem Prozentsatz) und den Authentifizierungs-Status („erfolgreich“/„nicht erfolgreich“) als Rückmeldung darstellen.

4.3 Konzept Versuchssystem

verfasst von: Johannes Brandenburger, Lukas Braun

Die Konzeption des Versuchssystems unterteilt sich in mehrere Unterkapitel. Zunächst wird die allgemeine Systemidee vorgestellt. Anschließend wird auf die in dieser Arbeit evaluierten Feature Kombinationen eingegangen. Daraufhin werden die Anforderungen an den Datensatz konkretisiert, woraufhin anschließend noch einmal auf den allgemeinen Verarbeitungsprozess der Audiodateien eingegangen wird. Abgeschlossen wird das Kapitel mit einer Beschreibung der Klassifikation und Evaluierung, sowie der spezifischen Softwarearchitektur des Versuchssystems.

4.3.1 Systemidee

verfasst von: Johannes Brandenburger, Lukas Braun

Aus der Literaturrecherche in Kapitel 3 gehen verschiedene Stimmmerkmale zur Benutzerauthentifizierung hervor. Die Ergebnisse der dargestellten Untersuchungen unterscheiden sich darin, inwieweit die unterschiedlichen Stimmmerkmale die Stimme repräsentieren bzw. zuverlässig für eine korrekte Authentifizierung sind.

Da die Features LPC, LPCC, MFCC und dMFCC die grobe Schnittmenge der Untersuchungen darstellt, werden diese in einer eigens durchgeführten Versuchsreihe getestet und evaluiert. Hier-

für wird ein System entworfen, das die Merkmale aus einem Datenset extrahiert und verschiedene Kombinationen vergleicht. Das Ziel dieses Systems ist, eine ideale Kombination aus Features herauszufinden. Ein grober Ablaufplan des Systems ist in Abbildung 4.2 dargestellt.

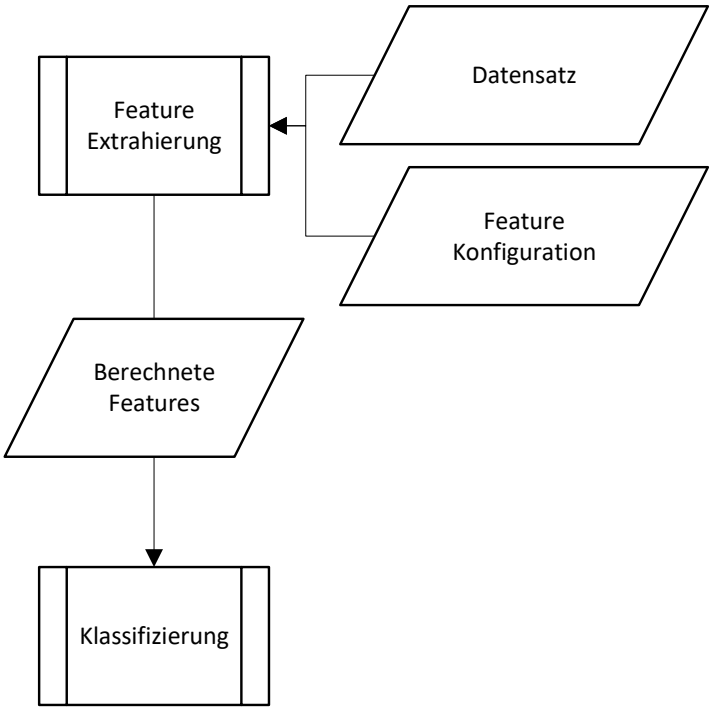


Abb. 4.2: Ablauf Versuchssystem

4.3.2 Feature Kombinationen

verfasst von: Johannes Brandenburger, Lukas Braun

Um die Kombinationen zu generieren, werden zunächst die verschiedenen Möglichkeiten definiert und anschließend miteinander kombiniert, was zu über 500 verschiedenen Konfigurationen führt. Hierzu werden zusätzlich zu den oben genannten Features auch die Anzahl und Länge der Frames definiert. Für die Anzahl der Frames wurden 10000 und 15000 Frames definiert, für die Länge der Frames 400 und 600 Samples. Diese Werte berechnen sich aus vorherigen Versuchen, der verwendeten Abtastrate und der gewünschten Trainingsclip-Länge. Für die Features werden die Anzahl der Features pro Frame auf 13 und 20 festgelegt. Der Wert 13 ist aus der Literatur abgeleitet (vgl. Valerio Velardo 2020, S. 69). Der Wert 20 ist selbstständig ergänzt, um einen Vergleich zu ermöglichen. Da die Relevanz von dMFCC am geringsten ist (s. Abb. 3.1), wird hier auf den zweiten Wert verzichtet, um die Anzahl der Konfigurationen zu reduzieren. Die vorhin genannten Werte sind übersichtlich in der nachfolgenden Tabelle dargestellt.

Bezeichner	**Werte**
Anzahl der Frames	10000, 15000
Länge der Frames	400, 600
LPC	13, 20
MFCC	13, 20
LPCC	13, 20
delta MFCC	13

Tab. 4.1: Übersicht Featurekombinationen

4.3.3 Datensatz

verfasst von: Lukas Braun

Um die unterschiedlichen Kombinationen an Features zu vergleichen, wird ein Datensatz benötigt, der mindestens 20 verschiedene Sprecher mit mindestens ca. 15 min unterschiedlichem Audio-Material enthält. Die Audio-Clips müssen unter gleichen Bedingungen aufgenommen sein und in kompressionslosem Audio-Format vorliegen. Der Datensatz wird aufgeteilt in Training- und Testdaten.

4.3.4 Vorverarbeitung und Feature-Extraktion

verfasst von: Johannes Brandenburger

In der Feature-Extraktion werden die Stimmmerkmale aus den Trainingsdaten extrahiert. Hierzu müssen die Audiosignale zunächst vorverarbeitet werden (siehe Kapitel 2.3.1). In welcher Form die Merkmale extrahiert werden hängt dabei von der aktuellen Konfiguration ab.

Abbildung 4.3 zeigt den Ablauf von diesem Teil des Versuchssystems mit Bezug auf die im Klassendiagramm dargestellten Klassen und Methoden (Abbildung 4.4).

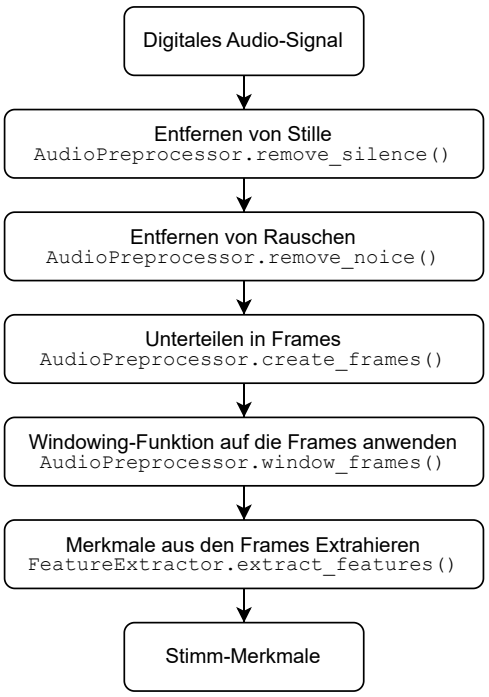


Abb. 4.3: Ablaufdiagramm Vorverarbeitung und Feature-Extraktion

Näher wird dieser Ablauf in dem dazugehörigen Umsetzungs-Kapitel (5.2.3) beschrieben.

4.3.5 Klassifikation und Evaluierung

verfasst von: Lukas Braun

Die Literaturrecherche hat ergeben, dass sich neuronale Netze gut als Klassifikationsmodell eignen (siehe Kapitel 3). Deshalb werden bei der Klassifikation die berechneten Features aus dem Schritt zuvor genutzt, um neuronale Netze zu trainieren, sodass die Feature-Kombinationen bewertet werden können. Hierbei muss das Netz mehrmals durchlaufen werden, da zu Beginn des Trainings die Gewichtung zufällig gesetzt wird (siehe Kapitel 2.5). Anschließend werden die neuronalen Netze mit Testdaten getestet, die nicht in den Trainingsdaten enthalten sind. Die Auswertung des neuronalen Netzes wird gespeichert und nach Berechnung aller Kombinationen ausgewertet. Somit soll die beste Kombination gefunden und das damit trainierte neuronale Netz für das Demosystem verwendet werden.

4.3.6 Software Architektur

verfasst von: Johannes Brandenburger, Lukas Braun

Um eine solide Grundlage für die Entwicklung zu schaffen, wird im Folgenden die Softwarearchitektur für das System entwickelt. Hierzu wird zunächst basierend auf den vorangegangenen

Definitionen ein Klassendiagramm erstellt. In einem Folgeschritt wird der Ablauf in einem Sequenzdiagramm dargestellt.

Das Klassendiagramm ist in Abbildung 4.4 dargestellt. Es zeigt die Aufteilung des Systems in mehrere Komponenten (Klassen). Die wichtigsten Klassen werden nachfolgend kurz beschrieben.

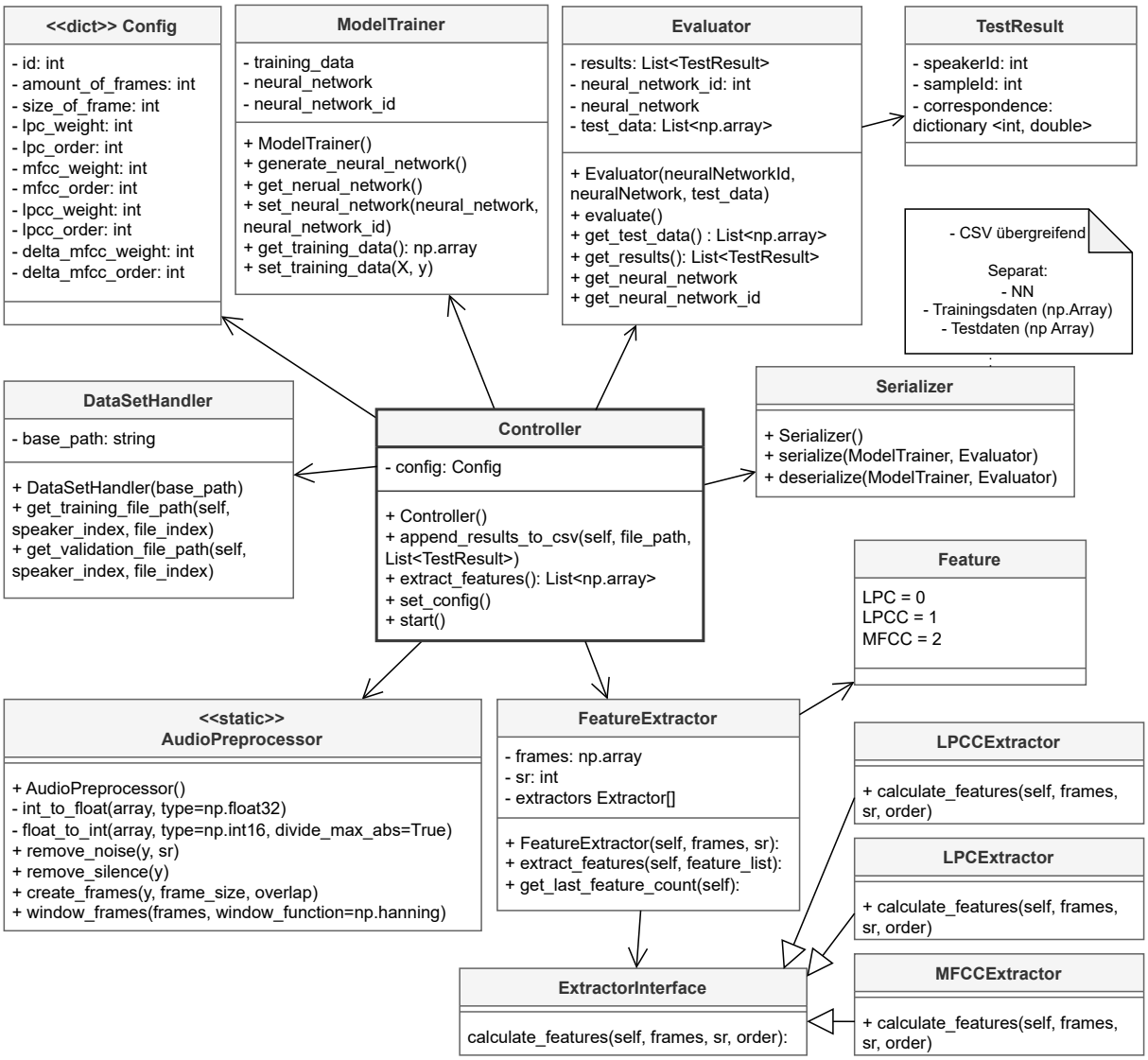


Abb. 4.4: Klassendiagramm Versuchssystem

Controller

Der Controller stellt das zentrale Element dar, von hier werden alle Abläufe gesteuert.

DatasetHandler

Der DatasetHandler stellt die Verbindung zum Datenset dar und ermöglicht den Umgang mit den Dateien.

AudioPreprocessor
Der AudioPreprocessor dient im Allgemeinen zur Vorverarbeitung des Audiosignals.
FeatureExtractor
Mithilfe der FeatureExtractor-Klasse werden variabel die benötigten Audio-Merkmale aus den Audio-Clips generiert. Sie ruft die verschiedenen Implementierungen des ExtractorInterface auf.
ModelTrainer
Der ModelTrainer generiert und trainiert das neuronale Netz.
Evaluator
Mit dem Evaluator werden die neuronalen Netze auf ihre Genauigkeit geprüft.
Serializer
Die Serzializer-Klasse speichert die zu Laufzeit generierten Daten für weitere Untersuchungen.

In Abb. 4.5 ist das Sequenzdiagramm dargestellt, dies stellt den Durchlauf einer Konfiguration durch Objekte der verschiedenen Klassen dar.

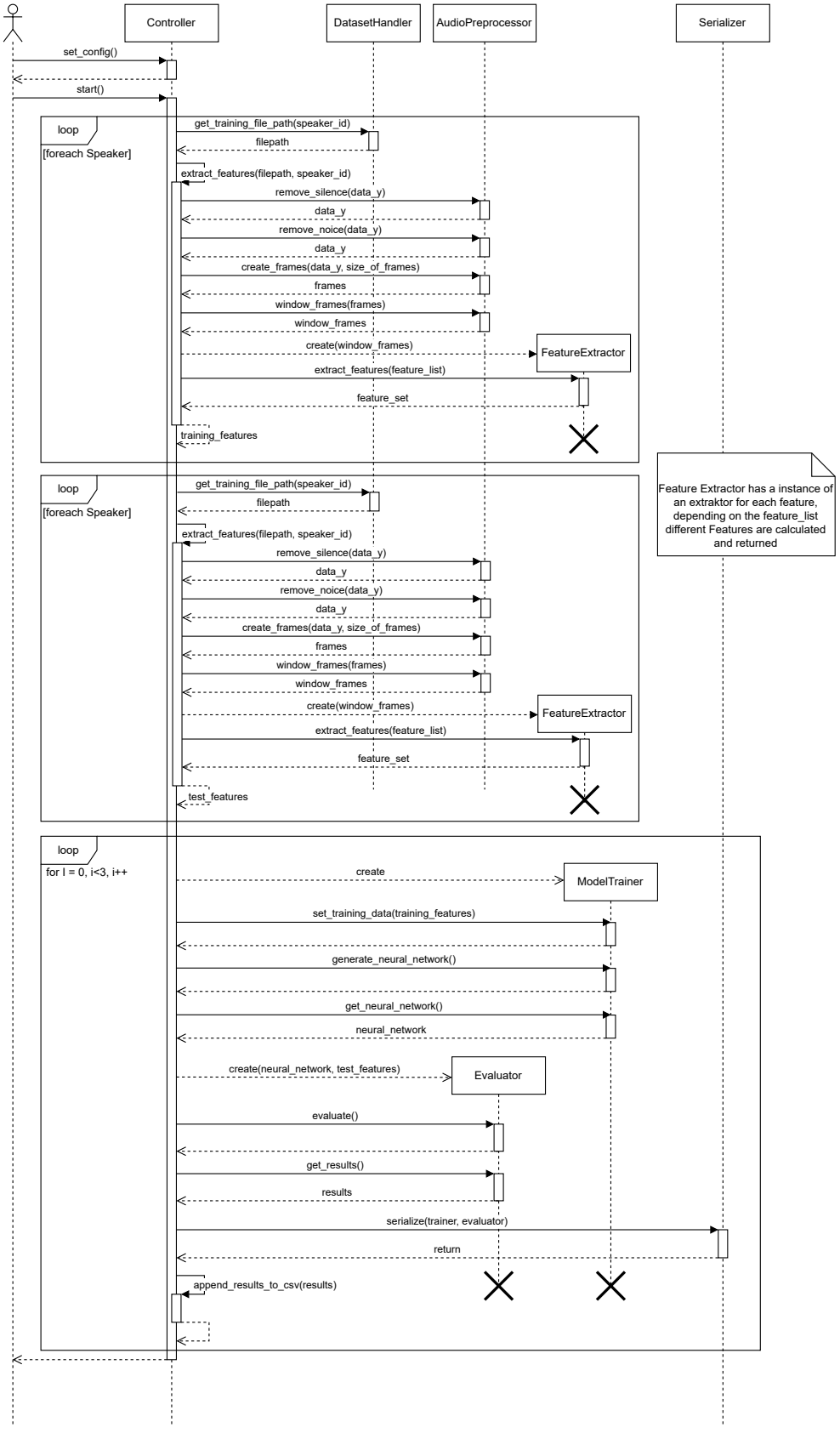


Abb. 4.5: Sequenzdiagramm Versuchssystem

4.4 Demosystem

verfasst von: Henry Schuler

Ziel des Demosystems ist es, den Authentifizierungsprozess mittels Sprechererkennung zu implementieren und damit die Ergebnisse der Arbeit zu präsentieren. Aus den Anforderungen in Kapitel 4.2 geht dabei hervor, dass dies als Webapplikation umzusetzen ist. Dadurch ergibt sich als Basisarchitektur eine Client-Server-Architektur, wobei der Client die Web-Oberfläche und der Server den Authentifizierungsprozess implementiert. Abbildung 4.6 zeigt die grundlegende Architektur des Demosystems.

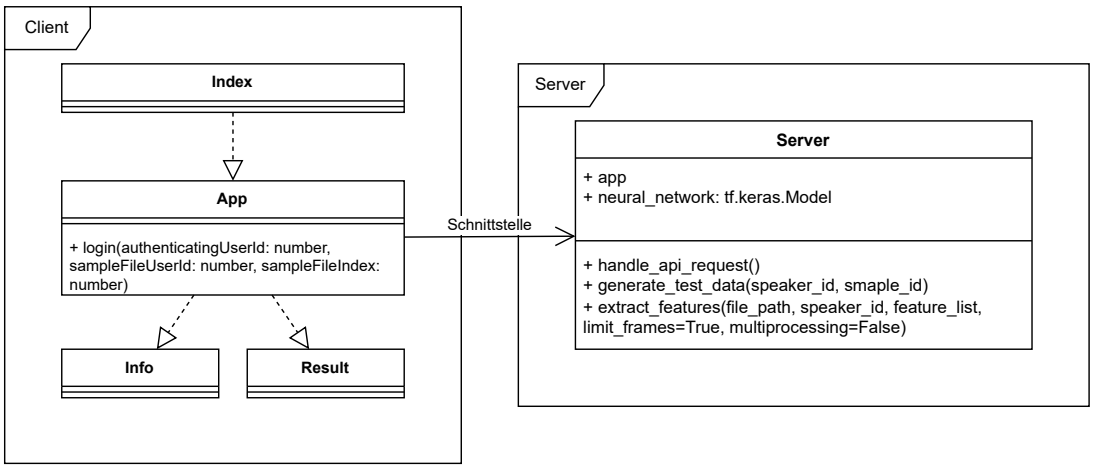


Abb. 4.6: Architektur Demosystem

4.4.1 Client

Der Client wird wie bereits erwähnt als Web-Oberfläche konzipiert. Dabei stellt er Eingabemöglichkeiten in Form eines Logins bereit, die für das Anstoßen des Authentifizierungsprozesses benötigt werden. Im Architekturbild geschieht dies über die `App` Komponente. Über die `login()` Funktion der Komponente wird die Authentifizierungsanfrage an den Server gestellt.

Dabei werden die Daten `authenticatingUserId`, `sampleFileUserId` und `sampleFileIndex` über eine Schnittstelle übermittelt. Da sich das Demosystem auf einen vordefinierten Datensatz beschränkt muss keine Funktion zur Aufnahme neuer Sprechdaten bereitgestellt werden. Aus diesem Grund werden vordefinierte Audiosequenzen für jeden Sprecher bereitgestellt, die mittels den Variablen `sampleFileUserId` und `sampleFileIndex` eindeutig identifiziert werden können.

Die Darstellung des Ergebnisses des Authentifizierungsprozesses wird in eine separate Komponente namens `Results` ausgelagert. Hierbei handelt es sich um eine reine darstellende Komponente.

Für eine verbesserte Nutzerfreundlichkeit wird zusätzlich eine `Info` Komponente eingebunden, welche die grundlegende Bedienung und Vorgehensweise des Demosystems erklärt.

4.4.2 Server

Der Server stellt einen Schnittstellenendpunkt bereit, der Authentifizierungsanfragen eines Clients annimmt, verarbeitet und beantwortet. In Abbildung 4.7 ist ein exemplarischer Ablauf des gesamten Prozesses modelliert.

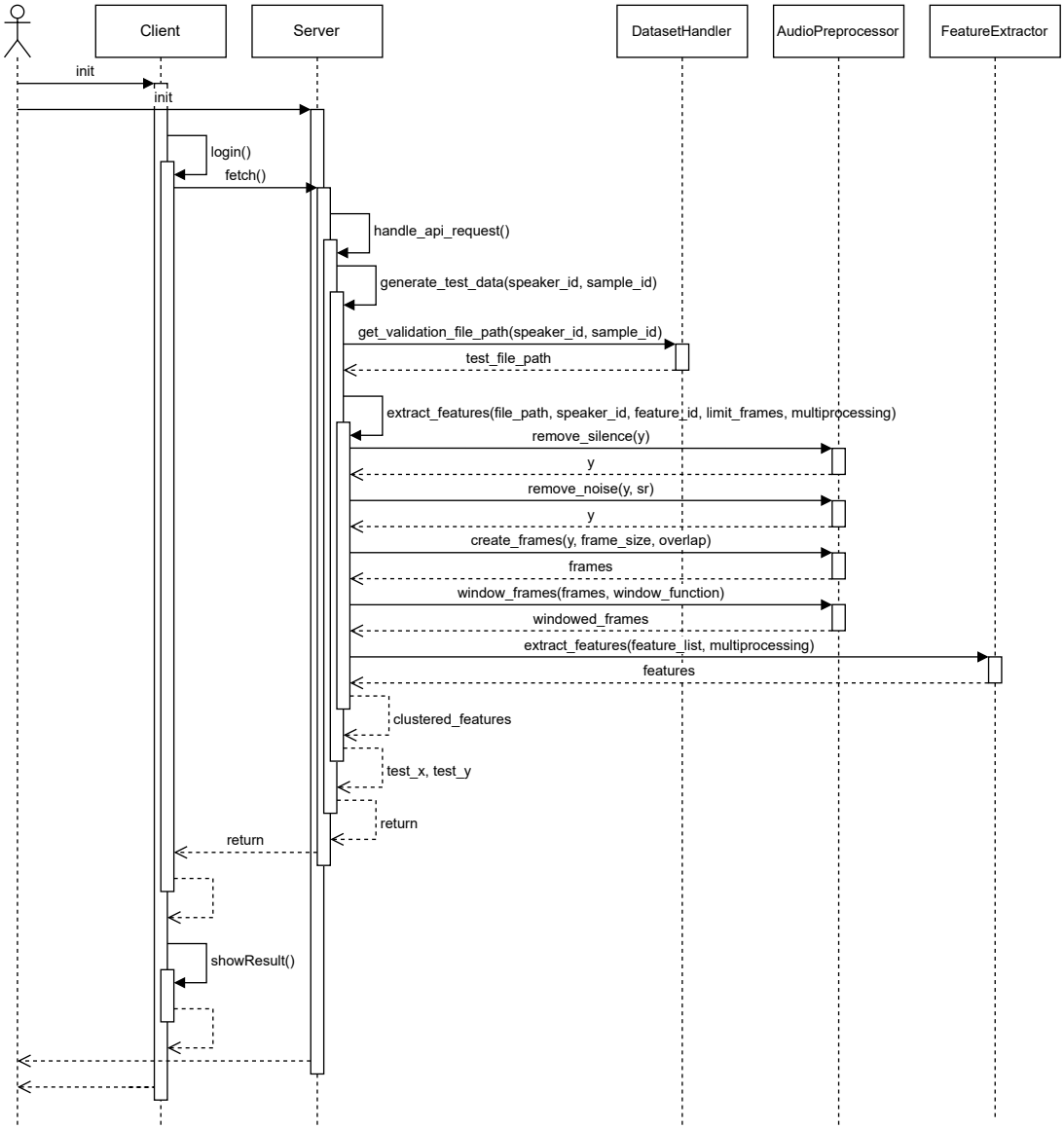


Abb. 4.7: Sequenzdiagramm login()

Eine eingehende Authentifizierungsanfrage wird von der Funktion `handle_api_request()` verarbeitet. Die Klassifizierung der übergebenen Audiodatei verläuft dabei analog zum Ver-

suchssystem, weshalb hier die Komponenten des Versuchssystems (`DatasetHandler`, `AudioPreprocessor`, `FeatureExtractor`) innerhalb der Funktionen `generate_test_data()` und `extract_features()` genutzt werden können.

Die Ergebnisse der Authentifizierung werden über die Schnittstelle zurück an den Client gesendet.

4.4.3 Schnittstellendefinition

Im Konkreten enthält die Anfrage des Clients die folgenden Informationen:

- Zu authentifizierender Sprecher
- Index der ausgewählten Sprachdatei
- Sprecher der ausgewählten Sprachdatei

Die Antwort des Servers beinhaltet die Informationen:

- Ermittelte Wahrscheinlichkeit des zu authentifizierenden Sprechers
- Authentifizierungsstatus
- Ermittelte Wahrscheinlichkeit je Sprecher

5 Umsetzung

verfasst von: Johannes Brandenburger, Lukas Braun, Henry Schuler

Das vorliegende Kapitel befasst sich mit der Umsetzung des Versuchssystems und des Demosystems. Dazu wird zunächst eine Technologieentscheidung durchgeführt. In einem weiteren Schritt wird die systematische Umsetzung der beiden Systeme dargestellt.

5.1 Technologieentscheidung

verfasst von: Johannes Brandenburger, Lukas Braun, Henry Schuler

Die Technologieentscheidung unterteilt sich ebenfalls in eine eigenständige Entscheidung für das Versuchssystem, sowie das Demosystem. Durch die Abhängigkeit der beiden Systeme, bezieht sich die Technologieentscheidung des Demosystems auf die Entscheidung des Versuchssystems.

5.1.1 Versuchssystem

verfasst von: Johannes Brandenburger, Lukas Braun

Aus der Konzeption in Kapitel 4 gehen verschiedene Anforderungen an die Technologie hervor. Durch die Verwendung von neuronalen Netzen werden die verfügbaren Technologien bzw. Programmiersprachen bereits eingeschränkt. Um weitere Technologieentscheidungen zu treffen, muss zunächst eine Programmiersprache ausgewählt werden. Da in einer Umfrage unter Machine-Learning-Entwicklern und Data Scientists 57 % Python als Programmiersprache nutzen und 33 % diese sogar priorisieren, wird für dieses Projekt Python als Programmiersprache festgelegt (vgl. Vision Mobile 2017, S. 16). In einer allgemeinen Umfrage zu verwendeten Programmiersprachen liegt Python mit 49,2 % auf Platz drei (vgl. Yepis 2023). Das bedeutet, dass knapp die Hälfte der befragten Entwickler unter anderem Python verwenden, somit ist eine ausreichende Verbreitung gewährleistet. Die Verbreitung ist vor allem deshalb ein wichtiger Aspekt, da es durch eine hohe Verbreitung ein großes Eco-System mit vielen quelloffenen Ressourcen und bereits durchgeführten Projekten gibt.

Um Machine-Learning mit Python zu betreiben, gibt es mehrere Bibliotheken zur Auswahl. Die drei bekanntesten Open-Source-Bibliotheken sind dabei „TensorFlow“, „PyTorch“ und „SciKit-Lern“ (vgl. MSV 2020).

Diese Bibliotheken sind sich relativ ähnlich. Vorteile einer Bibliothek bringen automatisch auch Nachteile, die sich somit gegenseitig aufheben.

Für diese Studienarbeit wird TensorFlow genutzt, da es laut der „Stack Overflow Developer

Survey 2022“ eine größere Nutzerbasis hat (vgl. Stack Overflow 2022). Zudem spricht für diese Entscheidung auch die Tatsache, dass die Entwickler dieser Studienarbeit bereits Vorkenntnisse in TensorFlow haben.

Zusätzlich wird das Modul „Keras“ verwendet, welches eine entwicklerfreundliche High-Level-Schnittstelle für TensorFlow ist und somit eine einfachere Entwicklung ermöglicht (vgl. *Keras* 2023).

5.1.2 Demosystem

verfasst von: Henry Schuler

Für die Technologieentscheidung muss zunächst zwischen den zwei Komponenten Client und Server unterscheiden werden. Da für beide Teile unterschiedliche Anforderungen erfüllt werden müssen, erfolgt die Technologieentscheidung separat. Außerdem findet eine Festlegung des verwendeten Protokolls für die Schnittstellenkommunikation statt.

Client Die grundlegende Anforderung an den Client ist die Umsetzung als Webapplikation. Daraus leiten sich in erster Linie vier verschiedene Programmiersprachen ab: JavaScript/TypeScript, Python, PHP und Ruby.

Für eine weitergehende Einschränkung ist der vorausgesetzte Funktionsumfang relevant. Dieser ist sehr gering, da lediglich Input-Elemente für die Angabe eines zu authentifizierenden Sprechers, sowie zur Auswahl einer zu verwendeten Datei bereitgestellt werden müssen. Außerdem muss eine Verbindung zu einem Back-End (Server) aufgebaut werden können.

Da diese Anforderungen durch alle vier Kandidaten erfüllt werden, muss die Entscheidung durch andere Kriterien erfolgen. Hierbei spielen vor allem zwei Aspekte eine Rolle. Eine Umfrage von Stack Overflow aus dem Jahr 2020 zeigt, dass JavaScript die beliebteste Programmiersprache von professionellen Entwickler ist. Moderne Frameworks wie ReactJS und Angular haben die klassische Webprogrammierung in PHP über die letzten Jahre abgelöst (vgl. Stack Overflow 2020).

Da es sich außerdem um ein Demosystem handelt, welches lediglich die Ergebnisse dieser Arbeit präsentieren soll und somit nicht in einem produktiven Umfeld betrieben wird, spielt die Technologieentscheidung nur eine untergeordnete Rolle. Die Kompetenzen des Entwicklerteams liegen vor allem im Bereich der React Programmierung mit JavaScript beziehungsweise TypeScript.

Zusammenfassend fällt die Technologieentscheidung für die Client-Implementierung somit auf das JavaScript Framework ReactJS in Kombination mit TypeScript. TypeScript ermöglicht dabei

gegenüber JavaScript die Verwendung von Datentypen, wodurch Datentypfehler im Rahmen der Implementierung vermieden werden können.

Schnittstelle Im Web-Umfeld spielt das Hypertext Transfer Protocol (HTTP) eine besondere Rolle. Dabei handelt es sich um ein zustandsloses Protokoll zur Übertragung von Daten auf der Anwendungsschicht. Ein Anwendungsfall ist dabei die Anfrage von Webseiten im Browser über die URL. Dazu wird eine sogenannte HTTP-GET Anfrage an den Zielserver gesendet, welcher anschließend mit der angeforderten Ressource antwortet.

Nach demselben Prinzip kann das HTTP Protokoll ebenfalls dazu verwendet werden, eine Kommunikation zwischen Client und Server zu ermöglichen. Die zu übergebenden Variablen des Clients an den Server können dabei innerhalb der URL des GET-Requests übertragen werden. Gleichzeitig können die Ergebnisse des Servers beispielsweise im JSON-Format an den Client zurückgesendet werden.

Da das HTTP Protokoll somit alle benötigten Anforderungen erfüllt, kommt es für die Schnittstellenkommunikation zum Einsatz.

Server Für den Authentifizierungsprozess implementiert der Server denselben Ablauf wie das Versuchssystem. Um eine doppelte Entwicklung zu vermeiden, entsteht somit eine Abhängigkeit zwischen der Technologieentscheidung des Versuchssystems und des Servers. Aus der Technologieentscheidung des Versuchssystems in Kapitel 5.1.1 geht die Programmiersprache Python hervor. Somit sollte zunächst geprüft werden, ob die Anforderungen des Servers in der Sprache Python umsetzbar sind.

Neben dem Authentifizierungsprozess spielt ausschließlich die Schnittstellenkommunikation zwischen Client und Server eine Rolle für die Technologieentscheidung des Servers. Da als Kommunikationsprotokoll HTTP eingesetzt wird, muss also eine Bibliothek gefunden werden, die die Verwendung dieses Protokolls in Python ermöglicht. Hierzu kann Flask verwendet werden.

Bei Flask handelt es sich um ein Web Framework, welches eine Bereitstellung von Application Programming Interface (API) Endpunkten, die über das HTTP Protokoll abrufbar sind, ermöglicht.

Somit können alle Anforderungen an die Serverapplikation mittels der Programmiersprache Python umgesetzt werden. Dazu können die bereits entwickelten Komponenten für den Authentifizierungsprozess aus dem Versuchssystem übernommen werden.

5.2 Versuchssystem

verfasst von: Johannes Brandenburger, Lukas Braun

Die Umsetzung des Versuchssystems unterteilt sich in vier Unterkapitel. Analog zu der Konzeption des Versuchssystems in Kapitel 4.3, wird zunächst auf die Generierung der Feature-Kombinationen eingegangen. Anschließend wird ein Datensatz ausgewählt und für die Verwendung in diesem System vorbereitet. Daraufhin wird die Umsetzung der Audio-Vorverarbeitung sowie der Feature-Extraktion genau beschrieben. Abgeschlossen wird das Kapitel mit der Umsetzung der Klassifikation und Evaluierung.

5.2.1 Feature-Kombination

verfasst von: Lukas Braun

Wie bereits in Kapitel 4.3.2 erwähnt, müssen zuerst die verschiedenen Konfigurationen erzeugt werden. Hierzu werden die vorher definierten Werte in einer JSON-Datei erfasst und durch ein eigenes Kombinierungs-Skript alle möglichen Konfiguration mit einer ID erzeugt.

Die Konfigurationswerte für die Kombinierung sind im Listing 1 dargestellt.

```
Configs = {
    "amount_of_frames": [10000, 15000],
    "size_of_frame": [400, 600],
    "LPC": {
        "order": [13, 20],
        "weight": [0, 1]
    },
    "MFCC": {
        "order": [13, 20],
        "weight": [0, 1]
    },
    "LPCC": {
        "order": [13, 20],
        "weight": [0, 1]
    },
    "delta_MFCC": {
        "order": [13],
        "weight": [0, 1]
    }
}
```

Listing 1: Konfigurationsmöglichkeiten

Der `weight` Parameter gibt lediglich an, ob in dieser Konfiguration dieses Feature verwendet werden soll oder nicht, um verschiedene Kombinationen zu erreichen. So gibt es auch manche Kombinationen, die beispielsweise nur MFCC-Merkmale in der Analyse berücksichtigen. Hierbei muss beachtet werden, dass durch die Kombination auch Konfigurationen entstehen, in

welchen nichts berechnet werden muss. Diese werden manuell entfernt. Die anderen Variablen wie `order` und dessen Werte wurden bereits in Kapitel 4.3.2 erklärt.

5.2.2 Datensatz

verfasst von: Lukas Braun

Für die Evaluierung der Stimmmerkmale wird ein geeigneter Datensatz benötigt. Hierzu wurde nach einer Internetrecherche auf der Plattform „Kaggle“ ein entsprechender Datensatz gefunden, der die Anforderung aus Kapitel 4.3.3 erfüllt (vgl. Jain 2019). Der Originaldatensatz enthält Audiodaten zu 50 Sprechern mit mindestens 60 Minuten Aufzeichnung pro Sprecher in mehreren kompressionslosen WAV Dateien.

In einem ersten Schritt müssen die Daten manuell aufbereitet werden. Der Originaldatensatz lässt sich in zwei Teile teilen, während der erste Teil aus YouTube Videos besteht, sind im zweiten Teil Aufnahmen von englischen Hörbüchern enthalten. Da die Aufnahmequalität der Hörbücher deutlich besser ist, werden nur diese Datensätze weiterverwendet. In einem Folgeschritt werden die einzelnen Dateien der Datensätze zu einer Datei zusammengefügt und genauer betrachtet. Hierbei können längere Pausen und Abweichungen der Datensätze, wie z.B. mehrere Sprecher oder eine andere Sprache identifiziert werden. Diese Pausen werden entfernt und die betroffenen Datensätze aus dem Datensatz entfernt. Dadurch bleiben 25 geeignete Datensätze übrig, aus diesen werden unter dem Aspekt des Geschlechts 20 Datensätze ausgewählt, sodass neun weibliche und elf männliche Sprecher im finalen Datensatz sind, da nur neun weibliche Datensätze geeignet sind. Abschließend wird für jeden Sprecher eine Audiodatei zum Trainieren des neuronalen Netzes mit acht Minuten und zur Evaluation neun Sequenzen mit jeweils 15 Sekunden erstellt. Das Testmaterial ist nicht im Trainingsmaterial enthalten.

5.2.3 Vorverarbeitung und Feature-Extraktion

verfasst von: Johannes Brandenburger

Um eine effektive Verarbeitung eines Audiosignals zu ermöglichen, muss zunächst eine Vorverarbeitung des Signals erfolgen (s. Kapitel 2.3.1). Hierfür wird der Audio-Clip mittels der Bibliothek „Librosa“ geladen. Anschließend folgt der erste Schritt der Vorverarbeitung, das Entfernen von stillen Passagen. Dafür wird ein selbst entwickelter Algorithmus verwendet, der erkennt, wenn sich die Amplitude des Audiosignals über einen gewissen Zeitraum unterhalb eines definierten Schwellwerts befindet.

```
1 for i, amp in enumerate(y):
2 if abs(amp) < threshold:
3     counter_below_threshold += 1
4 else:
```

```
    if counter_below_threshold > pause_length_in_ms:
        for index in range(i-counter_below_threshold+keep_at_start_and_end, i-
    keep_at_start_and_end):
            indices_to_remove.append(index)
    counter_below_threshold = 0
```

Listing 2: Remove Silence

Nach dem Entfernen der Pausen wird das Hintergrundrauschen entfernt. Dafür wird ein externer Algorithmus von Tim Sainburg verwendet. Dieser basiert auf dem Vorgehen des bekannten Ton-Verarbeitungsprogramm „Audacity“. Zusammengefasst transformiert der Algorithmus mithilfe von FFT das Audio-Signal zunächst in den Frequenzraum. Hier werden dann die Störfrequenzen über Schwellwertverfahren lokalisiert und mittels Filter eliminiert. Anschließend wird das Signal wieder zurücktransformiert (Sainburg 2019).

Die abschließende Unterteilung des Audiosignals in Frames, sowie das Windowing der Frames findet mit Hilfe von `numpy`-Operationen in den Funktionen `create_frames()` und `window_frames()` statt. Die passende Fensterfunktion wird dabei ebenfalls durch die `numpy`-Bibliothek bereitgestellt. Bei dem Aufteilen in Frames wird eine Überlappung von 50 % gewählt.

Für das Extrahieren der Merkmale aus dem Audiosignal wird ein Ansatz gewählt, der eine einfache Erweiterung des Programms um verschiedene andere Verfahren ermöglicht. Dazu wird das Designmuster Strategie in abgewandelter Form verwendet, wobei zunächst ein Interface für die Berechnungsverfahren erstellt werden muss. Dieses definiert die Funktion `calculate_features()`, welche in den abgeleiteten Klassen implementiert wird. Für die eigentlichen Berechnungen der Koeffizienten wird wieder die Signal-Verarbeitungs-Bibliothek Librosa verwendet. Beispielhaft für die Berechnung von MFCC sieht man in Listing 3, wie unkompliziert sich die Bibliothek Librosa verwenden lässt. Allerdings müssen viele Parameter, wie zum Beispiel die Länge des zu verwendenden FFT-Fensters (`n_fft`) gefüllt bzw. geändert werden. Diese Werte ergeben sich aus den von uns ausgewählten Parameter (beschrieben in 4.3.2) und den Anweisungen in der offiziellen Dokumentation der Librosa-Bibliothek (vgl. librosa development team 2023).

```
class MFCCExtractor(ExtractorInterface):

@staticmethod
def calculate_mfcc(frame, sr, order):
    mfcc = librosa.feature.mfcc(y=frame, sr=sr, n_mfcc=order, n_fft=1024, hop_length=512)
    mfcc_of_frame = np.mean(mfcc.T, axis=0)
    return mfcc_of_frame

def calculate_features(self, frames, sr, order, multiprocessing=False):
    if multiprocessing:
        with Pool(8) as p: # run function on 8 cores
            mfccs = p.starmap(self.calculate_mfcc, [(frame, sr, order) for frame in frames
    ])
```

```
13          return mfccs
14      else:
15          mfccs = []
16          for frame in frames:
17              mfccs.append(self.calculate_mfcc(frame, sr, order))
18          return mfccs
```

Listing 3: Klasse zur Extraktion von MFCC-Merkmalen

Da für die Versuchsreihe sehr viele zeitaufwändige Berechnungen laufen müssen, ist das Extrahieren der Merkmale mit Multiprocessing implementiert. Hierfür wird die Python-interne Bibliothek „mutliprocessing“ genutzt, um die Berechnung auf mehrere Kerne zu verteilen.

5.2.4 Klassifikation und Evaluierung

verfasst von: Johannes Brandenburger, Lukas Braun

In dem Kapitel 4.3.5 wurden neuronale Netze als Klassifikationsmodell gewählt, um die zuvor berechneten Merkmale zu evaluieren. Hierbei werden zuerst neuronale Netze mit den berechneten Trainingsdaten trainiert. Die Netze bestehen aus drei Hidden-Layer mit jeweils 128, 64 und 32 Neuronen. Dieser absteigende Aufbau stellt sich bei Versuchen als geeignete Lösung heraus.

Wie bereits im Konzept dargestellt müssen mehrere neuronale Netze trainiert werden, in diesem Fall werden für jede Konfiguration drei neuronale Netze erstellt.

```
# create model
tf.keras.backend.clear_session()
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=[input_layer_neurons]),
    *hidden_layer, # 128, 64, 32
    tf.keras.layers.Dense(output_layer_neurons, activation=tf.nn.softmax),
])
model.compile(optimizer=tf.optimizers.Adam(), loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])

# train model
model.fit(X, y, epochs=epochs)
```

Listing 4: Erstellen und Trainieren eines neuronalen Netzes

Nach dem Training erfolgt die Evaluation der neuronalen Netze, hierzu wird der tatsächliche Authentifizierungsprozess nachgebildet. Die berechneten Testdaten werden auf die neuronalen Netze angewandt, das neuronale Netz klassifiziert also die Daten, indem es die Features jedes Frames einem der 20 Sprecher zuordnet. Diese Klassifizierung wird durch die Tensorflow-Methode `model.predict()` realisiert, wie das Listing 5 zeigt.

```
prediction = self.neural_network.predict(np.asarray(self.test_data_x[i])) # generate
    prediction for the sample
correspondence = {} # mapping: speaker to result
for id in range(20): # for each speaker
    correspondence[f"{id}"] = np.count_nonzero(np.argmax(prediction, axis=1) == id) / len(
    prediction) # normalization
```

Listing 5: Evaluation mit model.predict() und Normalisierung des Ergebnisses

Die Anzahl der Frames ist dabei von der Frame-Länge der jeweiligen Konfiguration abhängig, weshalb die Zuordnungsverteilungen normiert, also von einer absoluten in eine relative Zuordnung umgerechnet, werden müssen.

Da für jeden Sprecher fünf Testdateien verwendet werden (die Schleife hierfür wird übersichtlichkeitshalber nicht in den Listings gezeigt), entstehen pro neuronalem Netz 100 Zuordnungsverteilungen. Durch die Berechnung von drei neuronalen Netzen pro Konfiguration ergibt dies 300 Zuordnungsverteilungen pro Konfiguration.

Nach Abschluss der Klassifikation aller Features können diese Datensätze ausgewertet und somit ein Rückschluss auf die beste Merkmalskombination getroffen werden. Die Durchführung und Evaluation des Versuchs sind in Kapitel 6 dargestellt.

5.3 Demosystem

verfasst von: Henry Schuler

Die Umsetzung von Client und Server ist im Folgenden kurz beschrieben. Dabei werden lediglich die wichtigsten Aspekte benannt. Da der Authentifizierungsablauf bereits im vorangehenden Kapitel erläutert wurde, wird in der Umsetzung des Servers nicht mehr darauf eingegangen.

5.3.1 Client

Durch die Verwendung von ReactJS wird die Ordnerstruktur durch das Framework vorgeschrieben. Diese ist in Abbildung 5.1 dargestellt.

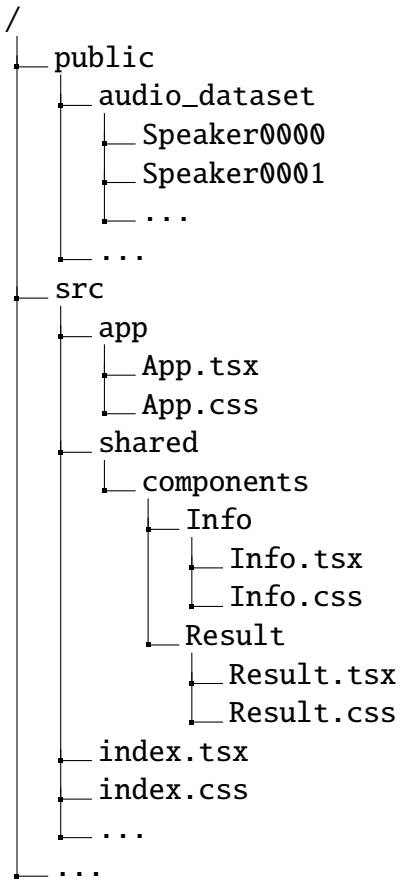


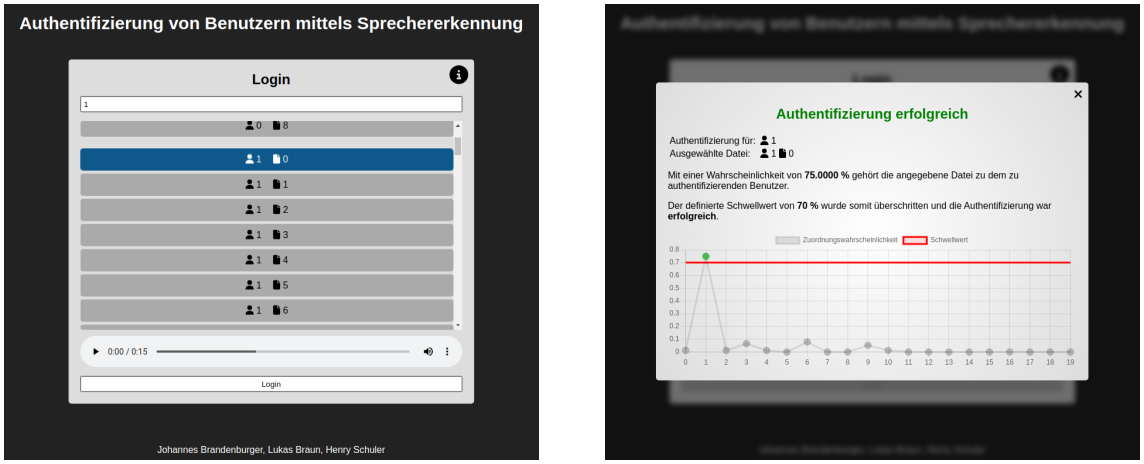
Abb. 5.1: Ordnerstruktur Demosystem Client

Die Projektspezifischen Dateien befinden sich dabei in dem Unterordner `public`, sowie `src`. Über den `public`-Ordner können Ressourcen wie Bilder oder sonstige Objekte, die durch die Website angezeigt werden zur Verfügung gestellt werden. Sie können dadurch ebenfalls direkt über die URL im Browser abgerufen werden.

Die aufrufbaren Webseiten befinden sich im `src`-Ordner. Der Hypertext Markup Language (HTML)-Code ist in ReactJS in den `*.tsx` Dateien enthalten und erlaubt somit die direkte Einbindung von JavaScript Variablen in die HTML Komponenten. Da es sich bei der Client-Applikation um eine Single Page Application handelt, stellt die `index.tsx` Datei den einzigen Einstiegspunkt der Applikation dar. Darin wird die Komponente `App` aus der Datei `App.tsx` eingebunden.

Zusätzlich können einzelne Komponenten in separate Dateien ausgelagert werden, um auf der einen Seite eine bessere Codeübersicht zu erzeugen und auf der anderen Seite einzelne Visualisierungen an verschiedenen Stellen zu implementieren. Diese Komponenten werden in dem Unterordner `src/shared/components` abgelegt.

Die Benutzeroberfläche der `App` Komponente zeigt eine Login-Abfrage (siehe Abbildung 5.2a).



(a) App/Login **(b)** Result

Abb. 5.2: Web-Oberfläche

Diese unterteilt sich in zwei Abschnitte. Zunächst kann über ein Eingabefeld die ID des zu authentifizierenden Nutzers angegeben werden. Die Eingabe ist dabei auf die Zahlen 0-19 begrenzt. Die Eingabe anderer Zeichen führt dazu, dass der Inhalt des Eingabefelds gelöscht wird. Anschließend kann eine zu verwendende Audiodatei aus einer Liste ausgewählt werden. Für die Erzeugung der Liste in der Benutzeroberfläche, wird die JSX-Funktion `map()` verwendet um Schaltflächen für jeweils neun Dateien pro Sprecher zu generieren (vgl. Listing 6, Z. 4,5). Über einen `onClick()` Event-Listener (Z. 10) wird dabei der jeweilige Index des selektierten Elements gespeichert.

```
1  <div className="fileList">
2      {
3          // create an array with all numbers from 0 to 20
4          Array.from(Array(20).keys()).map((id) => {
5              return Array.from(Array(9).keys()).map((fileIndex) => {
6                  return (
7                      <div
8                          key={"speaker" + id + "file" + fileIndex}
9                          className={"fileItem " + (sampleFileUserId === id &&
       sampleFileIndex === fileIndex ? "selected" : "")}
10                         onClick={() => {
11                             setSampleFileUserId(id);
12                             setSampleFileIndex(fileIndex);
13                         }}
14                     >
15                         <span>
16                             <FontAwesomeIcon icon={faUser} />
17                              {id}
18                         </span>
19                         <span>
20                             <FontAwesomeIcon icon={faFile} />
21                              {fileIndex}
22                         </span>
23                     </div>
```

```
24                 )
25             })
26         })
27     }
28 </div>
```

Listing 6: Auswahl der Audiodatei

Um die Benutzerfreundlichkeit zu verbessern, besteht dabei zusätzlich die Möglichkeit, die ausgewählte Datei im Browser abzuspielen. Dafür wird (wie in Listing 7 dargestellt) der HTML `<audio>` Tag verwendet. Die dafür benötigten Ressourcen (WAV-Dateien) werden im Ordner `public/audio_dataset` zur Verfügung gestellt.

```
1 <audio controls ref={audioRef}>
2     <source src={`./audio_dataset/Speaker${String(sampleFileUserId).padStart(4, '0')}/
      Validation_Speaker${String(sampleFileUserId).padStart(2, '0')}_${String(
      sampleFileIndex).padStart(4, '0')}.wav`} type="audio/wav" />
3     Audio files are not supported by your browser.
4 </audio>
```

Listing 7: Audio Tag

Über den Login-Knopf wird die Authentifizierungsanfrage an den Server gesendet. Der Knopf kann dabei erst durch den Benutzer gedrückt werden, wenn sowohl in dem Eingabefeld eine valide Zahl steht, als auch eine Authentifizierungsdatei ausgewählt ist.

Für die Anfrage an den Server wird die `fetch()` Funktion von JavaScript verwendet (vgl. Listing 8, Z. 3). Die zu übergebenden Parameter werden als URL-Parameter angegeben. Ist der Server über die angegebene IP-Adresse und Port nicht erreichbar, so wird dies dem Benutzer mittels eines `alert()` mitgeteillt (Z. 19).

```
1  async function login(authenticatingUserId: number, sampleFileUserId: number,
       sampleFileIndex: number) {
2      try {
3          const response = await fetch(`http://${configData.SERVER_URL}:${configData.
       SERVER_PORT}/?speaker_id=${sampleFileUserId}&sample_id=${sampleFileIndex}&
       selected_speaker_id=${authenticatingUserId}`);
4
5          console.log(response)
6          if (!response.ok) {
7              return {
8                  absolute_accuracy_of_selected_speaker: 0,
9                  is_authenticated: false,
10                 absolute_accuracy_of_all_speakers: []
11             };
12         }
13
14         const json = await response.json();
15
16         return json;
17     } catch (e) {
```

```
18          console.error(e);
19          alert(‘Es konnte keine Verbindung zum Server (${configData.SERVER_URL}:${
        configData.SERVER_PORT}) hergestellt werden!‘)
20          return {
21              absolute_accuracy_of_selected_speaker: 0,
22              is_authenticated: false,
23              absolute_accuracy_of_all_speakers: []
24          }
25      }
26  }
```

Listing 8: login()

Die Darstellung des Ergebnisses mittels der `Result` Komponente ist in Abbildung 5.2b dargestellt. Dem Benutzer werden sowohl seine Authentifizierungsangaben, als auch das Ergebnis der Authentifizierung in Textform angezeigt. Die Klassifikation als „erfolgreich“ oder „fehlgeschlagen“ kann sowohl dem Text als auch der Farbe der Überschrift (grün oder rot) entnommen werden.

Zusätzlich erhält der Benutzer einen grafischen Einblick in die Wahrscheinlichkeitsverteilung der Zuordnung der ausgewählten Sprecher-Datei zu den verschiedenen Sprechern. Das Diagramm wird mithilfe der Bibliotheken chart.js und react-chartjs-2 eingebunden.

5.3.2 Server

Wie in der Technologieentscheidung bereits erwähnt, wird der Server mithilfe der Bibliothek Flask implementiert. Listing 9 zeigt dabei die wesentlichen Codeabschnitte für die Umsetzung des Servers als Flask Applikation.

```
1  app = Flask(__name__)
2  CORS(app)
3
4  @app.route("/", methods=["GET", "POST"])
5  def handle_api_request():
6      ...
7
8  if __name__ == '__main__':
9      app.run(debug=False, host='127.0.0.1', port=5500)
```

Listing 9: Ausschnitt server.py

Zunächst muss eine neue Flask `app` erstellt werden (Z. 1). Mittels `CORS(app)` wird dabei sichergestellt, dass eine Verbindung zwischen Client und Server trotz unterschiedlichem Ursprung möglich ist.

Die Authentifizierungsfunktion wird über die Wrapper-Funktion `handle_api_request()` und dem Decorator `@app.route()` als API Endpoint verfügbar gemacht. Die Abarbeitung der Anfragen unterteilt sich in die in Abbildung 5.3 dargestellten fünf Schritte.

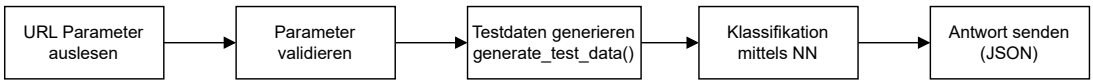


Abb. 5.3: Ablaufdiagramm handle_api_request()

Dafür werden zunächst die übergebenen Parameter aus der URL ausgelesen. Anschließend wird überprüft ob es sich bei den Werten um numerische Werte handelt, die im vorgegebenen Definitionsbereich liegen (vgl. Listing 10).

```
1 if not (
2     speaker_id.isnumeric() and
3     sample_id.isnumeric() and
4     selected_speaker_id.isnumeric() and
5     0 <= int(speaker_id) <= 19 and
6     0 <= int(sample_id) <= 8 and
7     0 <= int(selected_speaker_id) <= 19
8 ):
9     return parameter_error_message
```

Listing 10: Parametervalidierung handle_api_request()

Da die Generierung der Testdaten analog zu Kapitel 5.2 verläuft, wird darauf nicht weiter eingegangen. Mit den generierten Daten wird anschließend die Klassifikation durch das NN durchgeführt und die Ergebnisse aufbereitet und im JSON-Format an den Client zurückgesendet.

Das neuronale Netz, welches vom Server für die Authentifizierung von Benutzern verwendet wird, ergibt sich aus der Evaluation des Versuchssystems im nachfolgenden Kapitel 6.

Über die Anweisung `app.run()` unter Angabe der IP-Adresse und des Ports, kann der Server gestartet werden (Listing 9, Z. 9).

6 Evaluation

verfasst von: Johannes Brandenburger, Henry Schuler

In dem vorliegenden Kapitel wird kurz die Durchführung der Versuchsreihe beschrieben und die Evaluierung der Ergebnisse des Versuchssystems durchgeführt. Hierzu werden die Ergebnisse aufbereitet und anhand mehrerer Metriken verglichen.

Das Versuchssystem wird über das Ausführen der `main.py`-Datei gestartet (siehe linke Lane im Sequenzdiagramm (Abbildung 4.5)). Hier lässt sich spezifizieren, welche der Konfigurationen bei der aktuellen Ausführung analysiert werden sollen, wie man in Zeile 8 des Listing 11 erkennen kann.

```
1  # load configs from json file
2  configs = []
3  with open(os.path.join(os.path.dirname(__file__), "Configs", "configs.json"), "r") as
       json_file:
4      configs = json.load(json_file)
5
6  # range for current run
7  controller = Controller(os.path.join(os.path.dirname(__file__), "results.csv"))
8  for i in range(0, 20): # range for current run
9      controller.set_config(configs[i])
10     controller.start()
```

Listing 11: Ausführen einer Versuchsreihe

Da die Analyse von einer Konfiguration im Durchschnitt etwa 35 Minuten dauert, werden die Konfigurationen auf mehrere Geräte aufgeteilt. Es kommen insgesamt fünf verschiedene Geräte mit unterschiedlicher Leistung zum Einsatz, die die Konfigurationen blockweise abarbeiten. Der gesamte Prozess dauert ca. eineinhalb Wochen, wobei zu beachten ist, dass die verwendeten Geräte nicht rund um die Uhr rechnen. Die Ergebnisse der Analysen werden in CSV-Dateien geschrieben, welche nach dem Beenden der Berechnungen zusammengeführt werden.

Für die Auswertung der Zuordnungsverteilungen aller Konfigurationen des Versuchssystems werden die Ergebnisse zunächst grafisch dargestellt. Dabei werden in allen Grafiken die Ergebnisse der drei neuronalen Netze pro Konfiguration miteinander verrechnet. Es werden vier verschiedene Aspekte betrachtet (vgl. Abbildung 6.1).

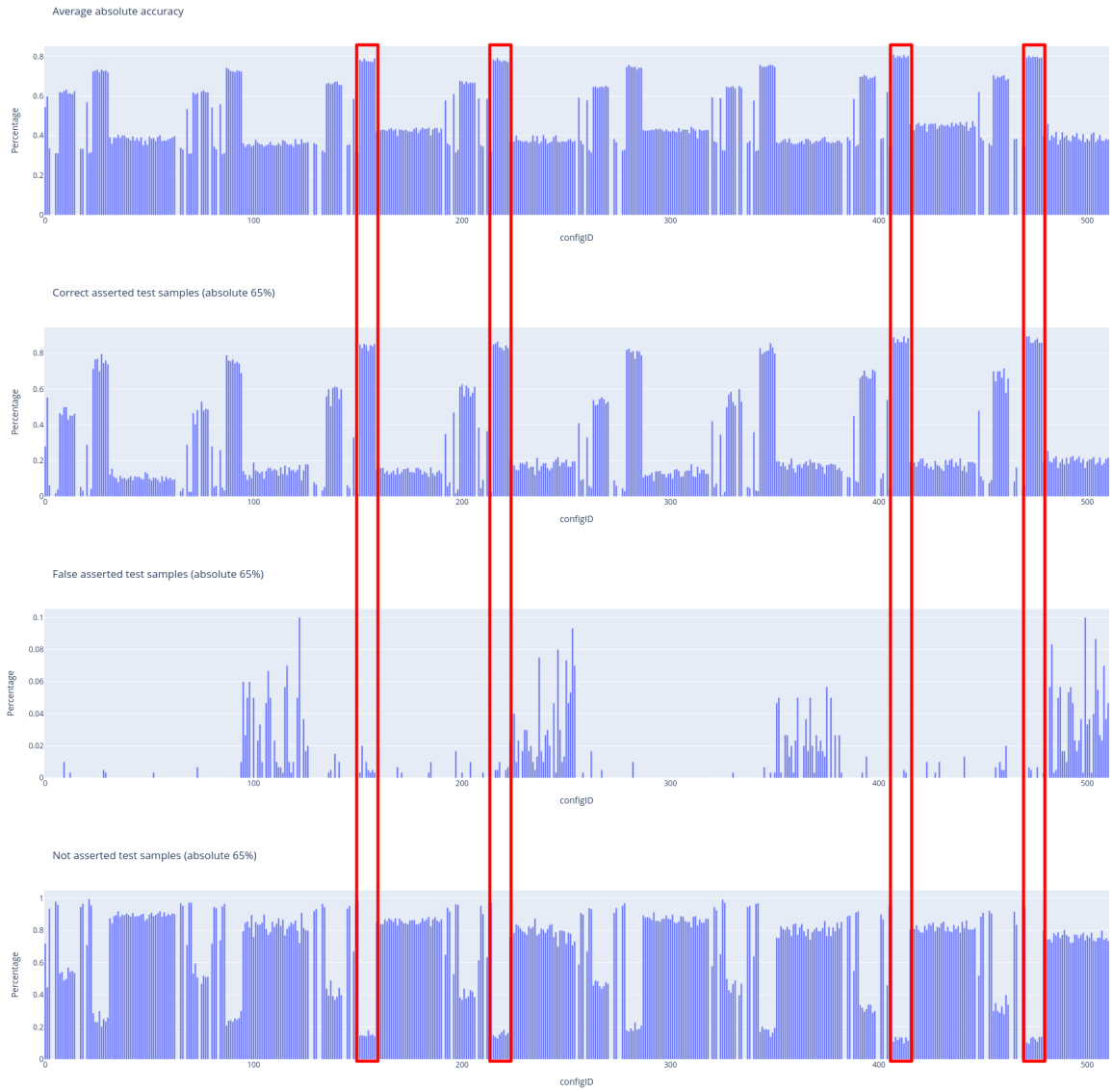


Abb. 6.1: Auswertung Versuchssystem

Für die durchschnittliche absolute Genauigkeit (erstes Diagramm) werden die Wahrscheinlichkeiten für den zu identifizierenden Benutzer pro Testdatei aufsummiert und der Durchschnitt gebildet. Eine Genauigkeit von 80 % entspricht damit der Aussage, dass das neuronale Netz in Kombination mit der jeweiligen Konfiguration dem zu authentifizierenden Benutzer im Durchschnitt 80 % der Frames der Testdatei zuordnet.

Bereits in dieser Grafik zeigt sich die Bildung von Clustern. Die besten Konfigurationen erreichen eine Genauigkeit von über 65 %, weshalb dieser Wert für die folgenden drei Auswertungen verwendet wird. Hier werden die Testdateien mittels dieses Wertes in die drei Kategorien korrekt zugeteilt, falsch zugeteilt und nicht zugeteilt eingeordnet.

Eine Testdatei gilt als korrekt zugeteilt, wenn dem zu authentifizierenden Benutzer mindestens 65 % der Frames zugeordnet werden. Analog gilt die Testdatei als nicht korrekt zugeteilt, wenn einem anderen Benutzer mindestens 65 % der Frames zugeordnet werden. Erreicht kein Nutzer mindestens 65 %, so wird die Testdatei als nicht zugeteilt eingestuft.

Für die Bewertung der Konfigurationen gibt es grundsätzlich zwei unterschiedliche Ansätze. Zunächst kann eine Konfiguration anhand der Anzahl an richtig zugewiesenen Samples bewertet werden. Dabei steht der Fokus besonders darauf, dass möglichst viele Sprecher möglichst zuversichtlich erkannt und damit auch authentifiziert werden. In dieser Betrachtung wird allerdings die Anzahl an falsch, beziehungsweise nicht zugewiesenen Samples vernachlässigt, was bedeutet, dass es trotz einer hohen Anzahl an richtigen Zuweisungen auch eine verhältnismäßig hohe Anzahl an falschen Zuweisungen geben kann. Dies bedeutet für das System, dass die Wahrscheinlichkeit erhöht ist, dass sich eine Person als einen anderen Benutzer ausgeben und authentifizieren kann.

Der dazu gegenteilige Ansatz fokussiert sich auf die Anzahl der falsch zugeordneten Samples. Um das System möglichst sicher zu halten, sollte diese Zahl minimal (im Optimalfall null) sein. Es wird also sichergestellt, dass sich keine Person als ein anderer Benutzer erfolgreich authentifizieren kann. Im Gegenzug kann dies jedoch bedeuten, dass die Anzahl an richtig zugewiesenen Samples sinkt, beziehungsweise die Anzahl der nicht zugewiesenen Samples steigt. Somit ist die Wahrscheinlichkeit erhöht, dass ein Authentifizierungsversuch trotz korrektem Sprecher fehlschlägt und gegebenenfalls öfters wiederholt werden muss.

Für die Bewertung der Ergebnisse zum Einsatz in dem in dieser Arbeit entwickelten Demosystem wird eine Kombination der beiden Ansätze verfolgt. Priorisiert wird hierbei der erste Ansatz, also die Orientierung an dem Datensatz mit den Meisten richtig zugeordneten Testdateien um eine gute Benutzerfreundlichkeit zu erzielen. Gleichzeitig wird aber auch darauf geachtet, dass die Anzahl der falsch zugeordneten Samples möglichst gering ist. Der Anzahl der nicht zugewiesenen Samples wird eine geringe Wichtigkeit zugeschrieben, da durch die Optimierung der zwei anderen Werte automatisch ein akzeptabler Wert gewährleistet ist. Außerdem sind die Folgen einer hohen Anzahl nicht zugeordneter Sample vergleichsweise gering, da zunächst keine Authentifikation stattfindet und die Authentifikation mit einem neuen Testsample wiederholt werden kann.

Unter Betrachtung der durchschnittlichen absoluten Genauigkeit, sowie der korrekt zugeteilten Testdateien (zweites Diagramm), ergeben sich die vier markierten Cluster als beste Konfigurationen. Auch in den zwei verbleibenden Kategorien, zeichnen sich diese Konfigurationen vor allem durch eine niedrige Anzahl an nicht zugeordneten Dateien (kleiner 25 %, viertes Diagramm), sowie falsch zugeordneter Dateien (kleiner 2 %, drittes Diagramm) aus.

Die Bildung der Cluster ist dabei auf die Art und Weise wie die Konfigurationen erzeugt werden zurückzuführen. Da hier eine bestimmte Systematik vorliegt, enthalten diese Cluster jeweils ähnliche Feature-Kombinationen, wobei die für die Ergebnisse relevanten Features in jeder Konfiguration des Clusters vorhanden sind. In den rot markierten Clustern sind dies MFCC und dMFCC Features.

In einer detaillierteren Analyse im Direktvergleich ergeben sich die in Tabelle 6.1 dargestellten Konfigurationen als beste Kombinationen:

ID	Durchschnittliche absolute Genauigkeit	LPC	MFCC	LPCC	dMFCC
472	0.8033	0	20	0	13
474	0.8015	0	20	13	13
410	0.8015	0	20	13	13
408	0.7989	0	20	0	13
414	0.7986	0	20	13	13

Tab. 6.1: Auswertung der Konfigurationen

Die Evaluation ergibt somit, dass die Konfiguration mit der ID 472 die besten Ergebnisse erzielt. Dabei werden 15000 Frames bei einer Frame-Größe von 600 Samples generiert. Daraufhin werden 20 MFCC Koeffizienten, sowie 13 dMFCC Koeffizienten pro Frame erzeugt, welche durch das neuronale Netz ausgewertet werden. LPC und LPCC zeigen in den ausgewählten Konfigurationen keinen signifikanten Mehrwert, weshalb diese nicht verwendet werden.

In einem weiteren Schritt werden basierend auf der Konfiguration 472 weitere Konfigurationen erstellt, um zusätzliche Untersuchungen durchzuführen. Dabei werden die Parameter Anzahl der Frames, Länge der Frames, sowie Anzahl der MFCC Koeffizienten um jeweils einen neuen Wert erweitert, da hier zu erkennen ist, dass eine Erhöhung dieser Werte zu einer Verbesserung des Gesamtergebnisses führt. Folgend werden die Parameterwerte 20000 Frames, 800 Samples pro Frame und 27 MFCC Koeffizienten evaluiert. Die Parameterverteilung der Konfigurationen sind in Tabelle 6.2 dargestellt.

ID	Anzahl Frames	Länge Frames	MFCC	dMFCC
511	20000	600	20	13
512	15000	800	20	13
513	15000	600	27	13
514	20000	800	20	13
515	20000	600	27	13
516	15000	800	27	13

Tab. 6.2: Zusätzliche Konfigurationen

Die Ergebnisse der Konfigurationen sind in Tabelle 6.3 aufgelistet. Da wie bereits beschrieben die Anzahl an richtig zugeordneten Samples in der Evaluierung bevorzugt wird, ergibt sich ein neues optimales Modell mit der Konfigurations-ID 516. Dabei kann ein Anstieg der richtig zugeordneten Samples um 2,6 Prozentpunkte verzeichnet werden. Diese Verbesserung ist damit auf die Erhöhung der Framelänge, sowie der Anzahl an MFCC Koeffizienten zurückzuführen.

ID	Durchschn. abs. Genauigkeit	Richtig zug.	Falsch zug.	Nicht zug.
472	0,8033	0,897	0,007	0,097
511	0,8123	0,885	0,005	0,110
512	0,8159	0,885	0,010	0,105
513	0,8083	0,880	**0,003**	0,117
514	**0,8416**	0,900	0,007	0,093
515	0,8190	0,877	0,007	0,117
516	0,8377	**0,923**	0,007	**0,070**
Diff 516-472	+0,0344	+0,026	0,000	-0,027

Tab. 6.3: Ergebnisse der zusätzlichen Konfigurationen

Dabei muss jedoch mit beachtet werden, dass mit dieser kleinen Verbesserung auch ein zusätzlicher Aufwand kommt. Auf der einen Seite wird eine längere Audioaufzeichnung zur Authentifizierung benötigt, da die Framelänge um 200 Samples erhöht wird. Auf der anderen Seite steigt auch der Rechenaufwand, da anstelle von 20 MFCC Koeffizienten nun 27 Koeffizienten berechnet werden müssen.

Unter Anbetracht des erhöhten Aufwands wird somit von einer weiterreichenden Analyse mit mehr als 27 Koeffizienten abgesehen. Da die Veränderungen, die zu dem verbesserten Ergebnis führen für die einmalige Berechnung des neuronalen Netzes des Demosystems tragbar sind, wird die für das Demosystem zu verwendende Konfiguration auf die Konfiguration mit der ID **516** festgelegt.

Um die Genauigkeit der Konfiguration für den Einsatz im Demosystem noch einmal zu erhöhen, wird das neuronale Netz erneut trainiert, wobei die Epochen von 250 auf 500 erhöht werden. Die Ergebnisse sind in Tabelle 6.4 aufgeführt. Unter Verwendung desselben Thresholds von 65 % wird eine weitere Verbesserung um 2,7 Prozentpunkte im Bereich der richtig zugeordneten Samples erreicht.

Threshold	Richtig zug.	Falsch zug.	Nicht zug.
65 %	0,95	0,01	0,04
70 %	0,90	0,00	0,10

Tab. 6.4: Ergebnisse des neuronalen Netz des Demosystems

Da die Anzahl an falsch zugeordneten Samples mit einem Prozentpunkt bereits sehr gering ist, kann hier durch eine Erhöhung des Thresholds das System in Bezug auf den Aspekt der Sicherheit weiter optimiert werden. Durch die Erhöhung von 65 % auf 70 % werden keine Samples mehr falsch zugeordnet, gleichzeitig verschlechtert sich jedoch auch die Anzahl an richtig zugeordneten Samples.

Die Verschlechterung lässt sich durch zwei Aspekte rechtfertigen. Auf der einen Seite ist es für Benutzer durch die Optimierung nicht mehr möglich, sich als eine andere Person zu authentifizieren. Auf der anderen Seite sorgt die erhöhte Anzahl nicht zugeordneter Samples lediglich dafür, dass in 10 % der Fälle eine erneute Aufzeichnung der Stimme für die Authentifizierung vonnöten ist. Dieser Aufwand ist in Relation zu der gewonnenen Sicherheit sehr gering.

7 Analyse der Sicherheitsrisiken

verfasst von: Johannes Brandenburger, Lukas Braun

In dem vorliegenden Kapitel wird die Analyse der Sicherheitsrisiken von Sprecherauthentifikationssystemen durchgeführt. Dazu werden mögliche Angriffsszenarien und entsprechende Gegenmaßnahmen vorgestellt. Dabei wird die Analyse zunächst auf ein allgemeines System und die erarbeiteten Sicherheitsrisiken dann auf das entwickelte Demosystem bezogen. Dies ist möglich, da das Demosystem, dessen Entwicklungen in den vorausgegangenen Kapiteln beschrieben wurde, von diesem allgemeinen System abgeleitet ist.

7.1 Allgemeine Sicherheitsrisiken

verfasst von: Lukas Braun

In diesem Abschnitt erfolgt die Analyse der Sicherheitsrisiken für ein allgemeines Sprecherauthentifikationssystem. Hierbei geht es speziell um Bedrohungen von Sprecherauthentifikationssystemen, allgemeine Risiken von IT-Systemen werden nicht näher beachtet. Ein solches System wird in Kapitel 4.1 dargestellt und erläutert. Grundsätzlich gibt es drei Arten von Sicherheitsrisiken für Sprecherauthentifikationssysteme: Voice-Spoofing, Backdoor-Attacken und Eavesdropping. Im Folgenden sind diese drei Sicherheitsrisiken dargestellt und erläutert.

Voice Spoofing bezieht sich auf die Nachahmung einer bestimmten Stimme oder die Manipulation einer Sprachaufnahme um so jemanden zu täuschen oder Zugriff auf ein System zu erhalten. Die Nachahmung der Stimme wird auch als synthetische Stimmimitation bezeichnet. Eine Untersuchung der Universität of Birmingham in Alabama zeigt, dass bereits wenige Minuten Audio des Opfers ausreichen, um die Stimme vollständig zu klonen. Auch die Optionen zur Beschaffung dieser Informationen werden präsentiert. Der Angreifer kann sich in der Nähe des Opfers aufhalten und Sprachaufnahmen machen, im Internet nach Aufnahmen suchen oder gezielt über Spam-Anrufe die benötigten Daten erhalten (vgl. Katherine Shonesy 2015).

Die Manipulation einer Sprachaufnahme bedeutet das Anpassen, beziehungsweise das Bearbeiten, des ursprünglichen Audiosignals. Guangke Chen setzt in seinem Angriffssystem „FakeBob“ auf Pertubation. Hierbei wird auf ein Audio Quellsignal eine Störung angewandt, die es ermöglicht, dass das attackierte Sprecherauthentifikationssystem einen anderen Benutzer authentifiziert. Dabei ist es für einen Menschen nicht möglich, einen Unterschied zwischen der originalen Aufnahme und der veränderten Aufnahme wahrzunehmen, wodurch der Angriff vertuschbar ist. „FakeBob“ attackiert hierbei im Black-Box Modus und hat somit lediglich Zugang zum Authentifizierungsergebnis und bei zwei von drei Versuchssystemen auf die Ergebnisverteilung. Im Rahmen der Arbeit wird „FakeBob“ auf das Open-Source System „Kaldi“ und auf die kommer-

ziellen Systeme „Talentedsoft“ und „Microsoft Azure“ angewandt. „FakeBob“ erzielt hierbei eine 100 % Attack-Succes-Rate bei „Kaldi“ und „Talentedsoft“, welche sowohl die Verteilung als auch die Entscheidung bereitstellen. Bei „Microsoft Azure“ steht nur die Entscheidung zur Verfügung. Hier erreicht „FakeBob“ eine Attack-Success-Rate von 26 % (vgl. Chen u. a. 2020). Daraus folgt, dass Systeme die keine Verteilung bereitstellen sicherer gegen die Attacke von „FakeBob“ sind.

Um biometrischen Spoofing Attacken entgegenzuwirken, gibt es eine sogenannte Presentation Attack Detection (PAD). PAD verwendet verschiedene Techniken und Algorithmen, um gefälschte oder vorab aufgezeichnete Sprachaufnahmen zu identifizieren (vgl. Paravision 2022)

Backdoor Attacken beschreiben das Vorgehen, bei dem der Hacker gezielt die Trainingsdaten verändert, um somit eine Hintertür zu schaffen, über die das System infiltriert werden kann. Die Untersuchungen von Tongqing Zhai zeigen ein solches Verfahren auf. Hierzu bearbeitet der Hacker den Trainingsdatensatz, ohne Kenntnis über den eigentlichen Anmeldungsprozess. In der Arbeit werden unterschiedliche Ansätze zur Veränderung auf zwei Datensätzen durchgeführt. Hierbei werden Attack Success Rates zwischen mindestens 45 % und maximal 99,5 % erreicht (vgl. Zhai u. a. 2021)

Um sich vor Backdoor Attacken zu schützen, kann die Integrität des Datensatzes überprüft werden, um so Manipulationen zu identifizieren. Dies kann beispielsweise durch ein Prüfsummen Verfahren realisiert werden.

Eavesdropping bezeichnet im Allgemeinen das Abhören. Dies kann in Bezug auf Sprecherauthentifikationssysteme auf unterschiedliche Art und Weise durchgeführt werden. Grundsätzlich kann der zu authentifizierende Sprecher direkt aufgenommen werden. Hierzu eignen sich beispielsweise Telefongespräche oder Fake-Interviews. Diese Aufnahme kann im Anschluss zur Authentifizierung am System verwendet werden. Alternativ können Angreifer auf online Sprach- bzw. Videoaufnahmen zurückgreifen.

Ein weiterer Ansatz ist das klassische Eavesdropping, hierbei spioniert der Angreifer die Kommunikationsverbindung zwischen Benutzer und System aus und zeichnet diese auf. Mit dieser Aufzeichnung kann er sich zu späterem Zeitpunkt ebenfalls am System anmelden.

Um oben genannte Eavesdropping Angriffe zu vermeiden, kann vor das Sprecherauthentifikationssystem eine Texterkennung geschaltet werden. Somit kann durch Vorgabe eines Satzes überprüft werden, ob es sich um eine Aufnahme handelt. Das klassische Eavesdropping lässt sich durch die Verwendung von verschlüsselten Kommunikationsverbindungen vermeiden.

7.1.1 Einordnen in die STRIDE-Kategorien

verfasst von: Johannes Brandenburger, Lukas Braun

Im Folgenden werden die oben genannten Sicherheitsrisiken in die STRIDE-Kategorien eingeordnet. Wie bereits oben genannt, werden lediglich spezielle Bedrohungen von Sprecherauthentifikationssystemen behandelt. STRIDE steht für sechs verschiedene Kategorien zur Bewertung von Sicherheitsrisiken für IT-Systeme (s. Kapitel 2.6.1).

Spoofing Wie im Namen enthalten fällt unter Spoofing das Voice Spoofing. Das Risiko besteht darin, dass die Stimme eines autorisierten Sprechers nachgeahmt wird, oder eine manipulierte Sprachaufnahme verwendet wird.

Tampering Tampering bedeutet zu Deutsch Manipulation. Deshalb fällt das Voice Spoofing ebenfalls in die Kategorie Tampering. Ebenso fällt die Backdoor Attacke in diese Kategorie, da hierbei die Trainingsdaten gezielt manipuliert werden.

Repudiation Repudiation bezieht sich auf die Fähigkeit Aktionen innerhalb des Systems zu leugnen, somit kann die Integrität des Systems infrage gestellt werden. Diese Kategorie findet jedoch auf allgemeine Bedrohungen für Sprecherauthentifikationssystemen keine Anwendung.

Information Disclosure Information disclosure beschreibt die Offenlegung von vertraulichen Daten. Dies kann durch Eavesdropping Angriffe auftreten, da ein Angreifer durch Aufzeichnung von Authentifizierungsinformationen Zugang zum System erhalten kann.

Denial of Service Denial of service bedeutet „Verweigerung des Dienstes“. Bei einer solchen Attacke wird gezielt das System überlastet, um einen Ausfall zu provozieren. Diese Kategorie findet wie Repudiation keine Anwendung auf allgemeine Bedrohungen für Sprecherauthentifikationssystemen.

Elevation of Privilege Elevation of privilege beschreibt das Erlangen zusätzlicher Berechtigungen um so Zugang zu weiteren Ressourcen zu erhalten. In diese Kategorie fällt die Backdoor Attacke, da der Angreifer gezielt die Trainingsdaten verändert, um eine Hintertür zu schaffen, sodass er das System infiltrieren kann.

7.1.2 DREAD-Analyse

verfasst von: Johannes Brandenburger, Lukas Braun

Im vorliegenden Kapitel werden die zuvor ermittelten Risiken mit der DREAD-Methodik bewertet. Die DREAD-Methodik ermöglicht den Schweregrad bzw. das Risiko des Angriffs zu messen. Die Ergebnisse sind in der Tabelle 7.1 dargestellt.

ID	Grundsätzliches Sicherheitsrisiko	**Attackierungs-möglichkeit**	STRIDE Ein-stufung	Damage	Repro-ducibility	Ex-ploitabili-ty	Affected Users	Discover-ability	**Average**
1	Voice Spoofing	**Synthetische Stimme**	ST	8	8	3	1	3	**4,6**
2	Voice Spoofing	**Manipulation der Stimme durch Störung**	ST	8	7	3	1	5	**4,8**
3	Backdoor Attacke	**Manipulation der Trainingsdaten für Backdoor**	TE	9	3	2	10	4	**5,6**
4	Eavesdropping	**Aufzeichnen des Opfers**	I	8	8	8	1	10	**7**
5	Eavesdropping	**Verwenden von vorhandenen Sprach- oder Video-aufnahmen**	I	8	6	10	1	10	**7**
6	Eavesdropping	**Aufzeichnen der Verbindung zwischen Client und Server**	I	8	5	5	1	9	**5,6**

Tab. 7.1: DREAD-Analyse

Die Tabelle stellt die STRIDE-Einstufung aus dem vorangegangenen Kapitel und die DREAD-Methodik dar. In der letzten Spalte werden die Ergebnisse der Untersuchung aufgezeigt, hierbei steht eine hohe Zahl für ein hohes Risiko, für dieses Szenario. Bei einer Entwicklung als Produktivsystem sollten diese Risiken beachtet werden und durch die ebenfalls vorhin genannten Möglichkeiten zum Schutz vor solchen Angriffen reduziert werden.

7.2 Bezug zum Demosystem

verfasst von: Johannes Brandenburger, Lukas Braun

Das in Kapitel 5.3 entwickelte Demosystem unterscheidet sich gravierend von einer Produktivlösung. Dieses System wurde ausschließlich dazu entwickelt, um die Ergebnisse aus dem Versuchssystem anhand eines praktischen Authentifizierungsablaufs darzustellen und kann auf keinen Fall mit einem ausgereiften System verglichen werden. Die oben genannte Sicherheitsrisiken existieren somit nicht für das Demosystem, da zum einen keine externen Eingaben wie Stimm-Aufnahmen ins System gegeben werden können und das System zum anderen keine persönlichen oder kritischen Daten schützt.

Wenn das System als Produktivlösung implementiert werden würde, müssten die oben genannten Maßnahmen zur Risikominimierung berücksichtigt werden. Zusätzlich müssten zu den oben genannten speziellen Sicherheitsrisiken auch allgemeine Sicherheitsrisiken für ein solches Authentifikationssystem bewertet werden.

8 Fazit und kritische Reflexion

verfasst von: Henry Schuler

Im Rahmen dieser Arbeit wurde ein Versuchssystem, sowie ein darauf aufbauendes Demosystem zunächst konzipiert und anschließend implementiert. Ausgehend von einer ausführlichen Grundlagenrecherche, wurden dabei zunächst die Features LPC, LPCC, MFCC und dMFCC in Kombination mit einem NN als geeignete Kandidaten für die Umsetzung einer Sprecherauthentifikation ermittelt. In einem weiteren Schritt ergab sich daraus sowohl die grundlegende Architektur für ein Sprecherauthentifikationssystem, als auch eine spezifische Architektur für das Versuchs- und Demosystem. Dazu wurden zunächst Systemanforderungen definiert, um die zu entwickelnden Systeme genauer einzugrenzen und grundlegende Entscheidungen zu treffen. In einer ausführlichen Technologieentscheidung, wurden verschiedene Technologien miteinander verglichen und schlussendlich die Verwendung von Python in Kombination mit TensorFlow (Versuchssystem/Back-End), sowie React (Front-End) begründet. Diese bildetet die Grundlage für die anschließende Implementierung der beiden Systeme. Die Ergebnisse der Durchführung des Versuchssystems sind in der Evaluation ausführlich beschrieben und führen zu der Schlussfolgerung, dass MFCC und dMFCC für die Sprecherauthentifikation relevant sind, während LPC und LPCC keine gewinnbringenden Ergebnisse erzielen. Die ermittelte Konfiguration zum Einsatz im Demosystem authentifiziert dabei in 90 % der Fälle die richtige, in 10 % keine und in 0 % die falsche Person. Aufbauend auf diesem Ergebnis, wurde abschließend eine Sicherheitsbewertung in Form einer Threat-Analyse durchgeführt, welche verfahrensspezifische Sicherheitsrisiken erläutert und bewertet.

Insgesamt konnte erfolgreich gezeigt werden, dass eine Sprecherauthentifikation innerhalb einer fest definierten Gruppe an Benutzern nicht nur möglich, sondern auch zuverlässig umsetzbar ist. Die abschließende Sicherheitsbewertung zeigt aber auf, dass Sprecherauthentifikation aufgrund der vielen verschiedenen Risiken nicht sicher in der Praxis eingesetzt werden kann. Durch die zusätzliche Integration des Authentifizierungsprozesses in das Demosystem, wurde das in Kapitel 1.2 definierte Ziel dieser Arbeit somit erreicht.

Das entwickelte Demosystem kann zur Präsentation des in dieser Arbeit erzielten Ergebnisses im Informatik-Labor der Hochschule bereitgestellt werden. Dies ermöglicht das eigenständige Verifizieren und Nachvollziehen der Ergebnisse durch andere Studenten. Gleichzeitig bietet diese Arbeit eine Grundlage für weiterführende anschließende Studienarbeit. Potenzielle Themenbereiche werden dabei im anschließenden Kapitel (Ausblick) vorgestellt.

Wirft man einen kritischen Blick auf diese Arbeit, so kann zunächst angemerkt werden, dass es sich bei den in diesem System verwendeten Audio-Aufzeichnungen nicht um Beispiele eines Realeinsatzes handelt. Die verwendeten Aufzeichnungen von Hörbüchern sind unter idealen Be-

dingungen erstellt worden, wodurch sich die Qualität der verschiedenen Audio-Aufzeichnungen nur geringfügig unterscheidet. Zusätzlich sind alle Audio-Segmente aus einem kontinuierlichen Redefluss entnommen, wodurch Schwankungen der Stimme nur begrenzt auftreten. Eine Integration des Systems in ein Produktivsystem erreicht somit aufgrund schwankender Qualitätsunterschiede der Stimmaufzeichnung gegebenenfalls schlechtere Ergebnisse.

Auch eine Vergleichbarkeit zu anderen Studien ist nur begrenzt möglich, da das System verschiedene Faktoren enthält, die das Ergebnis maßgeblich beeinflussen. Dazu zählt unter anderem der verwendete Datensatz, der Aufbau des NN und die Art und Weise der Durchführung und Evaluation. Diese Tatsache beeinflusst auch den Detailgrad das Kapitels Stand der Technik, da hier auf die Ergebnisse der aufgelisteten Studien aufgrund der fehlenden Vergleichbarkeit nicht näher eingegangen wird.

9 Ausblick

verfasst von: Henry Schuler

Wie bereits im vorangehenden Kapitel beschrieben, wurden in dieser Arbeit ein grundlegendes System für die Sprecherauthentifikation erarbeitet. In dieser Arbeit wird dabei nur ein kleiner Teilaspekt der Sprecherauthentifikation genauer untersucht, während andere Aspekte nicht genauer betrachtet werden. Diese Aspekte können im Rahmen einer anschließenden Studie genauer untersucht werden. Eine Auswahl möglicher Vertiefungen ist in den folgenden Unterkapiteln beschrieben.

9.1 Praktische Evaluation der Sicherheitsrisiken

Die in Kapitel 7 aufgezeigten Risiken können im Rahmen einer anschließenden Arbeit genauer evaluiert werden. Dazu werden ausgewählte Angriffspunkte konzipiert und implementiert. In einer ausführlichen Durchführung kann die in dieser Arbeit getroffene Einstufung der Sicherheitsrisiken verifiziert, beziehungsweise widerlegt werden. Gleichzeitig bietet die Arbeit die Möglichkeit zur Entwicklung einer Gegenmaßnahme, sodass diese Sicherheitslücken geschlossen werden. Denkbar wäre hier auch ein vorher-nachher Vergleich zur Evaluierung der Effektivität der entwickelten Gegenmaßnahme.

9.2 Weiterführende Evaluierung der in dieser Arbeit verwendeten Merkmale

In der Evaluation (Kapitel 6) wurde die Aussage getroffen, dass eine weiterführende Analyse der verwendeten MFCC und dMFCC Merkmalen unter Betracht der geringen Verbesserung über den Rahmen dieser Arbeit hinaus geht. Im Rahmen weiterer Arbeiten kann dieser Punkt aufgegriffen werden und gezielt die zu verwendende Koeffizientenanzahl, sowie Framegröße und -länge evaluiert werden. Um den Aspekt des Rechenaufwands dabei nicht zu vernachlässigen, kann eine Metrik erstellt werden, die die Erhöhung der Koeffizientenanzahl und damit auch des Rechenaufwands mit der erzielten Verbesserung ins Verhältnis setzt.

9.3 Optimierung durch Anpassung des Neuronalen Netzes

Der in dieser Arbeit verwendete Aufbau des NN wurde zu Beginn festgelegt und nicht näher untersucht. Die Struktur des NN spielt dabei jedoch eine zentrale Rolle in der Klassifikation durch das NN. Mittels detaillierterer Recherchen im Bereich der NN sowie zusätzliche Versuchsreihen, kann evaluiert werden, ob eine Verbesserung der Authentifizierung durch eine andere NN-Struktur erreichbar ist.

Ein weiterer möglicher Aspekt in diesem Zusammenhang stellt die Art und Weise der Klassifizierung durch das NN dar. In dieser Arbeit wird mittels des NN jedem Frame eine SprecherId zugeordnet, wobei für jeden Sprecher ein Output-Neuron existiert. Zu überprüfen ist, ob eine Klassifikation als „authentifiziert“, beziehungsweise „nicht authentifiziert“ direkt durch das NN vorgenommen werden kann. Dabei ist zu entscheiden, ob als Resultat für jeden Sprecher ein separates NN erzeugt, oder ob der Input-Layer lediglich angepasst werden muss.

9.4 Dynamische Erweiterbarkeit um zusätzliche Sprecher

Das in dieser Arbeit implementierte System ist für die Verwendung mit einem fest definierten Datensatz ausgelegt. Um das System an die Anwendungsfälle eines Produktiveinsatzes anzupassen, muss das System die Möglichkeit bieten, neue Benutzer hinzuzufügen. Die Art und Weise wie dies ermöglicht werden kann ist dabei Teil einer weiterführenden Arbeit. Hierbei kann neben der theoretischen Umsetzung auch der Punkt betrachtet werden, wie sich die Zuverlässigkeit mit einer zunehmenden Anzahl an Sprechern verändert und was die Auswirkungen für bestehende, sowie neue Benutzer sind.

Gleichzeitig gewinnt auch die benötigte Länge an Audiomaterial für den Trainingsprozess von Bedeutung. Hier kann einerseits der allgemeine Zusammenhang zwischen der Länge des zur Verfügung stehenden Audiomaterials und der Zuverlässigkeit des NN untersucht werden. Zusätzlich kann aber auch die Relevanz der Länge des zur Verfügung stehenden Audiomaterials bei einer steigenden Anzahl an Benutzern untersucht werden.

9.5 Zuverlässigkeitsanalyse ähnlicher Sprecher

Für diese Arbeit wurde ein Datensatz ausgewählt, welcher Sprecher enthält, die sich akustisch durch die Evaluierung eines Menschen eindeutig auseinanderhalten lassen. Dies ist im Realeinsatz jedoch nicht immer der Fall. Es besteht die Möglichkeit, dass sich Stimmen verschiedener Sprecher nur minimal unterscheiden (beispielsweise bei Zwillingen). Durch eine Untersuchung kann evaluiert werden, inwiefern das System in der Lage ist, diese Personen korrekt zu authentifizieren und damit auseinanderzuhalten. Neben der Evaluation des Systems aus dieser Arbeit gilt es dabei zu erörtern, ob es spezielle Merkmale gibt, die sich in besonderer Weise für die Authentifizierung ähnlicher Sprecher eignen.

9.6 Qualität und Inhalt des Audiomaterials

Der in dieser Arbeit verwendete Datensatz besteht ausschließlich aus Aufzeichnungen englischer Sprache, die unter gleicher Bedingung aufgenommen sind. Im realen Einsatz besteht die

Möglichkeit, dass sowohl andere Sprachen, als auch verschiedene Aufnahmequalitäten auftreten. Diese beiden Punkte können getrennt voneinander in einer weiterführenden Arbeit evaluiert werden. Dabei kann bei der verwendeten Sprache sowohl eine Bewertung zwischen mehreren Systemen, die jeweils eine Sprache verwenden, durchgeführt werden, als auch innerhalb eines Systems, welches mehrere Sprecher mit unterschiedlichen Sprachen implementieren. Für den Aspekt der Aufnahmequalität können einerseits technische Einflüsse wie Störungen oder unterschiedliche Mikrofone betrachtet werden oder andererseits Störgeräusche in Form von Umgebungslärm, zum Beispiel durch Stimmen mehrerer Personen im Hintergrund.

Literatur

Anz, Philipp (Feb. 2023). *Mit KI-Stimme ins Bankkonto eingedrungen*. de. URL: https://www.inside-it.ch/mit-ki-stimme-ins-bankkonto-eingedrungen-20230227 (besucht am 22. 06. 2023).

Atal, B. S. (Juni 1974). „Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification". en. In: *The Journal of the Acoustical Society of America* 55.6, S. 1304–1312. ISSN: 0001-4966. DOI: 10.1121/1.1914702. URL: http://asa.scitation.org/doi/10.1121/1.1914702 (besucht am 16. 02. 2023).

Authentifizierung mit Stimmerkennung (o. D.). de. URL: https://www.postfinance.ch/de/support/daten/persoenliche-daten/authentifizierung-stimmerkennung.html (besucht am 25. 06. 2023).

Backhaus, Klaus u. a. (2016). *Multivariate Analysemethoden: eine anwendungsorientierte Einführung*. ger. 14., überarbeitete und aktualisierte Auflage. Lehrbuch. Berlin [Heidelberg]: Springer Gabler. ISBN: 978-3-662-46076-4 978-3-662-46075-7.

Beucher, Ottmar (2011). *Signale und Systeme: Theorie, Simulation, Anwendung Eine beispielorientierte Einführung mit MATLAB*. ger. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-20294-0.

Canela, Miguel Ángel, Inés Alegre und Alberto Ibarra (2019). „Multiple Regression". en. In: *Quantitative Methods for Management*. Cham: Springer International Publishing, S. 37–45. ISBN: 978-3-030-17553-5 978-3-030-17554-2. DOI: 10.1007/978-3-030-17554-2_4. URL: http://link.springer.com/10.1007/978-3-030-17554-2_4 (besucht am 17. 02. 2023).

Chelali, Fatma Zohra und Amar Djeradi (Sep. 2017). „Text dependant speaker recognition using MFCC, LPC and DWT". en. In: *International Journal of Speech Technology* 20.3, S. 725–740. ISSN: 1381-2416, 1572-8110. DOI: 10.1007/s10772-017-9441-1. URL: http://link.springer.com/10.1007/s10772-017-9441-1 (besucht am 23. 01. 2023).

Chen, Guangke u. a. (Apr. 2020). *Who is Real Bob? Adversarial Attacks on Speaker Recognition Systems*. arXiv:1911.01840 [cs, eess]. URL: http://arxiv.org/abs/1911.01840 (besucht am 12. 07. 2023).

Clauss, Wolfgang und Cornelia Clauss (2018). *Humanbiologie kompakt*. ger. 2. Auflage. Lehrbuch. Berlin: Springer Spektrum. ISBN: 978-3-662-55850-8 978-3-662-55849-2. DOI: 10.1007/978-3-662-55850-8.

Daniel John (Mai 2022). *Neuronales Netz · Seite · HOOU*. DE. URL: https://www.hoou.de/projects/neuronale-netze-kurz-erklart/pages/neuronales-netz (besucht am 24. 05. 2023).

DOMARS (März 2023). *Threat modeling for drivers - Windows drivers*. en-us. URL: https://learn.microsoft.com/en-us/windows-hardware/drivers/driversecurity/threat-modeling-for-drivers (besucht am 04. 07. 2023).

Ertel, Wolfgang (2016). *Grundkurs Künstliche Intelligenz: eine praxisorientierte Einführung*. ger. 4., überarbeitete Auflage. Computational Intelligence. Wiesbaden: Springer Vieweg. ISBN: 978-3-658-13548-5.

Etaati, Leila (2019). *Machine Learning with Microsoft Technologies: Selecting the Right Architecture and Tools for Your Project*. en. Berkeley, CA: Apress. ISBN: 978-1-4842-3657-4 978-1-4842-3658-1. DOI: 10.1007/978-1-4842-3658-1. URL: http://link.springer.com/10.1007/978-1-4842-3658-1 (besucht am 26. 04. 2023).

Guresen, Erkam und Gulgun Kayakutlu (2011). „Definition of artificial neural networks with comparison to other networks“. en. In: *Procedia Computer Science* 3, S. 426–433. ISSN: 18770509. DOI: 10.1016/j.procs.2010.12.071. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050910004461 (besucht am 17. 05. 2023).

IDC (Nov. 2022). *Künstliche Intelligenz - Umsatz weltweit bis 2024*. de. URL: https://de.statista.com/statistik/daten/studie/1211850/umfrage/umsatz-im-bereich-kuenstliche-intelligenz-weltweit/ (besucht am 25. 02. 2023).

Jain, Vibhor (Aug. 2019). *Speaker Recognition Audio Dataset*. en. URL: https://www.kaggle.com/datasets/vjcalling/speaker-recognition-audio-dataset (besucht am 07. 07. 2023).

Janson, Matthias (Jan. 2023). *Infografik: ChatGPT's Sprint zu einer Million Nutzer:innen*. de. URL: https://de.statista.com/infografik/29195/zeitraum-den-online-dienste-gebraucht-haben-um-eine-million-nutzer-zu-erreichen (besucht am 25. 02. 2023).

Judith Hurwitz und Daniel Kirsch (2018). *Machine Learning for Dummies, IBM Limited Edition*. Hoboken: John Wiley & Sons.

Katherine Shonesy (Sep. 2015). *UAB research finds automated voice imitation can fool humans and machines*. en-US. URL: https://www.uab.edu/news/research/item/6532-uab-research-finds-automated-voice-imitation-can-fool-humans-and-machines (besucht am 12. 07. 2023).

Keras (2023). *Keras: The high-level API for TensorFlow | TensorFlow Core*. en. URL: https://www.tensorflow.org/guide/keras (besucht am 07. 07. 2023).

Kiapuchinski, Davi Miara, Carlos Raimundo Erig Lima und Celso Antonio Alves Kaestner (Dez. 2012). „Spectral Noise Gate Technique Applied to Birdsong Preprocessing on Embedded Unit“. In: *2012 IEEE International Symposium on Multimedia*. Irvine, CA, USA: IEEE, S. 24–27. ISBN: 978-1-4673-4370-1 978-0-7695-4875-3. DOI: 10.1109/ISM.2012.12. URL: http://ieeexplore.ieee.org/document/6424625/ (besucht am 27. 02. 2023).

Kröger, Bernd J. (2018). *Neuronale Modellierung der Sprachverarbeitung und des Sprachlernens: eine Einführung*. ger. 1. Auflage. Berlin [Heidelberg]: Springer Spektrum. ISBN: 978-3-662-55459-3 978-3-662-55458-6.

Kumar Rajeev u. a. (2009). „MULTILINGUAL SPEAKER RECOGNITION USING NEURAL NETWORK“. In.

Li, Haizhou, Kar-Ann Toh und Liyuan Li (2011). *Advanced topics in biometrics*. OCLC: 5011576346. ISBN: 978-981-4287-85-2.

Li, Qi (2012). *Speaker authentication*. Signals and communication technology. OCLC: ocn778697291. Heidelberg ; New York: Springer. ISBN: 978-3-642-23730-0.

librosa development team (2023). *librosa — librosa 0.10.0 documentation*. URL: https://librosa.org/doc/latest/index.html (besucht am 11. 07. 2023).

Logan, Beth (2000). „Mel Frequency Cepstral Coefficients for Music Modeling“. In.

Lokesh, S. und M. Ramya Devi (Sep. 2019). „Speech recognition system using enhanced mel frequency cepstral coefficient with windowing and framing method“. en. In: *Cluster Computing* 22.S5, S. 11669–11679. ISSN: 1386-7857, 1573-7543. DOI: 10.1007/s10586-017-1447-6. URL: http://link.springer.com/10.1007/s10586-017-1447-6 (besucht am 07. 02. 2023).

Marcel Mikl (Aug. 2018). *Wie trainiert man eigentlich neuronale Netze?* URL: https://www.codecentric.de/wissens-hub/blog/wie-trainiert-man-eigentlich-neuronale-netze (besucht am 03. 06. 2023).

Marple, L. (Aug. 1980). „A new autoregressive spectrum analysis algorithm“. en. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4, S. 441–454. ISSN: 0096-3518. DOI: 10.1109/TASSP.1980.1163429. URL: http://ieeexplore.ieee.org/document/1163429/ (besucht am 17. 02. 2023).

Maschke, Christian und André Jakob (2018). „Psychoakustische Messtechnik“. de. In: *Psychoakustische Messtechnik*. Hrsg. von Michael Möser. Series Title: Fachwissen Technische Akustik. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 1–34. ISBN: 978-3-662-56630-5 978-3-662-56631-2. DOI: 10.1007/978-3-662-56631-2_1. URL: http://link.springer.com/10.1007/978-3-662-56631-2_1 (besucht am 22. 04. 2023).

Meine Stimme – mein Passwort | Telekom Hilfe (o. D.). de. URL: https://www.telekom.de/hilfe/sprach-id?samChecked=true (besucht am 25. 06. 2023).

MSV, Janakiram (Nov. 2020). *TensorFlow Turns 5 - Five Reasons Why It Is The Most Popular ML Framework*. en. URL: https://www.forbes.com/sites/janakirammsv/2020/11/27/tensorflow-turns-5five-reasons-why-it-is-the-most-popular-ml-framework/ (besucht am 07. 07. 2023).

Neha Chauhan, Dongju Li und Tsuyoshi Isshiki (2019). *2019 IEEE 4th International Conference on Computer and Communication Systems: ICCCS 2019 : February 23-25, 2019, Singapore*. eng. OCLC: 1119977808. Piscataway, NJ: IEEE Press. ISBN: 978-1-72811-322-7.

numpy.hanning — NumPy v1.24 Manual (2022). URL: https://numpy.org/doc/stable/reference/generated/numpy.hanning.html (besucht am 07. 02. 2023).

Oppenheim, Alan V., Ronald W. Schafer und John R. Buck (1999). *Discrete-time signal processing*. 2nd ed. Upper Saddle River, N.J: Prentice Hall. ISBN: 978-0-13-754920-7.

Paravision (Mai 2022). *An Introduction to Presentation Attack Detection (PAD)*. en. URL: https://www.paravision.ai/whitepaper-an-introduction-to-presentation-attack-detection/ (besucht am 12. 07. 2023).

Pfister, Beat und Tobias Kaufmann (2017). *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. ger. 2., aktualisierte und erweiterte Auflage. Berlin; [Heidelberg]: Springer Vieweg. ISBN: 978-3-662-52838-9 978-3-662-52837-2.

Picard, Rainer (1996). „Fourier-Analyse“. In: DOI: 10.25521/HQM27. URL: https://madoc.bib.uni-mannheim.de/50958/ (besucht am 27. 01. 2023).

Reynolds, D.A. und R.C. Rose (Jan. 1995). „Robust text-independent speaker identification using Gaussian mixture speaker models“. In: *IEEE Transactions on Speech and Audio Processing* 3.1, S. 72–83. ISSN: 10636676. DOI: 10.1109/89.365379. URL: http://ieeexplore.ieee.org/document/365379/ (besucht am 18. 04. 2023).

Rich, Elaine (1983). *Artificial intelligence*. McGraw-Hill series in artificial intelligence. New York: McGraw-Hill. ISBN: 978-0-07-052261-9.

Richter, Michael u. a. (2022). *Signal processing and machine learning with applications*. eng. OCLC: 1347386653. Cham: Springer. ISBN: 978-3-319-45372-9.

Rieß, Bernhard und Christoph Wallraff (2018). „Fourier-Reihe“. de. In: *Übungsbuch Signale und Systeme*. Wiesbaden: Springer Fachmedien Wiesbaden, S. 19–32. ISBN: 978-3-658-19225-9 978-3-658-19226-6. DOI: 10.1007/978-3-658-19226-6_2. URL: http://link.springer.com/10.1007/978-3-658-19226-6_2 (besucht am 07. 02. 2023).

Sainburg, Tim (Juni 2019). *timsainb/noisereduce: v1.0*. DOI: 10.5281/ZENODO.3243139. URL: https://doi.org/10.5281/zenodo.3243139 (besucht am 08. 07. 2023).

Shostack, Adam (2014). *Threat modeling: designing for security*. OCLC: ocn855043351. Indianapolis, IN: Wiley. ISBN: 978-1-118-80999-0.

Smith, Steven W. (1997). *The scientist and engineer's guide to digital signal processing*. 1st ed. San Diego, Calif: California Technical Pub. ISBN: 978-0-9660176-3-2.

Sonnet, Daniel (2022). *Neuronale Netze kompakt: Vom Perceptron zum Deep Learning*. de. IT kompakt. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 978-3-658-29080-1 978-3-658-29081-8. DOI: 10.1007/978-3-658-29081-8. URL: https://link.springer.com/10.1007/978-3-658-29081-8 (besucht am 24. 05. 2023).

Stack Overflow (2020). *Stack Overflow Developer Survey 2020*. URL: https://insights.stackoverflow.com/survey/2020/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2020 (besucht am 07. 07. 2023).

– (2022). *Stack Overflow Developer Survey 2022*. Stack Overflow. URL: https://survey.stackoverflow.co/2022/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2022 (besucht am 11. 07. 2023).

Subsets of AI - Javatpoint (o. D.). en. URL: https://www.javatpoint.com/subsets-of-ai (besucht am 24. 04. 2023).

Thakkar, Mohit (2019). *Beginning Machine Learning in iOS: CoreML Framework*. en. Berkeley, CA: Apress. ISBN: 978-1-4842-4296-4 978-1-4842-4297-1. DOI: 10.1007/978-1-4842-4297-1. URL: http://link.springer.com/10.1007/978-1-4842-4297-1 (besucht am 26. 04. 2023).

Thullier, Florentin, Bruno Bouchard und Bob-Antoine J. Menelas (Dez. 2017). „A Text-Independent Speaker Authentication System for Mobile Devices“. en. In: *Cryptography* 1.3, S. 16. ISSN: 2410-387X. DOI: 10.3390/cryptography1030016. URL: https://www.mdpi.com/2410-387X/1/3/16 (besucht am 18. 04. 2023).

Tsolkas, Alexander und Klaus Schmidt (2017). *Rollen und Berechtigungskonzepte*. de. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 978-3-658-17986-1 978-3-658-17987-8. DOI: 10.1007/978-3-658-17987-8. URL: http://link.springer.com/10.1007/978-3-658-17987-8 (besucht am 22. 06. 2023).

Valerio Velardo (Okt. 2020). *Mel-Frequency Cepstral Coefficients Explained Easily*. URL: https://github.com/musikalkemist/AudioSignalProcessingForML/blob/master/19-%20MFCCs%20Explained%20Easily/Mel-Frequency%20Cepstral%20Coefficients%20Explained%20Easily.pdf.

Vision Mobile (2017). „State of the Developer Nation Q1 2017“. In: Vision Mobile. URL: https://mwc.gr/presentations/2017/konstantinou.pdf (besucht am 07. 07. 2023).

Wu, Minshun, Degang Chen und Guican Chen (Mai 2012). „New Spectral Leakage-Removing Method for Spectral Testing of Approximate Sinusoidal Signals“. In: *IEEE Transactions on Instrumentation and Measurement* 61.5, S. 1296–1306. ISSN: 0018-9456, 1557-9662. DOI: 10.1109/TIM.2011.2180971. URL: http://ieeexplore.ieee.org/document/6134664/ (besucht am 07. 03. 2023).

Yepis, Erin (Juni 2023). *2023 Developer Survey results are in: the latest trends in technology and work from the Stack Overflow community*. en-US. URL: https://stackoverflow.blog/2023/06/13/developer-survey-results-are-in/ (besucht am 09. 07. 2023).

Zhai, Tongqing u. a. (Feb. 2021). *Backdoor Attack against Speaker Verification*. arXiv:2010.11607 [cs, eess]. URL: http://arxiv.org/abs/2010.11607 (besucht am 12. 07. 2023).

Zulfiqar, Ali u. a. (2010). „Text-Independent Speaker Identification Using VQ-HMM Model Based Multiple Classifier System“. In: *Advances in Soft Computing*. Hrsg. von Grigori Sidorov, Arturo Hernández Aguirre und Carlos Alberto Reyes García. Bd. 6438. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 116–125. ISBN: 978-3-642-16772-0 978-3-642-16773-7. DOI: 10.1007/978-3-642-16773-7_10. URL: http://link.springer.com/10.1007/978-3-642-16773-7_10 (besucht am 23. 01. 2023).