Zurich University
of Applied Sciences

School of
Engineering

CAI Centre for
Artificial Intelligence

Bachelorarbeit (Informatik)

Auswirkungen von Speech Enhancement auf die automatische Spracherkennung

Autoren	Manuel Berweger Marvin Tseng
Hauptbetreuung	Prof. Dr. Mark Cieliebak Nicola Good
Datum	09.06.22

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)
Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	**Name Studierende:**
Saland, 09.06.22	Manuel Berweger
Greifensee, 09.06.22	Marvin Tseng

Zusammenfassung

Die automatische Spracherkennung funktioniert zuverlässig bei sauberen Audioaufnahmen. Allerdings beeinträchtigen Hintergrundgeräusche und Nachhall im Audio die Transkriptionsqualität erheblich. In dieser Bachelorarbeit werden Speech Enhancement Verfahren im Kontext der automatischen Spracherkennung (ASR) untersucht. Ziel ist es, die Genauigkeit von automatischen Spracherkennungssystemen in Umgebungen mit Hintergrundgeräuschen und Nachhall zu verbessern. Um dieses Ziel zu erreichen, werden verschiedene Speech-to-Text (STT) Engines und Speech Enhancement Präprozessoren quantitativ untersucht und verglichen. Wir zeigen, dass Kombinationen aus ASR und Präprozessoren existieren, die die Wortfehlerrate (Word Error Rate, WER) verbessern, aber auch solche, die zu viele Artefakte erzeugen und die STT-Zuverlässigkeit beeinträchtigen. In einem Best-Case-Szenario konnte bei einem Audiokorpus mit hoher Varianz die WER um 7.1% reduziert werden. Dieses Szenario setzt jedoch voraus, dass der beste Audiopräprozessor für eine bestimmte Audioaufnahme bekannt ist. Daher beschäftigt sich der zweite Teil dieser Arbeit mit der Vorhersage des besten Präprozessors. Die Experimente mit Support-Vektor-Maschinen (SVMs) und neuronalen Netzen (NNs) zeigen, dass mit den Metriken von Dolby.io und librosa der beste Präprozessor für einen bestimmten Audioclip nicht vorhergesagt werden kann. Allerdings erzielten die WER-Approximation Experimente vielversprechende Ergebnis. So kann beispielsweise ein NN verwendet werden, um die WER vorherzusagen. Mit diesen Vorhersagen ist es dann möglich zu bestimmen, ob der Dolby.io-Enhancer auf einer Aufnahme angewendet werden sollte oder nicht, was zu einer WER-Reduzierung von 3.1% führte. Das beste Ergebnis wurde mit Sentence Scoring erzielt. Durch Wahrscheinlichkeitsberechnungen der Transkripte konnten die fehlerhaften Transkripte erkannt werden und führten zu einer WER-Reduktion von 7%, bei einem Testdatensatz, bei dem der beste Fall 9.8% betragen würde. Die Vorhersagemodelle basieren auf DeepSpeech. In einer Folgearbeit soll untersucht werden, ob die Ergebnisse auch mit anderen STTs erzielt werden können.

Abstract

Automatic speech recognition works reliably on clean audio recordings. However, background noise and reverberation in the audio significantly affect transcription quality. This bachelor thesis investigates speech enhancement techniques in the context of automatic speech recognition (ASR). The goal is to improve the accuracy of ASR systems in environments with background noise and reverberation. To achieve this goal, various speech-to-text (STT) engines and preprocessors for speech enhancement are quantitatively investigated and compared. We show that ASR and speech enhancement combinations exist that improve Word Error Rate (WER), but there are also those that produce too many artifacts and degrade STT reliability. In a best-case scenario, we were able to reduce the WER by 7.1% on a high variance audio corpus. However, this scenario assumes that the best audio preprocessor for a given audio clip is known. Thus, the second part of this thesis deals with the prediction of the best preprocessor. The experiments with support vector machines (SVMs) and neural networks (NNs) show that, using Dolby.io and librosa metrics, the best preprocessor for a given audio clip cannot be predicted. However, experiments using WER approximations have shown some promising results. For instance, an NN can be used to predict the WER and thus determine whether the Dolby.io enhancer should be applied to a recording, leading to a WER reduction of 3.1%. The best result was obtained with Sentence Scoring. Through probability calculations of the transcripts, the erroneous transcripts could be detected and led to a WER reduction of 7%, on a test data set where the best-case would be 9.8%. The prediction models are based on DeepSpeech. A follow-up work shall investigate whether the results can also be achieved with other STTs.

Vorwort

Diese Arbeit befasst sich mit der automatischen Spracherkennung und den Speech Enhancement Verfahren. Insbesondere fokussieren wir uns auf die Evaluation von Methoden zur robusten Spracherkennung in Umgebungen mit Störgeräuschen. Wir untersuchen den Einsatz verschiedener Präprozessoren zur Verbesserung der Leistung von automatischen Spracherkennungssystemen und der Vorhersage des besten Speech Enhancement Verfahrens. Die Arbeit hat uns von Anfang bis Ende Freude bereitet und uns neue Technologien und Methoden näher gebracht

Die Zusammenarbeit mit unseren Betreuern, Herrn Prof. Dr. Mark Cieliebak und Herrn Nicola Good, war freundlich und aufschlussreich. Die wertvollen Anregungen und Ratschläge im Laufe dieser Arbeit haben wir sehr zu schätzen gewusst, wofür wir uns an dieser Stelle ganz herzlich bedanken möchten.

Inhaltsverzeichnis

1 Einleitung

Im Zuge der Covid-Pandemie ist es zur Norm geworden an Sitzungen online teilzunehmen und die Gespräche aufzuzeichnen. Dies bringt die Vorteile der Digitalisierung mit sich: Sitzungen können im Nachhinein noch einmal angeschaut und Transkripte können automatisch generiert werden.
Die automatische Spracherkennung (Automatic Speech Recognition, ASR) findet in unserem täglichen Leben eine Vielzahl von Anwendungsfällen. Es können nicht nur Transkripte erzeugt werden, sondern sie ermöglichen auch den Umgang mit Computern und Smartphones ohne physische Interaktionen. Spätestens seit der Einführung von Apples Sprachassistentin Siri sind alle Verbraucher mit ASR in Berührung gekommen. Doch die Geschichte der ASR reicht weit zurück. Bereits 1952 versuchten drei Forscher des Bell Labs mit einem System namens Audrey", Ziffern anhand von Audioaufnahmen zu erkennen. Seitdem wurden viele verschiedene Ansätze entwickelt, um aufgezeichnete Sprache in Text umzuwandeln. In der Vergangenheit basierte ASR auf Methoden wie Gaussian Mixture Modells (GMMs), Hidden Markov Modells (HMMs) und Mel-Frequency Cepstral Coefficients (MFCCs). Das ASR-Forschungsfeld war lange Zeit träge und ASR war nur mit einem kleinen vorgegebenen Vokabular möglich. Kontinuierliche Sprache war ebenfalls eine Herausforderung, so dass die Spracherkennung diskrete Sprache mit deutlichen Pausen zwischen den Wörtern erforderte.
Mit zunehmender Rechenleistung, neuen Entwicklungen im Bereich Deep Learning und der Verfügbarkeit grosser Datenmengen wurde das Forschungsgebiet jedoch um 2010 wiederbelebt. Mit der heutigen Rechenleistung und den verfügbaren Datenmengen können tiefe neuronale Netze (DNNs) trainiert werden. Diese sind nicht nur zuverlässiger als herkömmliche Techniken, sondern ermöglichen auch eine kontinuierliche Spracherkennung mit einem umfangreichen Wortschatz. Fast alle grossen Technologieunternehmen bieten ihre eigenen Speech-to-Text (STT) Dienste an, sei es für den Endverbraucher und über Produkte wie Google Assistant oder als B2B-Service in der Cloud. Es existieren auch Open-Source-STT-Implementierungen von gemeinnützigen Organisationen, wie Mozilla mit DeepSpeech. Hier ist nicht nur die STT-Engine frei verfügbar, sondern auch vortrainierte Modelle für verschiedene Sprachen.
Nicht zu Unrecht wird heute oft gesagt, dass Daten das neue Öl sind. ASR-Systeme mit einer DNN-Architektur benötigen in der Trainingsphase eine Menge Audiodaten mit Referenztexten. Sogenannte ASR-Korpora sind im Internet zu finden. LibriSpeech ist eines der bekanntesten ASR-Korpora, welches unter der CC BY 4.0-Lizenz veröffentlicht wurde

und somit frei verwendet werden kann. Der LibriSpeech-ASR-Korpus enthält 1000 Stunden Audioaufnahmen aus Hörbüchern und Referenztranskripten. Je mehr Audiodaten man hat, desto besser, deshalb wurden auch neue Methoden entwickelt, um die Audiokorpora durch Manipulation der Audioaufnahmen zu erweitern. SpecAugment ist eine solches Verfahren, das sehr erfolgreich die Audiodaten erweitert und die Genauigkeit der ASR erhöht. Bei dieser Methode werden neue zusätzliche Audiodaten erzeugt, indem Teile der Sprachmerkmale in den Aufnahmen werden.
Um einen Eindruck davon zu vermitteln, wie zuverlässig ASR derzeit (Stand: Feb. 2022) funktioniert, ist in der Tabelle 1.1 ein Vergleich zwischen verschiedenen kommerziellen Diensten aufgeführt. Solche Vergleiche sind jedoch mit Vorsicht zu geniessen, da diese Werte nur für die verwendeten Korpora repräsentativ sind.

Engine	LibriSpeech test-clean	LibriSpeech test-other	TED-LIUM	Common Voice	Average
Azure Speech-to-Text	**4.96%**	9.66%	4.99%	**12.09%**	**7.93%**
Amazon Transcribe	5.20%	**9.58%**	**4.25%**	15.94%	8.74%
Picovoice Leopard	5.39%	12.45%	9.04%	17.13%	11.00%
Google Speech-to-Text (Enhanced)	6.62%	13.59%	6.68%	18.39%	11.32%
Picovoice Cheetah	7.08%	16.28%	10.89%	23.10%	14.34%
Mozilla DeepSpeech	7.27%	21.45%	18.90%	43.82%	22.86%
IBM Watson Speech-to-Text	11.08%	26.38%	11.89%	38.81%	22.04%
Google Speech-to-Text	11.23%	24.94%	15.00%	30.68%	20.46%

Tabelle 1.1.: Tabelle aus dem ASR Benchmark von Picovoice [1], die Ergebnisse lassen sich mit dem Code auf Github reproduzieren.

Das Testset LibriSpeech test-clean enthält nur saubere Aufnahmen, so dass mit Azure STT eine Word Error Rate (WER) von 4.96% erreicht werden kann. Weitere Informationen zur WER sind in Kapitel 2.3 zu finden.
Grundsätzlich besteht das Ziel eines ASR-Systems darin, das Erlernte möglichst zu verallgemeinern und auf Sprachaufnahmen in der Grundgesamtheit anzuwenden. Die für das Training verwendeten Aufnahmen werden als Trainingsset bezeichnet, während die Aufnahmen, die nur für die Inferenz/Auswertung zum Einsatz kommen, zum Testset gehören. ASR- und STT-Systeme arbeiten zuverlässig, wenn die Trainings- und Testdaten gut aufeinander abgestimmt sind und die Aufnahmequalität hoch ist [2]. Die Audioqualität kann hochgehalten werden, indem sichergestellt wird, dass der Sprecher gut zu hören ist und keine Hintergrundgeräusche die Aufnahme stören. In der Praxis erweist sich dies jedoch als schwierig, da die Umgebung nicht kontrolliert werden kann, unerwartete Ereignisse auftreten und es schwierig ist, die Audioaufnahme während der Aufzeichnung zu überprüfen. Geräusche und andere Umgebungsfaktoren verschlechtern die Sprachqualität, was sich wiederum auf die Qualität der Transkription auswirkt.
In dieser Bachelorarbeit werden verschiedene Speech Enhancement Verfahren evaluiert und deren Einfluss auf die WER untersucht. Darüber hinaus wird geprüft, ob es Audioeigenschaften gibt, die auf das beste Speech Enhancement Verfahren hinweisen, um die bestmögliche Transkriptionsgenauigkeit zu erreichen.

1.1 Ausgangslage

1.1.1 Performance Faktoren

In der Veröffentlichung von Errattahi et al. aus dem Jahr 2018 [3] wurden die negativen Einflüsse, für eine automatische Transkription, in drei Faktoren unterteilt. Die ersten beiden Faktoren betreffen die Sprecher:innen. Die Aussprache von Texten kann situationsabhängig sein, da sich zum Beispiel aufgrund von Emotionen die Äusserung ändern kann. Spontane und akzentuierte Sprache und der Grad an Aussprachevariation aufgrund von Dialekten oder Koartikulation führen zu weiterer Sprachvariabilitäten, welche die Diskrepanzen zwischen Trainings- und Testdaten vergrössern. Die dritte Kategorie umfasst, die in der Arbeit erwähnten Inkongruenz-Faktoren. Unterschiede zwischen den Trainings- und Testdaten können selbst dann auftreten, wenn die Äusserung die gleiche war. In dem Buch "Techniques For Noise Robustness In Automatic Speech Recognition"[4] wird beschrieben, dass zwei identische Mikrofone desselben Herstellers niemals denselben Frequenzgang haben werden. Auch der Übertragungskanal spielt eine Rolle. Analoge Kanäle, Bandbreitenbeschränkungen und unterschiedliche Digitalisierungsmethoden beeinflussen den Inhalt einer Audiodatei. Im Gegensatz zu traditionellen ASR-Systemen basierend auf GMMs können Systeme, die auf DNNs basieren, gut mit den erwähnten Inkongruenz-Faktoren umgehen. Eine viel grössere Herausforderung sind Störgeräusche und Nachhall im Audiomaterial [2]. Stationäre und nicht-stationäre Hintergrundgeräusche führen zu einer Diskrepanz zwischen der aufgenommenen Sprache und dem für die Erkennung verwendeten akustischen Modell. Diese Diskrepanz verschlechtert die Erkennungsleistung erheblich, insbesondere bei der Spracherkennung auf Distanz. Nun kann versucht werden, die Aufnahmen zu verbessern, indem beispielsweise Störgeräusche und Nachhall aus den Audiodaten entfernt werden. Diese Massnahmen werden in der Literatur als Front-End Preprocessing bezeichnet, da die Aufnahmen vor der Spracherkennung bearbeitet mit Speech Enhancement aufbereitet werden.

1.1.2 Speech Enhancement

Das Gebiet Speech Enhancement zielt darauf ab, die Sprachqualität mit Hilfe verschiedener Algorithmen zu verbessern, um die Verständlichkeit und/oder die allgemeine Wahrnehmungsqualität von Sprache in Audiodateien zu erhöhen [5]. Früher wurden Audiosignale mit Hilfe von Signalverarbeitungstechniken wie dem Wiener Filter oder Spectral Gating [6] verbessert. Mit dem Durchbruch von neuronalen Netzen und Deep Learning wurden jedoch neue Ansätze und Techniken entwickelt, um die Audioqualität durch das Entfernen von Hintergrundgeräuschen und Nachhall zu verbessern. Die Sprachqualität kann mit objektiven Metriken wie der Perceptual Evaluation of Speech Quality (PESQ) [7] oder der Perceptual Objective Listening Quality Analysis (POLQA) [8] gemessen werden. Zudem existieren normierte subjektive Methoden zur Bewertung der wahrnehmbaren Audioqualität. Methoden wie P.808 [9] oder Mean Opinion Score (MOS) [10]

Model Test	Noise-free Noise-free	Enhanced Enhanced	Noisy Noisy	Noise-free Enhanced	Noise-free Noisy
DNN (Ø Keyword Accuracy)	96.53	90.3	84.00	81.87	57.87

Tabelle 1.2.: Daten aus der Arbeit von Delcroix et al. [2]. Keyword-Genauigkeit auf der Trainingsmenge für die Modelle, die mit rauschfreier Sprache trainiert und mit rauschfreier, verrauschter und optimierter Sprache getestet wurden. Die Zeilen zeigen die durchschnittliche Leistung, die mit verschiedenen DNN-Modellen erzielt wurde.

können verwendet werden, um die subjektive Bewertung der Sprachqualität mit einem Crowdsourcing-Ansatz zu verfolgen. All diese Metriken werden verwendet, um neuronale Netze zu trainieren, die die Audioqualität für den Menschen hörbar verbessern [11], [12]. Unklar bleibt jedoch, ob die Bearbeitung des Audios, wie beim Menschen, die Verständlichkeit des Gesprächs erhöht und gleichzeitig die Word Error Rate (WER) der Transkription senken kann.

1.1.3 Stand der Technik

Audio Präprozessoren mit DNNs

Die Arbeit von Delcroix et al. [2] untersuchte Speech Enhancement im Zusammenhang mit automatischen Spracherkennungssystemen. Es wurden verschiedene Sprachmodelle erstellt und mit störungsfreiem, gestörtem und mit DOLPHIN aufgewertetem Audio getestet. Die Ergebnisse zeigten, dass die Konfiguration mit dem Modell, das mit ”noisy” Audio traniert und ”noisy” Audio evaluiert wurde, eine höhere Genauigkeit aufwies als das Modell mit ”noise-Free” Trainingsaudio und ”enhanced” Test Audio. Es wird vermutet, dass dies auf Verarbeitungsartefakte zurückzuführen ist, die zu Diskrepanzen in den Test- und Trainingsdaten führen. In Anbetracht der Tatsache, dass man mit gestörtem Audio arbeiten muss, schnitt das Modell mit den Enhanced Trainingsdaten und Enhanced Testaudio am besten ab, da die Inkongruenz-Faktoren bei diesem Setup möglicherweise geringgehalten werden können und gleichzeitig Störgeräusche entfernt werden können.

CHiME Challenge

Die DOLPHIN-Sprachoptimierungsmethode wurde ursprünglich für die CHiME Challenge entwickelt [13]. Der CHiME-Wettbewerb wird regelmässig veranstaltet, mit dem Ziel, die Erkennung mittels Sprachseparation zu verbessern und einen möglichst niedrige WER zu erreichen [14]. Die Auswertung der vierten Auflage des CHiME-Wettbewerbs zeigt, dass Sprachoptimierungsverfahren, die ein Multi-Array-Mikrofon-Setup verwenden, Sprache deutlich besser optimieren und damit niedrigere WERs erreichen [15]. Wie

bei DOLPHIN handelt es sich dabei um Algorithmen, die räumliche Merkmale aus den Audiosignalen mehrerer Mikrofone extrahieren und diese Merkmale nutzen, um Sprache von Umgebungsgeräuschen zu trennen. Bei Verfahren, bei denen nur ein Mikrofon verwendet wird, ist die Fehlerrate etwa doppelt so hoch [15]. Offen bleibt die Frage, wie sich STT-Engines wie Google oder Deep Speech im Zusammenspiel mit Präprozessoren verhalten.

1.2 Zielsetzung

Ziel dieser Bachelorarbeit ist es, den Einfluss von Algorithmen zur Speech Enhancement und Rauschunterdrückung nach der Aufnahme auf die WER zu untersuchen. Es soll geklärt werden, wie sich verschiedene Produkte aus dem Bereich Speech Enhancement auf unterschiedliche Speech-to-Text-Implementierungen auswirken und ob es bestimmte Implementierungskonfigurationen gibt, die sich positiv auf die Qualität der Transkripte auswirken. Zudem soll untersucht werden, ob es bestimmte Charakteristiken in Audiodateien gibt, die den Einsatz einer bestimmten Implementierung aus dem Bereich der Sprachanhebung nahelegen. Im Anhang ist die offizielle Aufgabenstellung der Bachelorarbeit zu entnehmen A.1.

2 Grundlagen

In diesem Kapitel werden die notwendigen Grundlagen für das Verständnis unserer Experimente erläutert. Gängige Tools, Verfahren und Metriken, die in allen Experimenten verwendet werden, sind hier erläutert. Einzelne Experiment-Kapitel können neue Grundlagen aufgreifen, da sie nur in diesem einen Fall behandelt wurden.

2.1 Präprozessoren

Um die Sprache in den Audiodateien zu optimieren, wurde eine breite Palette bestehender Speech Enhancement Tools verwendet. Module wie NVIDIA Maxine [16], PoCoNet [17] und Deep Xi [18] basieren auf Machine Learning-Modellen, während Noisereduce [19] auf einer Signalverarbeitungstechnik beruht. Darüber hinaus unterscheiden sich die Produkte auch in der Lizenzierung. NVIDIA Maxine und Dolby.io [20] sind kommerzielle Produkte, deren Implementierung nicht offengelegt wird. Die auf Github verfügbaren Projekte Noisereduce, PoCoNet und Deep Xi sind quelloffen und ihre Implementierung / Modelle wurde in Forschungsarbeiten veröffentlicht.

Präprozessor	Parameter	Für 2 min Audio
Noisereduce	- Stationary: [True | False]	∼1.2s
NVIDIA Maxine	- Effect: [dereverb_denoiser | dereverb | dereverb] - Intensity Ratio: [0.1, ..., 1.0]	∼55.0s
Dolby.io Enhance	Viele (Siehe Dokumentation)	∼25.6s
PoCoNet	Keine	∼27.7s
Deep Xi (resnet-1.1n)	Keine	∼66.3s (CPU)

Tabelle 2.1.: Die Präprozessoren können unterschiedlich konfiguriert werden. Dabei unterscheiden sie sich auch in der Bearbeitungszeit. Die letzte Spalte zeigt, die Bearbeitungszeit für ein Audio mit der Länge von zwei Minuten.

2.1.1 Noisereduce

Das Python-Modul Noisereduce ist ein Algorithmus zur Rauschunterdrückung in Signalen wie z.B. Sprache. Noisereduce basiert auf eine Methode namens Spectral-Gating, dabei wird ein Spektrogramm eines Signals berechnet und ein Rauschschwellenwert für jedes Frequenzband des Signals bestimmt [19]. Dieser Schwellenwert wird zur Berechnung einer Maske verwendet, die das Rauschen unterhalb des frequenzabhängigen Schwellenwerts herausfiltert. Bei der stationären Rauschunterdrückung wird der Schwellenwert für das gesamte Signal beibehalten. Es ist jedoch auch möglich, den Schwellenwert kontinuierlich zu aktualisieren, dies wird als nicht stationäre Rauschunterdrückung bezeichnet.

2.1.2 NVIDIA Maxine - Audio Effects

NVIDIA bietet mit Maxine ein SDK an für GPU-beschleunigtes Verarbeiten von Audio, Video und Augmented Reality. Der Fokus der Audioverarbeitung liegt bei diesem SDK darauf, die Sprachqualität in Audiodaten zu verbessern in dem Hintergrundgeräusche und Störfaktoren reduziert werden. Dies wird gemäss dem Hersteller durch den Einsatz eines Neuronalen Netzes erreicht, das auf entsprechenden Grafikkarten ausgeführt wird. Auf der Produktseite des SDKs [16] gibt es auch kurze Demonstrationen der Möglichkeiten des SDKs, das unter dem Namen RTX Voice und als Teil von NVIDIA Broadcast auch für Endverbraucher zur Verfügung steht. Die Art und Weise wie das Audio verarbeitet wird kann über verschiedene Parameter konfiguriert werden [21]. Dabei stehen verschiedene Effekte wie ”Denoising”, ”Dereverb” oder die Kombination der beiden zur Auswahl. Diese Stärke der Effekte kann zusätzlich über eine ”Intensity Ratio” von Null bis Eins gesteuert werden. Aus der Dokumentation des SDKs [22] ist zu entnehmen, dass es gegenwärtig Grafikkarten wie die NVIDIA Tesla T4 erfordert. Diese wurden von der Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) bereitgestellt.

2.1.3 Dolby.io Media Enhance API

Dolby.io bietet eine kostenpflichtige API zur Audioverbesserung in der Cloud an. Der Dienst verfügt über eine Reihe von Funktionen zum Entfernen von Störgeräuschen, aber auch zum Angleichen von Lautstärken, Reduzieren von Zischlauten, Reduzieren von Mundgeräuschen und mehr [20]. Durch Angabe des Inhaltstyps, z. B. mobile_phone, werden Voreinstellungen für die Audiooptimierung geladen. Die Dokumentation von Dolby.io beschreibt die verschiedenen Inhaltstypen und es wird auch erklärt, welche Optimierungen vorgenommen werden. Der Typ mobile_phone wird gemäss Dolby [23] für folgendes Szenario verwendet, “Der Typ mobile_phone ist nützlich für Medien, die von unterschiedlichen Mikrofonpositionen aus aufgenommen werden, da sich das Telefon während der Dauer des Inhalts mal näher und mal weiter vom Sprecher entfernt.” Mit diesem Inhaltstyp werden dabei folgende Optimierungen vorgenommen, “[...] die Rauschunterdrückung und Sprachisolierung auf Maximum eingestellt, da die Umgebung variiert und sowohl Innen- als auch Aussenbereiche umfasst, in denen die Medien

möglicherweise aufgenommen wurden. Da ein Mobiltelefon oft in einiger Entfernung vom Mund des Sprechers gehalten wird, ist die Plosivreduzierung auf niedrig eingestellt". Gemäss der API-Dokumentation[24], können einige dieser Parameter überschrieben werden, für diese Arbeit wurden jedoch keine Änderungen vorgenommen.

2.1.4 PoCoNet

Jährlich veranstaltet Microsoft auf der Interspeech-Konferenz eine Deep Noise Suppression Challenge. Bei diesem Wettbewerb werden Deep-Learning-Modelle zur Störgeräuschunterdrückung eingesetzt, um den MOS-Wert von Audios zu maximieren. Die Ergebnisse der einzelnen Veranstaltungen werden jeweils veröffentlicht. Im Jahr 2020 fand die letzte Challenge mit den Kategorien Non-Realtime statt. Der Gewinner der Non-Realtime-Kategorie war das Team von Amazon Web Services mit dem Convolutional Neural Network (CNN) PoCoNet [12]. Dieses CNN wurde auf verschiedenen Hörbuch- und Fernsehsendedatensätzen trainiert, mit dem Ziel, die menschliche Wahrnehmung der Audioqualität zu verbessern [17]. Eine vortrainiertes Modell mit ähnlicher Architektur ist über das OpenVINO-Projekt von Intel verfügbar [25].

2.1.5 Deep Xi

In der Arbeit von Nicolson et. al. [18] wurde Deep Xi insbesondere als ASR-Vorprozessor untersucht. Deep Xi ist der Name für einen Ansatz, der eine a priori SNR-Schätzung und ein tiefes neuronales Netzwerk verwendet, um nicht-stationäre Störsignale aus Audiosignalen zu entfernen. Hinweise über das Signal-Rausch-Verhältnis (Signal-to-Noise Ratio, SNR) sind im Kapitel 2.4 zu finden. Experimente von Nicolson et. al. zeigen, dass Deep Xi die WER von DeepSpeech erheblich reduzieren kann.

	SNR level (dB)																			
Präprozessor	**"Voice babble"**					**Strassen Musik**					**F16**					**Fabrik**				
	-5	**0**	**5**	**10**	**15**	**-5**	**0**	**5**	**10**	**15**	**-5**	**0**	**5**	**10**	**15**	**-5**	**0**	**5**	**10**	**15**
Noisy speech	95.1	91.1	70.6	36.7	11.0	92.9	78.2	51.1	21.1	11.0	100.0	98.7	82.5	39.2	18.1	97.7	89.0	59.8	29.2	10.5
DD	94.9	87.8	70.4	32.0	10.4	88.5	72.4	47.6	24.6	14.0	98.4	82.1	43.2	18.8	11.6	94.6	83.6	52.7	26.2	13.8
Clust. recon.	98.2	84.5	54.2	21.0	10.2	86.2	72.2	41.3	15.4	10.1	98.0	83.1	45.4	18.6	7.0	97.1	81.6	50.3	29.2	16.7
Xu2017.0	95.0	78.1	54.3	31.3	13.0	90.0	75.4	45.2	28.6	17.1	93.8	81.7	51.2	25.1	21.7	97.4	90.7	67.1	35.4	18.3
LSTM-IRM	94.1	88.9	54.2	22.5	**9.6**	89.6	67.5	39.8	16.2	7.2	97.1	79.3	45.2	21.4	12.9	94.5	73.8	43.2	18.1	10.9
SEGAN	95.8	79.0	44.2	19.5	10.7	84.5	58.7	32.4	12.8	11.0	94.2	76.8	45.6	19.7	7.7	91.2	70.3	45.9	18.1	**8.5**
Deep Xi-ResLSTM	**92.9**	73.3	37.1	11.7	12.0	85.5	58.1	25.2	11.5	**6.4**	95.6	69.3	30.8	16.4	7.2	93.0	69.5	36.2	17.9	9.1
Deep Xi-ResBiLSTM	95.0	**66.0**	**27.1**	**10.7**	10.4	**73.0**	**43.0**	**23.7**	**10.3**	7	**88.7**	**56.6**	**21.4**	**12.7**	**4.2**	**84.9**	**51.5**	**29.5**	**16.8**	8.8

Tabelle 2.2.: Aus der Arbeit von Nicolson et. al. [18], wobei die linke Hälfte die WER mit den Präprozessoren bei nicht-stationären Störgeräuschen zeigt und die rechte bei stationären Störgeräuschen. Die Testdaten mit dem Lärm wurden künstlich auf die LibriSpeech-Korpusaufnahmen erzeugt

In einem Vergleich mit anderen Frontend-Präprozessoren erzielte das Modell Deep Xi-ResNet (1.1n) die besten Werte für die objektive Wahrnehmung der Audioqualität [26]. In 16 von 20 Fällen erreichte Deep Xi-ResBiLSTM die niedrigste WER [18]. Für diese Bachelorarbeit wird aufgrund der neueren Weiterentwicklungen das Deep Xi-ResNet (1.1n) Modell verwendet.

2.2 Speech-to-Text

Es existieren viele verschiedene Tools und Dienste für die automatische Erstellung von Transkripten. Einige Unternehmen wie Speechmatics [27] oder Nuance [28] haben sich auf ASR spezialisiert. Aber auch die grossen Technologieunternehmen wie Amazon [29], Google [30], Microsoft [31] und Apple [32] bieten verschiedene Möglichkeiten an, Audioaufnahmen zu transkribieren. Open-Source-Projekte wie DeepSpeech von Mozilla [33], wav2letter++ von Facebook [34] oder TensorFlowASR von TensorSpeech [35] können On-Premise aufgesetzt und inhouse betrieben werden. Nach Gartner lassen sich, wie in Abbildung 2.1 zu sehen ist, die Firmen und ihre Produkte in drei Kategorien unterteilen.

Abbildung 2.1.: Aus dem Bericht von Gartner [36]: Viele Unternehmen bieten verschiedene Möglichkeiten an, Audioaufnahmen zu transkribieren. Die Firmen und ihre Produkte werden in drei Kategorien unterteilt

In dieser Bachelorarbeit wurden die Pay-per-Use-Lösungen Google Cloud STT und Azure On-Premise evaluiert. Für die Auswertung von grossen Datenmengen mit hoher Varianz wurden die Lösungen von Apple und Mozilla verwendet, da sie an keine Kosten gebunden sind. Laut Gartner [36] decken diese vier ASR-Dienste einen hohen Marktanteil ab und stellen die Backend-Lösung dar, die häufig für verbraucherorientierte Anwendungen verwendet werden, da ihr SSelf-ServeAnsatz die Zugänglichkeit für Entwickler:innen erleichtert.

2.2.1 Google Cloud Speech-to-Text

Google betreibt Automatic Speech Recognition Systeme, das sich gemäss der Produktseite [30] durch starke Konfigurierbarkeit und eine hohe Accuracy auszeichnen. So kann

beispielsweise zwischen verschiedenen Modellen wie ”phone_call” oder ”video” gewählt werden. Die Benutzung der API ist kostenpflichtig, kann aber kostenlos getestet werden. Ein wesentliches Merkmal der STT ist, dass Google ausdrücklich davon abrät, die Audiodaten aufzubereiten, bevor sie an die API gesendet werden. Laut dem Best Practices Guide [37] ist der Dienst in der Lage, mit Störfaktoren umzugehen.

2.2.2 Azure STT / OnPrem

Eine ebenfalls kostenpflichtige Implementation von Automatic Speech Recognition kann bei Azure bezogen werden. Ein besonderes Merkmal dieser Implementation ist, dass das Modell um zusätzliches Vokabular erweitert werden kann [31]. Im Rahmen dieser Arbeit wird eine On-Premise Installation dieses Produktes benutzt. Daher wird es im Rahmen der Arbeit auch als OnPrem STT bezeichnet.

2.2.3 Mozilla DeepSpeech

DeepSpeech ist eine Opensource Speech-to-Text Implementation, die oft auf Anwendungsspezifischen Audios trainiert wird. Alternativ stehen aber auch ”pretrained” Modelle zur Verfügung, um die Trainingsphase überspringen zu können. [38] [39]
Im Rahmen dieser Arbeit wurde DeepSpeech auf einem Server mit GPU Beschleunigung installiert und mit dem englischen Pretrained Modell genutzt. Dabei zeigte sich, dass in dieser Kombination nur Audios mit 16 kHz Abtastrate und mit dem Codec ”pcm_s16le” transkribiert werden können.

2.2.4 Apple SFSpeechRecognizer

Mit der Einführung von macOS 10.15 hat Apple ein neues Speech Framework eingeführt. Dieses stellt den Entwickler:innen verschiedene APIs zur Verfügung, um Sprache in Echtzeit und in Aufnahmen zu transkribieren. Die On-Device Spracherkennung ist für eine Reihe von Sprachen verfügbar, in einigen Fällen kann das Speech Framework jedoch auf Apples Server für die Spracherkennung zurückgreifen. Im Rahmen dieser Arbeit wurde das Kommandozeilenprogramm hear verwendet, um WAV-Audiodateien in Text zu konvertieren. hear wurde in Objective-C geschrieben und dient als Wrapper für die Apple Speech API [40].

2.3 Word Error Rate

Die Word Error Rate ist das Standard Qualitätsmass für Speech-to-Text Anwendungen [41]. Dabei werden zwei Texte untereinander verglichen und die Fehlerrate anhand der Formel

$$WER = \frac{S + D + I}{N} \tag{2.1}$$

berechnet, wobei S für die Substitutionen, D für Deletions, I für Insertions und N für die Gesamtlänge des Referenztextes steht. Bemerkenswert ist, dass $N = S + D + C$ gilt, wobei C für die Anzahl korrekter Worte steht. Dies lässt erkennen, dass die Insertions stärker ins Gewicht fallen als die Substitutionen und Deletions. Eine weitere Eigenschaft der WER ist, dass sie 1 übersteigen kann, wenn der Referenz Text kürzer ist als die Hypothese. In dem Buch “Spoken language processing: a guide to theory, algorithm, and system development” [42] wird darauf hingewiesen, dass aus statistischer Sicht mehr als 500 Sätze, gesprochen von 5 bis 10 verschiedenen Sprecher:innen, erforderlich sind, um eine aussagekräftige Fehlerquote eines automatischen Spracherkennungssystems zu ermitteln. Zudem wird empfohlen, neue Techniken nur dann zu übernehmen, wenn sie die Fehlerquote um mindestens 10 % senken.

2.3.1 Einfluss der Textlängen auf die WER

Eine weitere Eigenschaft der WER ist, dass sie die Länge der Texte berücksichtigt. Werden beispielsweise zwei Texte A und B mit je zehn Wörtern verglichen und drei Wörter stimmen nicht überein, so wird die $WER_{10} = 0.3$ sein. Dasselbe Ergebnis würde auch entstehen, wenn zwei Texte C und D mit je 100 Wörtern und 30 Abweichungen verglichen würden: $WER_{100} = 0.3$. Wenn nun der Text B verbessert würde, so dass er gegenüber Text A lediglich zwei Abweichungen enthält, würde dies in einer Verbesserung von $WER_{\Delta 10} = -0.1$ resultieren. Wenn jedoch Text D ebenfalls um genau ein Wort verbessert würde, würde dies in $WER_{\Delta 100} = -0.01$ resultieren. Wenn nun $WER_{\Delta 10}$ und $WER_{\Delta 100}$ herangezogen werden, um die beiden Verbesserungen zu beurteilen, würde der Mechanismus hinter $WER_{\Delta 10}$ besser beurteilt werden als der hinter $WER_{\Delta 100}$. Das kann unter Umständen jedoch nicht der Wirklichkeit entsprechen.

Für die Bewertung der Speech Enhancement Systeme wird daher empfholen, die WER für Texte ähnlicher Länge zu berechnen.

2.3.2 Baseline WER

In dieser Arbeit wird die Baseline als WER zwischen dem Referenztext und dem Transkript des unbearbeiteten Audios bezeichnet.

$$WER_{\text{Baseline}} = WER(R, T) \tag{2.2}$$

Wobei R der Referenztext ist und T ein Transkript einer unbearbeiteten Audioaufnahme.

Die Abweichung zwischen der Baseline und dem WER des bearbeiteten Audiotranskripts resultiert in einem Delta, das die Bewertung eines Präprozessors ermöglicht.

$$WER_{\Delta} = WER(R, T') - WER_{\text{Baseline}} \tag{2.3}$$

Wobei $P(A) = T'$ gilt und P für einen Präprozessor steht.

Ein negatives Delta deutet darauf hin, dass sich die Transkriptionsgenauigkeit verbessert hat, ein positives Delta zeigt an, dass mehr Fehler erzeugt wurden und zu einer Verschlechterung geführt hat.

2.3.3 Cross Transkript WER

Die Word Error Rate zwischen zwei Transkripten von zwei verschiedenen STT-Engines wurde berechnet, um den Schwierigkeitsgrad der Transkription einer Audiodatei zu bestimmen.

$$WER_{\text{Cross STT}} = WER(T_x, T_y) \tag{2.4}$$

Wobei T_x und T_y für zwei Transkripte von unterschiedlichen STT-Engines stammen.

Ist die Fehlerquote 0, zeigt dies, dass beide ASR-Systeme vollständig miteinander übereinstimmen. Ist die $WER_{\text{Cross STT}}$ hoch, gab es Schwierigkeiten für beide ASR-Systeme, weil die Transkription mit Unsicherheiten verbunden war.

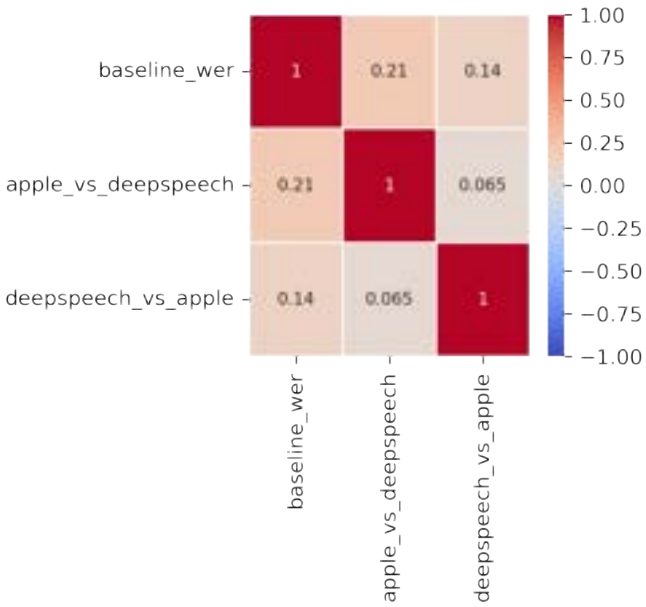

Abbildung 2.2.: Korrelationsmatrix der WER_{Baseline} und der $WER_{\text{Cross STT}}$ zwischen Apple und DeepSpeech

Wie im Abbild 2.2 zu sehen ist, wurde keine signifikante Korrelation zwischen der WER_{Baseline} und $WER_{\text{Cross STT}}$ gefunden. Dennoch erweist sich die $WER_{\text{Cross STT}}$ als ein guter Indikator dafür, ob sich die WER einer Audiodatei mit einem Präprozessor verbessern lässt, siehe 5.11 und 5.6.4.

2.3.4 WER API

Die Berechnung der Word Error Rate erfolgte über die Instanz der ZHAW, welche über HTTP erreicht werden konnte. Im Backend erfolgt die Berechnung über das Python-Modul jiwer [43].

2.4 Signal-to-Noise Ratio

Das Signal-Rausch-Verhältnis gibt Aufschluss darüber, wie viel Grundrauschen in einer Aufnahme vorhanden ist. Je höher der SNR-Wert ist, desto dominanter ist das Nutzsignal gegenüber dem Rauschen.

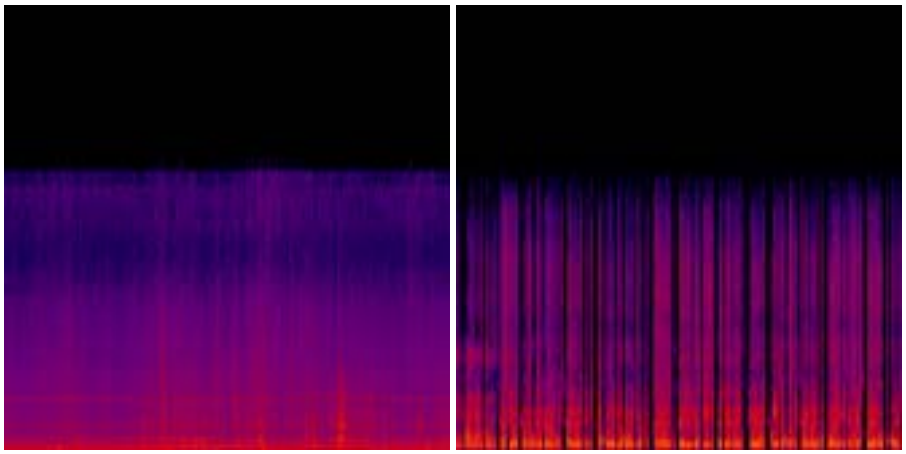

Abbildung 2.3.: Links: TtAM8h30.wav SNR von 22.95 dBFS, Rechts: TkbsQihk.wav SNR von 72.15 dBFS

Die Bilder in 2.3 zeigen zwei Audioaufnahmen mit unterschiedlichen SNR-Werten. TtAM8h30 hat einen niedrigen SNR-Wert und dementsprechend mehr Rauschen im Ton; dies ist in der Tonaufnahme zu hören, kann aber auch als Spektrogramm visualisiert werden. TkbsQihk hat einen hohen SNR-Wert; dies zeigt sich durch klare Konturen im Spektrogramm.
Das Verhältnis zwischen dem Signal und dem Rauschen ist auch unter "a priori SNR" bekannt. Nun kann man bei einer Audioaufnahme nur das Nutzsignal beobachten, welches mit dem Rauschsignal vermischt ist. Um den SNR-Wert zu bestimmen, müssen Nutzsignal und das Rauschsignal geschätzt werden. Hierfür gibt es verschiedene Methoden.

$$SNR = \frac{P_{\text{Signal}}}{P_{\text{Noise}}} \tag{2.5}$$

$$SNR = \frac{\mu}{\sigma} \tag{2.6}$$

Für viele Anwendungen reicht die Berechnung 2.6 aus, wobei μ der Erwartungswert und σ die Standardabweichung des Signals ist [44]. Fortgeschrittene Methoden versuchen, reine Rauschabschnitte z.B. mit Voice Activity Detection zu erkennen und daraus die Nutz- und Rauschsignalwerte zu extrahieren [45]. In dieser Arbeit erfolgt die SNR Berechnung über Dolby.io. Die Implementationsdetails werden von der Firma Dolby nicht angegeben. Die Berechnung des Korrelationskoeffizienten zwischen dem SNR und dem WER ergab einen r von 0.04, was darauf hinweist, dass das SNR keinen direkten Einfluss auf die WER hat.

2.5 MEL Frequency Cepstrum Coefficients

Im Bereich der natürlichen Sprachverarbeitung wird oft MEL-Frequency Cepstrum Coefficients (MFCC) eingesetzt. Beispielsweise wird in "Automatic speech recognition and speech variability: A review" [46] von M. Benzeghiba et. all das Berechnen der MFCCs als ein Standardverfahren für Feature Extraction bei ASR bezeichnet. In "Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition" von Md. Sahidullah und Goutam Saha werden MFCCs als das gängigste Verfahren zur Feature Extraction für Speaker Recognition genannt [47]. Wie der Name vermuten lässt wird dabei unter anderem die Mel-Skala genutzt, um die Audiofrequenzen auf eine Art zu transformieren, die nahe an der Funktionsweise des Menschlichen Gehörs angelegt ist [48].

2.6 Audiokorpora

Um im späteren Verlauf der Arbeit verlässliche Aussagen treffen zu können, musste schon früh ein grosser Audiodatensatz beschafft werden. Nachfolgend wird auf die verschiedenen Korpora eingegangen, die für diese Arbeit herangezogen wurden. Die wichtigsten Eigenschaften der Daten sind, dass sie für Speech-to-Text nachteilige Eigenschaften enthalten müssen, wie beispielsweise Hintergrundgeräusche, und dass das Gesprochene aus der Aufnahme auch als Text vorliegt.

2.6.1 SRF Video Archiv

Das Schweizer Radio und Fernsehen (SRF) stellt einen grossen Teil seines Videoarchives der Öffentlichkeit im Internet bereit [49]. Es ist möglich einzelne Aufnahmen mit einem einfachen Skript herunterzuladen. Dieses Archiv erschien als vielversprechend, da Live-Aufzeichnungen und Spielfilme mit gut hörbaren Störgeräuschen behaftet sind. Erste Versuche mit Aufnahmen von SRF zeigten jedoch, dass die Störgeräusche einen sehr kleinen Einfluss auf die Transkripte haben, da die in Standarddeutsch gesprochenen Teile trotz Störgeräuschen fast fehlerfrei transkribiert werden konnten. Dies legt die Vermutung nahe, dass die Tontechniker von SRF die Audios dahingehend optimiert haben, dass sie leicht zu verstehen sind.

Auch stellte sich heraus, dass offenbar sehr viele Abschnitte in Schweizer Dialekt gesprochen wurden. Diese Abschnitte lassen sich nicht transkribieren, da keine der Speech-to-Text APIs mit Schweizer Dialekt trainiert wurde.
Das Aufbauen dieses Korpus wurde daher beendet als er folgende Eigenschaften aufwies:

Gesamtaudiolänge	19:47:24.44 h
Durchschnittliche Audiolänge	00:45:40.59 h
Minimale Audiolänge	00:04:47.79 h
Maximale Audiolänge	01:36:42.71 h
Anzahl Wörter	116'425
Durchschnittliche Anzahl Wörter pro Audio	4477.88
Anzahl Audiodateien	26
Speicherbedarf	4.3 GB

Tabelle 2.3.: Eigenschaften des SRF Korpus

2.6.2 Deco Corpus

Dieser Korpus wurde von der ZHAW zusammengestellt. Es handelt sich dabei um eine Sammlung von Sprachaufnahmen, die in die Kategorien "Clean", "Average" und "Challenge" eingeteilt sind. Eine unvorteilhafte Eigenschaft dieses Korpus ist jedoch, dass die Referenztranskripte nicht in Standarddeutsch vorliegen, sondern buchstabengetreue Abschriften des Gesprochenen sind. Für den häufigen Fall, dass sprechende Person anstelle von 'ist' nur 'is' aussprach, fand sich das auch so im Referenztext. Wie sich zeigte, erkannten die Speech-to-Text APIs das gesprochene dennoch korrekt und gaben anstelle des abgekürzten Wortes das entsprechende vollständige Wort zurück. Dies erhöhte die Fehlerrate der Transkripte. Um diese Fehler zu reduzieren und die Aufnahmen zu nutzen, mussten die Referenztexte manuell überarbeitet werden. Da der Aufwand sehr gross wurde, konnte nur eine kleine Anzahl von Aufnahmen aus dem Deco Corpus verwendet werden. Dieser Teil weist folgende Eigenschaften auf:

Gesamtaudiolänge	00:30:52:77 h
Durchschnittliche Audiolänge	00:07:56.15 h
Minimale Audiolänge	00:00:35.33 h
Maximale Audiolänge	00:17:19.60 h
Anzahl Wörter	6008
Durchschnittliche Anzahl Wörter pro Audio	1502
Anzahl Audiodateien	4
Speicherbedarf	0.18 GB

Tabelle 2.4.: Eigenschaften des Deco Korpus

2.6.3 YouTube

Die zu Google gehörende Videoplattform YouTube bietet der Öffentlichkeit zahlreiche und unterschiedlichste Videos an, deren Audioqualität äusserst stark variiert. Viele Videos haben Untertitel, die vom engagierten Publikum des Videos geschrieben wurden. Wie erste Versuche zeigten, war die Übertragungsgeschwindigkeit beim Herunterladen der Daten sehr niedrig. Das hatte zur Folge, dass das Herunterladen um ein Vielfaches länger dauerte als die Wiedergabe des Audios. Es wurde bewusst darauf verzichtet die Ursache für diesen Effekt näher zu untersuchen, da sich auch das Extrahieren der Untertitel als umständlich erwies. Zudem liegt die Vermutung nahe, dass das ASR System von Google mit Daten von YouTube trainiert wurde, was die Resultate nachteilig beeinflussen könnte. Daher wurde der Aufbau dieses Korpus nach dem Download einer Datei aufgegeben.

Gesamtaudiolänge	00:05:06:73 h
Durchschnittliche Audiolänge	00:05:06:73 h
Minimale Audiolänge	00:05:06:73 h
Maximale Audiolänge	00:05:06:73 h
Anzahl Wörter	541
Durchschnittliche Anzahl Wörter pro Audio	541
Anzahl Audiodateien	1
Speicherbedarf	0.07 GB

Tabelle 2.5.: Eigenschaften des YouTube Korpus

2.6.4 Rev

In vielen Arbeiten werden die Störgeräusche künstlich auf die Audiodateien drauf gerechnet, realitätsnahe Testdaten und Experimente sind jedoch sehr wichtig. Deshalb wurden reale Aufnahmen mit Störgeräuschen und Umgebungsgeräuschen für die Experimente verwendet. Die Firma Rev.com ist im Bereich Speech-to-Text tätig und bietet auf ihrer Website rund 4'500 (Stand: Mai 2022) Video-/Audio-Transkripte an [50].

> bernie sanders: I've seen Russia and other countries interfering in our elections. The intelligence community has been very clear about it. Whether Trump recognizes that or not or acknowledges or not, they did interfere in 2016. The intelligence community is telling us they are interfering in this campaign right now in 2020, and when I say to Mr. Putin if elected president, trust me, you are not going to be interfering in American elections.
> speaker 2: Senator, when were you briefed on this?
> bernie sanders: ...

Abbildung 2.4.: Auszug aus einem Transkript aus dem Rev-Korpus. Politiker interview, während im Hintergrund Lärm von einem Flugzug zuhören ist.

Die Transkripte fallen u.a. in die Kategorien "Political Transcripts", "Press Briefing", Ïnterview Transcriptsünd zeichnen sich durch ihre hohe Variabilität aus. Die Audioaufnahmen im Korpus stammen aus verschiedenen Quellen und enthalten (nicht-) stationäre Hintergrundgeräusche, Nachhall und entfernte Sprache. Aufgrund der Aktualität ihres Inhalts kann sichergestellt werden, dass dieses Korpus nicht als Trainingsdaten für die Präprozessoren oder ASR-Systeme verwendet wurde.
Dieser Korpus weist folgende Eigenschaften auf:

Gesamtaudiolänge	382:25:41:67 h
Durchschnittliche Audiolänge	00:35:00:12 h
Minimale Audiolänge	00:00:55.43 h
Maximale Audiolänge	02:57:24.17 h
Anzahl Wörter	3'370'638
Durchschnittliche Anzahl Wörter pro Audio	5233.91
Anzahl Audiodateien	644
Speicherbedarf	124 GB

Tabelle 2.6.: Eigenschaften des Rev Korpus

2.6.5 Ceasr

Der Ceasr Korpus ist eine Zusammenstellung von verschiedenen Korpora, die oft in der Literatur im Zusammenhang mit ASR erwähnt werden. Die Besonderheit dieses Korpus ist, dass eine sehr breite Auswahl an verschiedensten Sprechstilen und Audioqualitäten abgedeckt wird. Damit lässt sich die Performance eines ASR Systems sehr gut evaluieren und auch erkennen, ob das System beispielsweise mit spontaner Sprache ähnlich gut umgehen kann wie mit vorgelesenen Texten [51].

3 Vorgehen

Nachdem im vorherigen Kapitel viele verschiedene Systeme für Speech Enhancement, Automatic Speech Recognition und Audio Korpora vorgestellt wurden, wird in diesem Kapitel beschrieben, wie diese als Bausteine genutzt werden. Schon in der Konzeptionsphase entstand die Idee, die verschiedenen Systeme als Komponenten einer Pipeline zu verwenden. Am Anfang der Pipeline sollen die Audios aufbereitet werden, dann durch Speech Enhancement laufen, transkribiert werden und schliesslich die Fehlerrate der Transkripte gegenüber dem Referenztext berechnet werden. Die Pipeline diente schliesslich als Leitfaden für diese Arbeit.

3.1 Pipeline

Die Pipeline wurde, wie in Abbildung 3.1 ersichtlich, in die vier Segmente Aufbereitung, Speech Enhancement, Automatic Speech Recognition und Evaluation gegliedert. Die Segmente wiederum bestehen aus einem oder mehreren Schritten, welche bestimmte der zuvor beschriebenen Systeme zum Einsatz bringt. So werden beispielsweise in Schritt 2.1 alle Speech Enhancement Systeme aufgerufen.

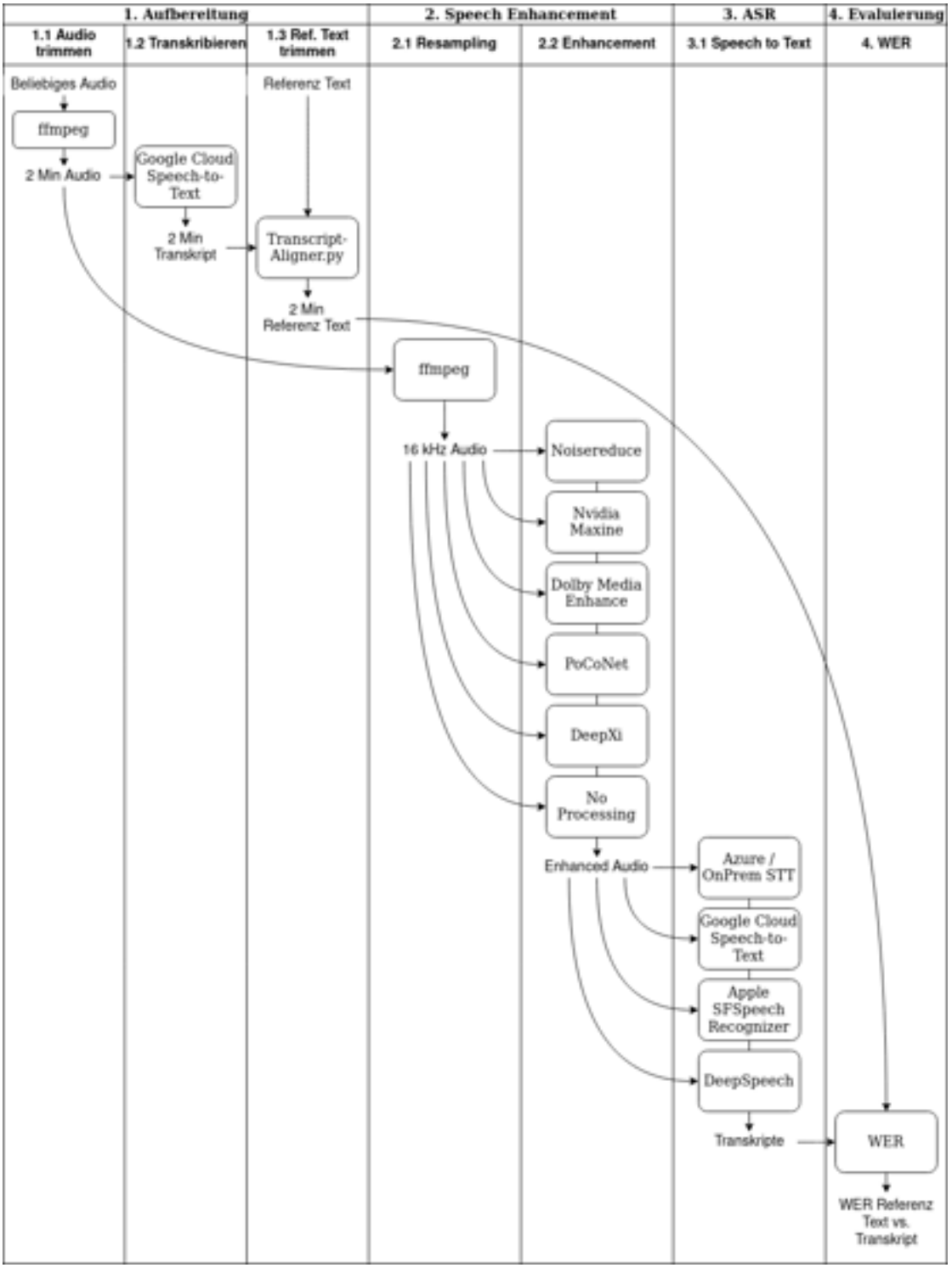

Abbildung 3.1.: Visualisierung der Schritte zur Generierung der Daten, für die Auswertung der Präprozessoren.

Auf die Aufbereitung folgte in den Schritten 2.1 und 2.2 das Speech Enhancement mit den Präprozessoren, bevor in Schritt 3.1 alle resultierenden Audiodateien mit den verschiedenen Speech-to-Text-Systemen transkribiert wurden. Zuletzt wurden in Schritt 4.1 die Transkripte ausgewertet und für Experimente verwendet.

3.1.1 Aufbereitung

Wie bereits in Abschnitt 2.3.1 beschrieben wurde, sollten die später entstehenden Transkripte einen ähnliche Anzahl Wörter aufweisen. Um dies zu erreichen, werden hier die Audiodateien aus dem Rev Korpus auf eine einheitliche Länge von zwei Minuten gekürzt. Diese Länge wurde gewählt, da die meisten der Dateien aus diesem Korpus diese Länge überschreiten und dennoch ausreichend Zeit bleibt für das Auftreten verschiedener Gesprächsteilnehmer und potenzieller akustischen Störungen. Der Schnitt des Audios konnte mit ffmpeg einfach automatisiert werden, aber die so neu entstandenen Audiodateien deckten nur noch einen Teil des originalen Referenztextes ab. Daher musste auch dieser geschnitten werden.

```
for i in *.wav; do
    ffmpeg -i "$i" -t 120 -c copy "../audio_2m/$i";
done
```

ffmpeg Aufruf zum Schneiden der Audiodateien

Das Kürzen und Abgleichen erfolgte mit dem Python-Skript `TranscriptAligner.py`, in dem das Originaltranskript mit einem Transkript der gekürzten Audiodatei abgeglichen wurde. In einem ersten Schritt wurden beide Texte bereinigt, indem die Buchstaben kleingeschrieben und Sonderzeichen entfernt wurden. Dies erleichterte den Vergleich der beiden Transkripte mit dem Tool wdiff. wdiff verwendet eine Methode, die der Levenshtein-Distanz ähnelt, indem es die kleinste Anzahl von Löschungen und Einfügungen ermittelt, um eine Textdatei aus einer anderen zu konstruieren [52].

i am donita williams i m [-an-] {+a+} optical fiber maker at corning in wilmington north carolina [-i-] {+our+} 326 members there and the [-still-] {+steel+} workers throughout our great union supply america [-do you-] {+we+} have the materials and components needed in our [-nations-] {+nation s+} infrastructure investments [-and-] {+in+} infrastructure are critical to workers like me my brothers and sisters in other [-in other-] industries and i was so proud to [-jay down-] {+join them+} to help pass this bill you [-were-] {+have+} signed today president biden one of the most [-exciting-] {+excited+} part about [-is build a-] {+this bill the+} 65 billion upgrade to expand broadband in communities across the country communities like mine [-and-] {+in+} rural north carolina this is not just [-the-] {+an+} investment in broadband [-buddy-] {+but+} in jobs [-corny-] {+at corning+} where i make fiber and investments in [-a-] fellow north [-carolina s star will help everyone have access to that-] {+carolinians+} this bill will help everyone have access to the internet that they need to teach their children run their businesses and sell products from their farms [-the stars-] {+these jobs+} are important [-where can i coin and make an-] {+working at corning making+} optical fiber i ve been there 26 years [-you help-] {+it helped+} me raise my daughter who [-is-] {+s+} now 30 as a single mom i was able to have healthcare [-provider-] {+provided+} able to send her to school to get [-an-] {+a+} education and it has [-allow-] {+allowed+} me to contribute to my local [-autonomy-] {+economy and community and it also helped me contribute to the tax [...]+}

Abbildung 3.2.: Output von `wdiff audioclip-2min-transcript.txt full-refrence-transcript.txt`, Blau zeigt Wörter die im Referenztext vorhanden sind aber im Audiocliptranskript fehlen, Rot entsprechend das Gegenteil und in Grün die letzte eindeutige Übereinstimmung.

Dieser Algorithmus erzeugt dann eine Ausgabe, welche die Unterschiede zwischen den beiden Texten anzeigt. Anhand dieser Ausgabe wird die letzte eindeutige Übereinstimmung im Text ermittelt und als Ankerpunkt definiert. Die Anzahl der Wörter, die dem Ankerpunkt im Audiotranskript folgen, werden nach dem Ankerpunkt im ursprünglichen Transkript hinzugefügt, und der Text wird dann an diesem Punkt abgeschnitten.
Wie Stichproben zeigten, funktionierte dieses Verfahren weitgehend gut. Wie sich jedoch später zeigte, gab es einige Texte, bei denen falsch geschnitten wurde. Diese konnten anhand ihrer hohen WER leicht identifiziert werden und wurden anschliessend manuell korrigiert.

3.1.2 Speech Enhancement

Die in Kapitel 2.1 erwähnten Präprozessoren wurden in unserer Arbeit integriert und evaluiert. Die Schnittstellen der Präprozessoren, mit Ausnahme von Dolby.io Enhance, wurden in einer Docker-Umgebung homogenisiert. Jeder Präprozessor wurde containerisiert und kann über HTTP angesprochen werden. In Abbildung 3.3 ist die Docker-Umgebung dargestellt. Das Load Balancing ermöglicht es, die Anfragen an die Präprozessor-Instanzen zu verteilen.

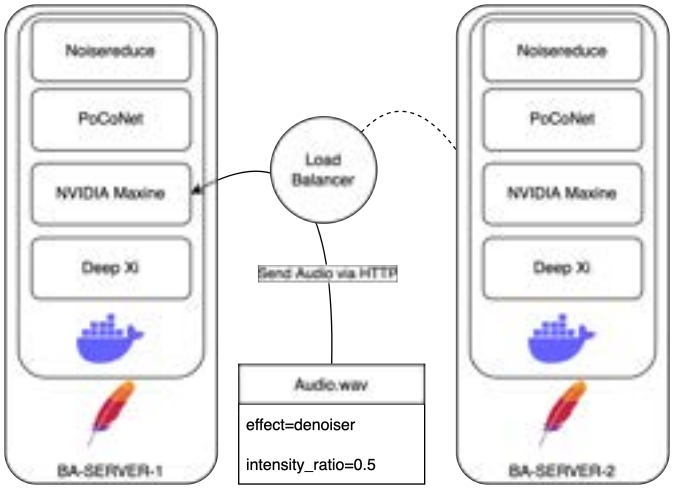

Abbildung 3.3.: Aufbau der Audio Processing Microservices

Die Optionen für das Speech Enhancement werden als HTTP Query Parameter übergeben. Präprozessoren, die GPU-Hardwarebeschleunigung unterstützen, wurden entsprechend konfiguriert.

3.1.3 ASR / STT

Um die Audiodateien zu transkribieren, wurden verschiedene Python-Skripte erstellt. Die Azure-OnPrem- und Google Cloud Speech-to-Text-Dienste wurden über ihre APIs gestartet. Die SFSpeechRecognizer-Transkripte von Apple wurden lokal auf einem MacBook gestartet. Für die Mozilla-DeepSpeech-Transkription wurde eine Instanz mit dem vortrainierten Standardmodell v0.9.3 aufgesetzt. Wo immer möglich, wurde die Sprache entsprechend den Audioinhalten konfiguriert.

3.1.4 Evaluierung

Nachdem die Audiodaten transkribiert waren, wurden die verschiedenen Outputs ausgewertet. Mit Hilfe der ZHAW-API wurden die WERs für die bereinigten Transkripte berechnet. Die WERs von zwei STT-Transkripten wurden ebenfalls berechnet, siehe **??**. Mit

Hilfe des Dolby.io Analyze-Dienstes wurden Metadaten generiert, um verschiedene Informationen und Merkmale über die Aufnahmen zu erhalten, z.B. speech_percentage oder snr_average.

3.2 Korpus Repository

Die generierten Daten wurden im Daten-Repository gespeichert. Dieser beinhaltet die Ordner "audio" für die originalen und bearbeiteten WAV-Dateien, "json" mit einer JSON-Datei, welche die WER-Werte enthält, "metadata" für Metadaten und "txt" mit den generierten Transkripten als Textdateien.

```
.
|-- audio
|   |-- 7btszJ36
|   |   |-- 7btszJ36.wav
|   |   |-- 7btszJ36_EN_16000___.wav
|   |   |-- 7btszJ36_EN_16000_deepxi__.wav
|   |   |-- 7btszJ36_EN_16000_dolby_mobile-phone_.wav
|   |   |-- ...
|   |-- ...
|-- json
|   |-- 7btszJ36
|   |   `-- scores.json
|   |-- ...
|-- metadata
|   |-- 7btszJ36
|   |   |-- 7btszJ36.json
|   |   |-- 7btszJ36_EN_16000___.json
|   |   |-- 7btszJ36_EN_16000___apple.json
|   |   |-- 7btszJ36_EN_16000___deepspeech.json
|   |   |-- ...
|   |-- ...
|-- txt
    `-- 7btszJ36
        |-- 7btszJ36_EN_16000___apple.txt
        |-- 7btszJ36_EN_16000___deepspeech.txt
        |-- 7btszJ36_EN_16000___google.txt
        |-- 7btszJ36_EN_16000___googleVideo.txt
        |-- 7btszJ36_EN_16000___onPrem.txt
        |-- 7btszJ36_EN_16000_deepxi___apple.txt
        |-- ...
```

Abbildung 3.4.: `tree` Auszug aus dem Korpus Repository Die Dateien folgen dem folgenden Namenskonzept: `<audio-name>_<sprache>_<sampling-rate>_<preprocessor>_<preprocessor-parameter>_<speech-to-text-engine>.<file-type>`.

3.3 Datenbereinigung

In einem Bereinigungsschritt wurden auffällige Audiodateien aussortiert. Aufnahmen mit einem geringeren Sprachanteil von 10 % wurden von den Experimenten ausgeschlossen. Dateien mit einer höheren WER von 80% wurden von Hand untersucht. Es wurde festgestellt, dass einige Audiodateien Fremdsprachen oder zu viele sich überlagernde Redebeiträge von mehreren Personen enthielten, diese wurden ebenfalls ausgeschlossen.

Eigenschaften	**Durchschnittliche Werte**
speech_percentage	92.509244
speech_num_sections	2.044118
…	…
silence_percentage	1.48229
silence_num_sections	0.334034

Tabelle 3.1.: Einige Werte zu unserem Korpus mit 463 Audioaufnahmen nach der Bereinigung

Nach der Bereinigung standen noch 463 Audioclips für die Bearbeitung zur Verfügung. Tabelle 3.1 zeigt die Durchschnittswerte für alle Audioclips. Die zweiminütigen Audiodateien enthalten im Durchschnitt 92.5% Sprache und 2.04 Sprachsegmente.
Transkripte und Referenztexte wurden durch die folgenden Schritte vereinheitlicht:

- Satzzeichen wurden in Python via `string.punctuation` durch Leerzeichen ersetzt, dies beinhaltet:

  `!"#$%&'()*+, -./:;<=>?@[\]^_‘{|}~`
- Zeilenumbrüche wurden durch Leerzeichen ersetzt
- Mehrere nachfolgende Leerzeichen wurden auf ein Leerzeichen reduziert
- Alle Buchstaben wurden in Kleinbuchstaben umgewandelt
- Anmerkungen in den Rev-Transkriptionen wie z.B. ”[Spanish]” wurden entfernt

4 Implementierung

4.1 Ausgangslage

Das Gesamtsystem setzt sich aus den in vorherigen Kapiteln beschriebenen Komponenten zusammen, die an unterschiedlichsten Orten betrieben werden bzw. installiert werden müssen und teilweise auch sehr spezifische Hardwareanforderungen haben. Es zeichnete sich daher ab, dass ein stark verteiltes System entstehen wird mit Verbindungen von ZHAW internen Servern zu verschiedenen Cloud Providern. Eine besondere Anforderung an das System ist, dass es möglich wird mit grossen Datenmengen zu arbeiten, da die Korpora einen erheblichen Umfang aufweisen.

4.2 Architektur

Im Verlauf der Arbeit wurden verschiedene Komponenten hinzugefügt, um weitere Erkenntnisse zu gewinnen. Daher wurde die Architektur des Systems an die neuen Bedürfnisse angepasst und entwickelte sich somit zum Zustand der in Abbildung 4.1 zu sehen ist. Auf Grund der gegebenen Verteilung der Komponenten, wurde beschlossen die Kommunikation der Komponenten über REST APIs abzuwickeln, mit Ausnahme der Datenablage. Diese wurde in zwei Kategorien aufgeteilt, eine für kleine Daten wie Transkripte, Audiofeatures und berechnete Word Error Rates und eine für die grossen Audiodateien. Die kleinen Dateien wurden einfachheitshalber in einem Git-Repository abgelegt und die grossen Dateien mit rsync auf den Audio File Server hoch- bzw. heruntergeladen.

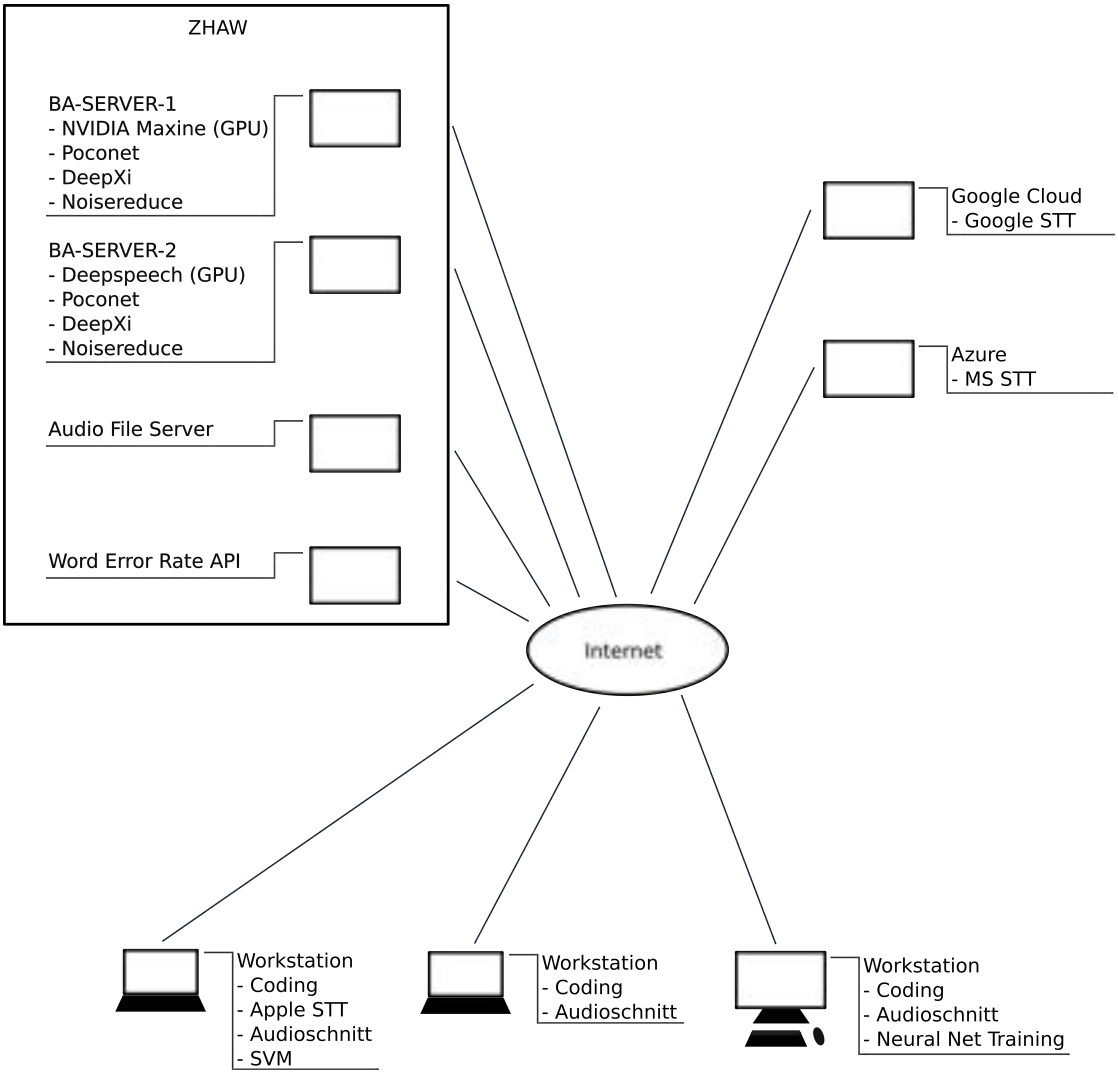

Abbildung 4.1.: Architekturübersicht

4.3 Herausforderungen

Der Speicherverbrauch der grossen Datenmenge wurde etwas unterschätzt. So mussten bei den Workstations weitere Speichermedien wie eine USB-HDDs bzw. ein NAS hinzugezogen werden. Zudem stelle sich die Synchronisation der Daten zwischen den Workstations als zeitintensiv heraus. Die Workstations befanden sich oft im Home-Office, was zur Folge hatte, dass der Up- und Download der Daten auf den Fileserver massgeblich von den Geschwindigkeiten der jeweiligen Internetverbindung eingeschränkt wurde. Auch zeigte sich, dass das Anwenden eines Speech Enhancers oder von Automatic Speech Recognition oft etwa gleich viel Zeit in Anspruch nahm, wie das Abspielen des betreffenden Audios. Dies resultierte stellenweise in Laufzeiten von

mehreren Stunden und einmalig von ca. sechs Tagen.

Im späteren Verlauf der Arbeit wurde ein Stück Code hinsichtlich besserer Lesbarkeit und kürzerer Laufzeit optimiert, damit der Zeitbedarf der Weiterentwicklung verkürzt werden konnte. Als das betreffende Stück Code schliesslich auf grösseren Datenmengen angewendet wurde, stieg jedoch der Memorybedarf dementsprechend an, was schlussendlich zu einer einschneidenden Hitzeentwicklung an der SSD führte, die den SWAP bereitstellte.

Für allfällige Folgearbeiten wird daher empfohlen die oben genannten Punkte frühzeitig zu bedenken.

5 Experimente

Ziel dieses Kapitels ist es, die Einflüsse der Präprozessoren auf die Audiodaten aufzuzeigen und einen Überblick über die Möglichkeiten und Grenzen des Speech Enhancement im Kontext der automatischen Spracherkennung zu geben. Auf Basis der Ergebnisse wurden erst Statistische Analysen und später mehrere Machine Learning Experimente durchgeführt, um das beste Preprocessing ohne Referenztext zu vorhersagen und die Word Error Rate zu minimieren. Dabei wurde immer ausschliesslich mit zweiminütigen Aufnahmen gearbeitet, ausser dort wo explizit auf die Dauer der Aufnahmen eingegangen wird.

5.1 Auswertung pro Audiodatei

5.1.1 Ziel

Da anfangs nur bekannt war, dass die Präprozessoren die gesprochene Sprache für das menschliche Ohr leichter verständlich machen, soll nun ein Überblick über die verschiedenen Kombinationen von Präprozessor und Speech to Text Implementation erstellt werden und die erreichten WER Werte visualisiert werden.

5.1.2 Aufbau

Es wurde ein kleiner Teil der Audiodateien aus dem Rev Korpus verwendet und die verschiedenen Präprozessoren angewendet. Die resultierenden Audiodateien wurden von verschiedenen Speech-to-Text Systemen transkribiert und die Transkripte wiederum wurden benutzt um die Fehlerraten zu berechnen. Die Fehlerraten wurden schliesslich abgespeichert und visuell aufbereitet.

5.1.3 Ergebnisse

In Abbildung 5.1 ist ein Teil der berechneten WER Werte aufgeführt für eine einzige Audiodatei. Es wurde bewusst darauf verzichtet alle Werte aufzuführen, da die Platzverhältnisse dafür zu beschränkt sind. Die Audiodatei wurde ausgewählt, da sie die wesentlichen Merkmale der meisten anderen Dateien ebenfalls wiedergibt und keine Ausreisser enthält.

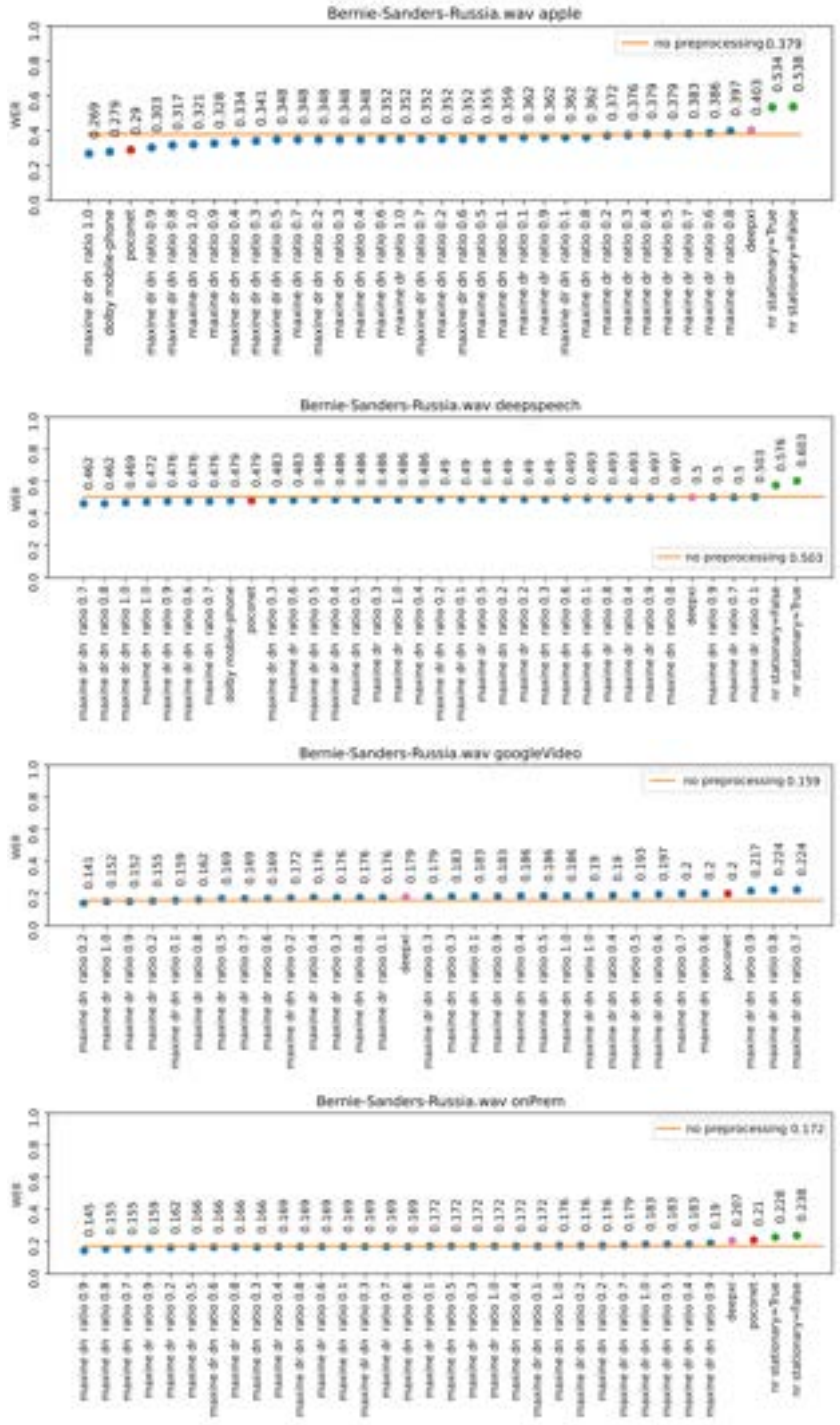

Abbildung 5.1.: Beispiel der Veränderung der WER durch Preprocessing

5.1.4 Diskussion

Wie in Abbildung 5.1 ersichtlich ist, ist es grundsätzlich möglich, die Fehlerrate in Transkripten durch das Anwenden eines Präprozessors zu senken. Es ist jedoch nicht offensichtlich welcher verwendet werden muss. Als Beispiel dafür kann der Präprozessor "PoCoNet" betrachtet werden. Dieser wurde für eine bessere Erkennbarkeit rot geplottet. Beim Speech-to-Text System von Apple konnte mit PoCoNet eine deutliche Verbesserung erreicht werden, aber beim System von Google beispielsweise wurde die Fehlerrate erhöht. Dieser Effekt zeigt sich auch von Audio zu Audio. Bei einer Datei kann PoCoNet eine deutliche Verbesserung bringen, jedoch bei einer anderen eine deutliche Verschlechterung, auch wenn dasselbe Speech-to-Text System verwendet wurde.
Auch wurde deutlich, dass der jeweils grün geplottete Noisereduce ausschliesslich starke Verschlechterungen mit sich brachte. Daher wurde dieser Präprozessor noch während dieses Experiments aufgegeben und ist daher nicht in allen Plots ersichtlich.

5.2 Auswertung pro Präprozessor

5.2.1 Ziel

Mit den Daten des vorherigen Experiments soll nun eine Auswertung pro Prozessor erstellt werden. Insbesondere soll geklärt werden, wie stark die Fehlerrate durch den Einsatz eines Präprozessors verändert wird. Auch soll geklärt werden, ob das Anwenden eines Präprozessors mehrheitlich eine Verbesserung oder Verschlechterung der WER bringt.

5.2.2 Aufbau

Der Aufbau ist analog zum vorherigen Versuch, daher werden die Daten wiederverwendet.

5.2.3 Ergebnisse

Die erreichten Veränderungen der WER zeigt sich auch in dieser Ansicht durchzogen. Als Beispiel dafür wird der Prozessor Nvidia Maxine mit dem Effekt "denoiser" und der intensity ratio 0.2 aus Abbildung 5.3 herangezogen. Dabei ist gut ersichtlich, dass die Fehlerrate mehrheitlich gesenkt wurde durch die Anwendung des Prozessors, jedoch kam es gelegentlich zu bemerkenswerten Verschlechterungen.
Die Auswertung der anderen Prozessoren zeigen jeweils ein ähnliches Bild. Bei Deep Xi ist gut erkennbar, dass die Fehlerrate mehrheitlich angehoben wurde, aber in eignen wenigen Fällen, wurde sie drastisch gesenkt. Bei der Anwendung des Prozessors von Dolby wurde ersichtlich, dass kleine Fehlerraten oft weiter gesenkt wurden, hohe Fehlerraten hingegen wurden oft stark angehoben.

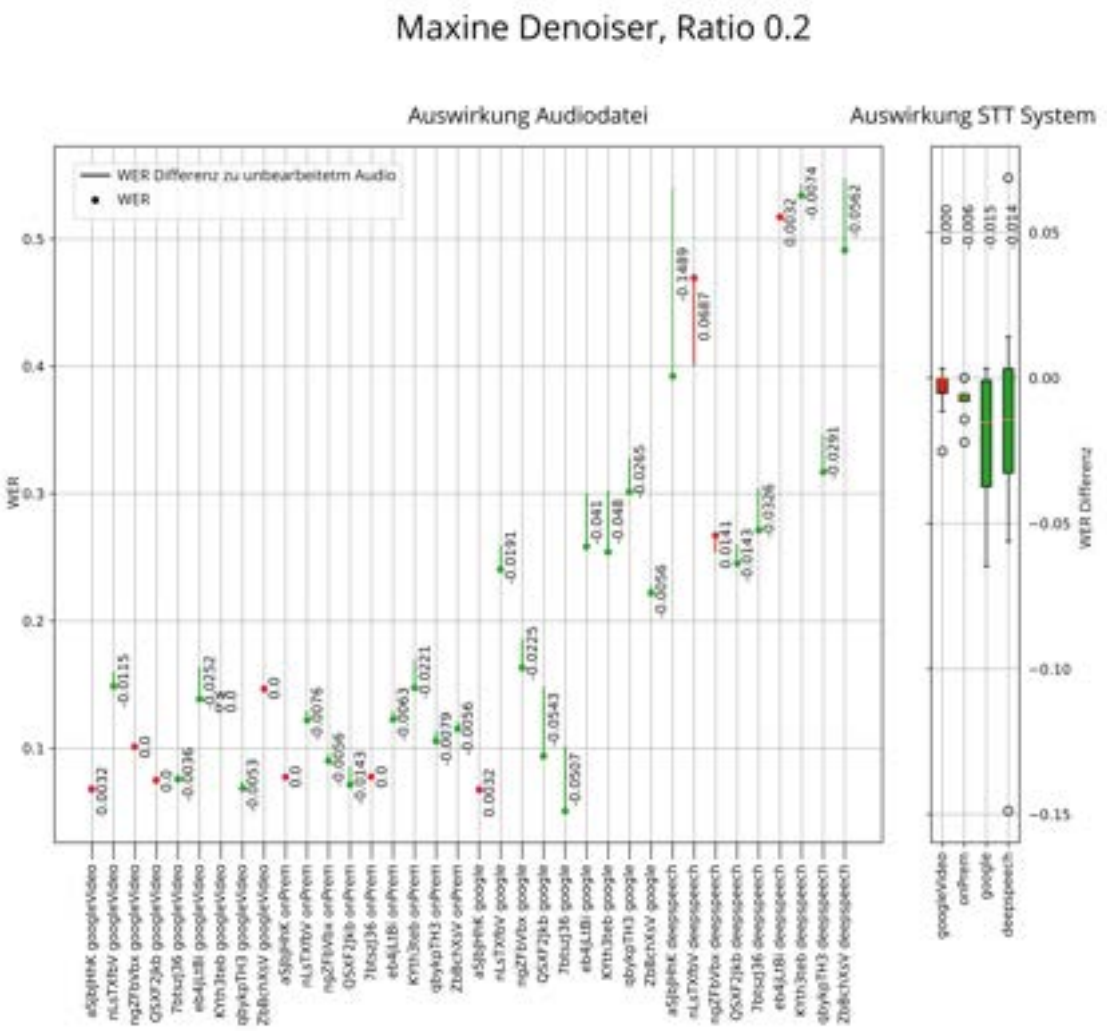

Abbildung 5.2.: Auswirkung von Nvidia Maxine auf die WER

5.2.4 Diskussion

Mit Nvidia Maxine können mehrheitlich Verbesserungen der Transkripte erzielt werden, jedoch ist nicht bekannt unter welchen Umständen das gelingt und unter welchen nicht. Es besteht aber die Möglichkeit, dass durch das Finden der optimalen Parameter für Maxine die Fehlerrate weiter gesenkt werden könnte.

5.3 Finden der Optimalen Intensity Ratio für Nvidia Maxine

5.3.1 Ziel

Bisher wurde bei der Verwendung von Nvidia Maxine immer mit den Intensity Ratios 0.1, 0.2, 0.3, ..., 1.0 gearbeitet, da nicht bekannt war welche die besten Ergebnisse liefert. Um für weitere Abklärungen Rechenzeit zu sparen und gleichzeitig die Fehlerraten etwas zu senken, soll für die Effekte ”denoise” und ”dereverb” sowie die Kombination der beiden die jeweils optimale Intensity Ratio bestimmt werden.

5.3.2 Aufbau

Es werden erneut die bereits bestehenden Daten verwendet. Als erstes wird berechnet, um wie stark jede Intensity Ratio die Fehlerrate verändert hat. Dann werden Punkte gebildet, die die Intensity Ratio als X-Koordinate haben und die Erreichte WER Differenz als Y-Koordinate. Durch die so entstandene Punktewolke wird schliesslich ein Polynom zweiten Grades gelegt. Dies da in den vorherigen Experimenten der Eindruck entstand, dass eine Intensity Ratio von 0.1 die WER nur minimal verändert, gegen 0.5 durchaus die WER verkleinert wird und gegen 1.0 stieg sie oft deutlich an. Dies deutet auf eine typische Form eines Polynoms zweiten Grades hin. Auf diesem Polynom wird schliesslich nach dem Minimum gesucht, was der grössten WER Reduktion entspricht.

5.3.3 Ergebnisse

Die oben genannten Schritte wurden für alle drei Effekte und für die bis hierhin verwendeten Speech to Text Systeme durchgeführt und in Abbildung 5.3 geplottet. Die Ergebnisse wurden zur besseren Lesbarkeit auch in Tabelle 5.1 aufgeführt.

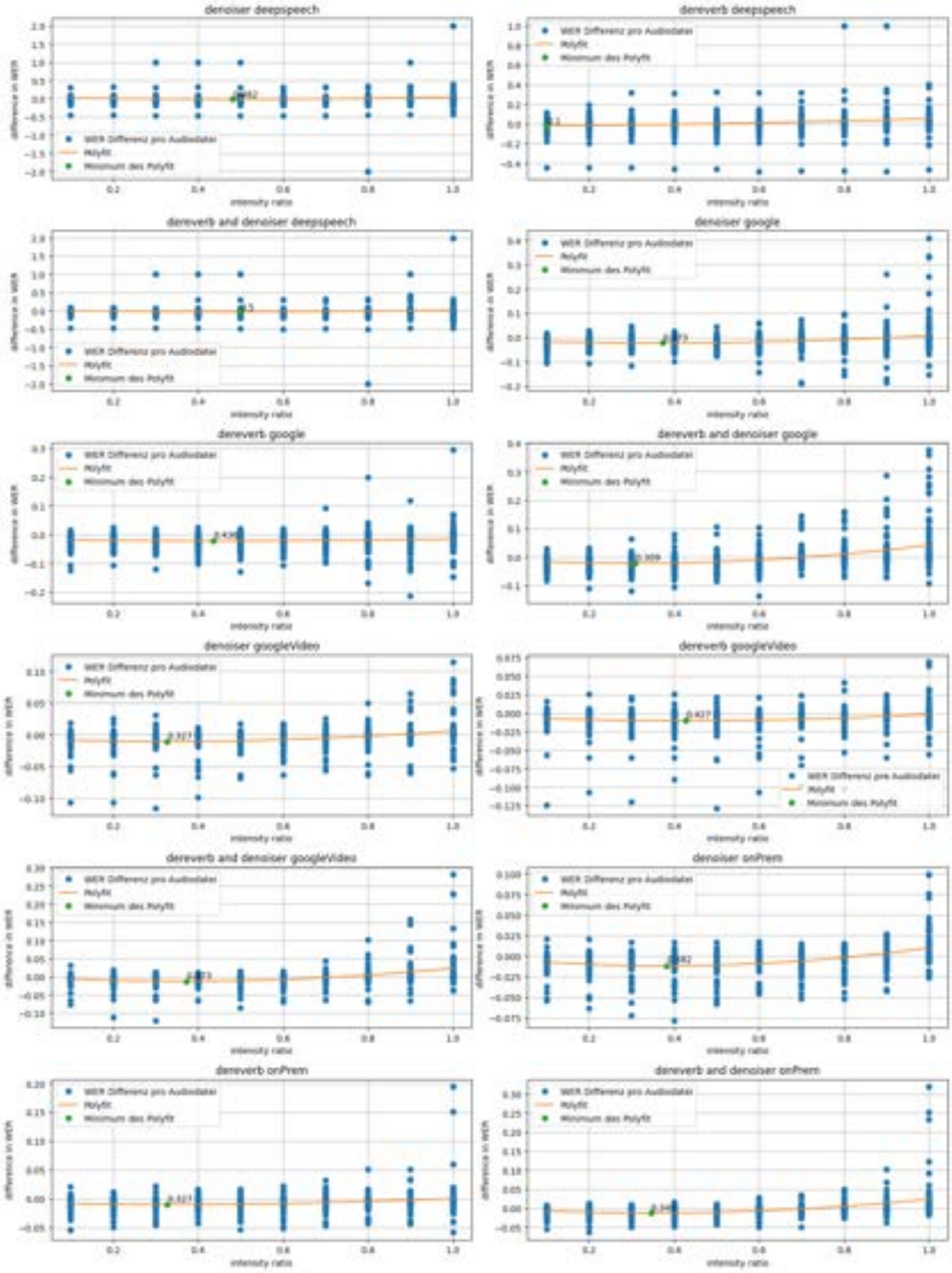

Abbildung 5.3.: Auswirkung von Nvidia Maxine auf die WER

STT System	Maxine Parameter	Optimale Intensity Ratio
DeepSpeech	Denoiser	0.482
DeepSpeech	Dereverb	0.1
DeepSpeech	Denoiser + Dereverb	0.5
Google	Denoiser	0.373
Google	Dereverb	0.436
Google	Denoiser + Dereverb	0.309
GoogleVideo	Denoiser	0.327
GoogleVideo	Dereverb	0.427
GoogleVideo	Denoiser + Dereverb	0.373
OnPrem	Denoiser	0.382
OnPrem	Dereverb	0.327
OnPrem	Denoiser + Dereverb	0.345
Durchschnitt		0.365

Tabelle 5.1.: Optimale Intensity Ratios

Wie sich zeigt ist Nvidia Maxine mit einer Intensity Ratio von ca. 0.365 am effektivsten.

5.3.4 Diskussion

In Abbildung 5.3 ist zu sehen, dass das gefittete Polynom eine sehr flache Krümmung aufweist. Es scheint damit im Allgemeinen keinen grossen Unterschied zu machen, welche Intensity Ratio gewählt wird. Eine einschneidende Limitation war sicherlich auch, dass die Intensity Ratio lediglich in Schritten von 0.1 verändert wurde und so die blaue Punktewolke unausgeglichen ist. Sollte dieses Experiment wiederholt werden, muss die Intensity Ratio zufällig gewählt werden beim Erzeugen der Daten, die für den Polyfit benutzt werden.

5.4 Einfluss der Audiolänge auf die Veränderung der WER

5.4.1 Ziel

Da schon sehr früh entschieden wurde nur mit zweiminütigen Audiodateien zu arbeiten, stellte sich die Frage, ob die Ergebnisse anders ausfallen würden, wenn längere Aufnahmen verwendet worden wären. Dies soll in diesem Experiment untersucht werden.

5.4.2 Aufbau

Es werden drei Audios aus dem Rev Korpus ausgewählt, die 15 Minuten oder länger sind. Diese wurden jeweils nach zwei, fünf und fünfzehn Minuten geschnitten, anschliessend von den Präprozessoren verarbeitet und schliesslich mit DeepSpeech transkribiert. Abschliessend wurde die WER gegenüber dem entsprechenden Abschnitt des Referenztextes berechnet. Es wurden bewusst nur drei Audios ausgewählt, da bei Verwendung der Präprozessoren alle resultierenden 225 Dateien transkribiert werden müssen. Jede zusätzliche Aufnahme erhöht den Rechenaufwand erheblich.

5.4.3 Ergebnisse

Es hat sich gezeigt, dass die Änderung der Fehlerrate nur minimal variierte, auch wenn die Länge der Aufnahmen um ein Vielfaches grösser werden. Jedoch wurde die Anzahl der Ausreisser mit zunehmender Audiolänge deutlich kleiner. Viele dieser Ausreisser wurden in der Abbildung 5.4 abgeschnitten, damit die Boxplots nicht zu flach ausfallen.

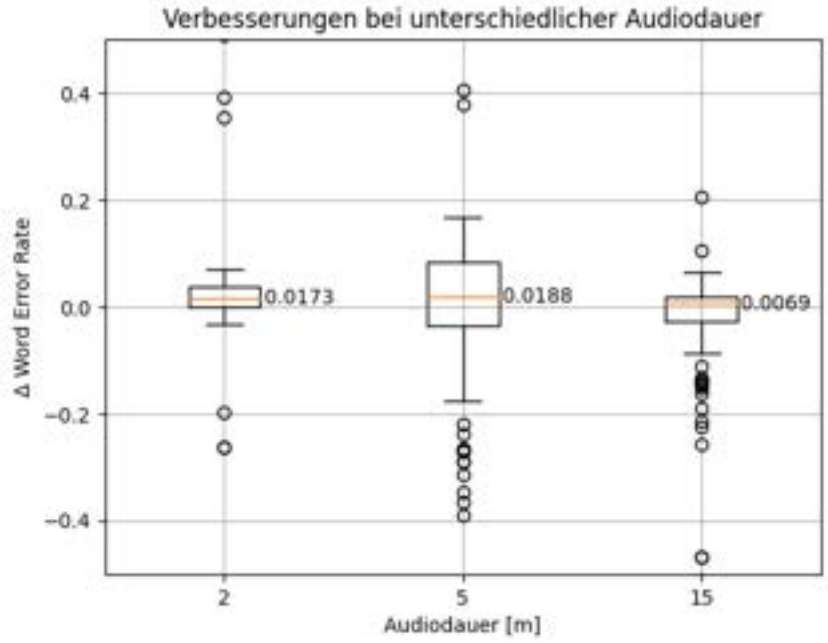

Abbildung 5.4.: Auswirkung der Audiodauer auf die Veränderung der WER

5.4.4 Diskussion

Die Länge der Audioaufnahmen scheint irrelevant zu sein für die Präprozessoren. Die Ausreisser lassen aber deutlich erkennen, dass das korrekte Erkennen eines Wortes bei kurzen Audios deutlich mehr ins Gewicht fällt als bei längeren.

5.5 Speech Enhancement Auswirkungen auf die WER mit DeepSpeech

5.5.1 Ziel

Das Ziel dieses Abschnitts ist es, die Präprozessoren auf Daten mit hoher Varianz zu evaluieren. Es soll ersichtlich werden, wie sich die Speech Enhancement Verfahren auf die Transkription auswirken, und entsprechend soll die WER ermittelt werden. Die Ergebnisse dienen dann als Grundlage für weitere Experimente zur Minimierung der Wortfehlerrate.

5.5.2 Aufbau

Alle 463 Audioclips aus dem bereinigten Rev-Audiokorpus wurden mit dem vielversprechendsten Präprozessor bearbeitet. Die unbearbeiteten/bearbeiteten Audioclips wurden dann mit DeepSpeech transkribiert und mit dem Referenztext verglichen, um die WER zu berechnen.

5.5.3 Ergebnisse

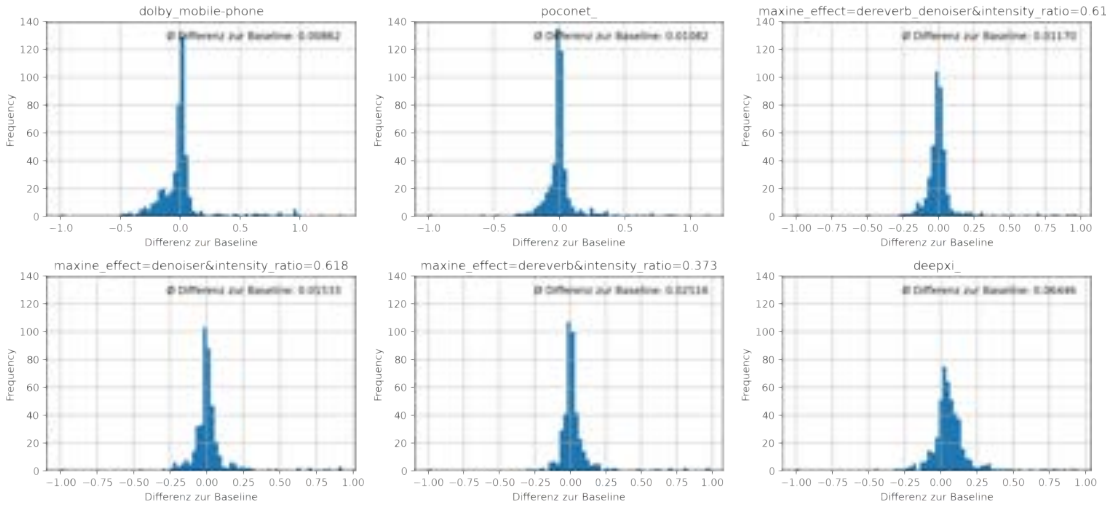

Abbildung 5.5.: Die Verteilung der WER-Differenzen der prozessierten Audiodateien im Vergleich zur Basislinie mit DeepSpeech.

Die Differenzen zwischen WER und der Basislinie können als Histogramm dargestellt werden, wie in 5.5 zu sehen ist. Dabei fällt auf, dass es sowohl Verbesserungen als auch Verschlechterungen geben kann. Die Durchschnittswerte zeigen, dass es insgesamt zu

einer Verschlechterung kommt. Dolby.io hat die geringste Verschlechterung mit 0.7% und Deep Xi die grösste mit 6.4%.
Da sich herausstellt, dass es keinen allgemein besten Präprozessor gibt, kann man nun einen theoretischen Best-Case-Wert berechnen, bei dem die jeweils beste Speech Enhancement Verfahren für jeden Audioclip eingesetzt wird. Dies führt zu einer Best-Case-WER, der nur erreicht werden kann, wenn man den Referenztext hat und alle Audiodateien transkribiert und daraus die WERs berechnet.

	Durchschnitt	**Median**
Baseline WER	0.3852	0.3455
Best-Case WER	0.3139	0.3365
Differenz (WER_{Δ})	-0.0713	-0.0090

Tabelle 5.2.: Baseline im Vergleich zum theoretischen Best-Case.

Wie in der Tabelle 5.2 zu sehen ist, liegt die durchschnittliche WER bei etwa 0.39. Wird stets der beste Präprozessor verwendet, kann die Wortfehlerrate um 0.071 auf 0.314 gesenkt werden.

Anzahl Fälle als bester Präprozessor		
Prozessor mit Parameter	**Anzahl**	**Prozent**
dolby_mobile-phone	129	27.86
no-processing_	98	21.17
poconet_	88	19.01
maxine_effect=dereverb_denoiser&intensity_ratio=0.618	57	12.31
maxine_effect=denoiser&intensity_ratio=0.618	52	11.23
maxine_effect=dereverb&intensity_ratio=0.373	39	8.42
deepXi_	0	0

Tabelle 5.3.: Übersicht in wie vielen Fällen welcher Prozessor der Beste war oder kein Verfahren angewendet werden sollte.

Die Best-Case-Szenarien können nach dem Verfahren zur Speech Enhancement aufgeschlüsselt werden, woraus sich die Tabelle 5.3 ergibt. Dolby.io bringt am häufigsten eine Verbesserung. In 98 Fällen kann die WER jedoch nicht mit einer Methode verbessert werden (no-processing_). Deep Xi bringt nie die beste Verbesserung.

5.5.4 Diskussion

Es wurde deutlich, dass es keinen allgemein besten Präprozessor gibt, der immer angewendet werden kann. Es ist davon auszugehen, dass im Audiomaterial Artefakte entstehen können, welche die Inkongruenzfaktoren gegenüber der trainierten STT-Engine

verstärken. Das Beispiel 85XPGxwo[1] zeigt, dass in diesem Audioclip mit Dolby.io die Lautstärke des Sprechers erhöht wurde, was sich ebenfalls positiv auf das menschliche Verständnis auswirkt. Die Wortfehlerrate wurde dadurch jedoch nicht verbessert, im Gegenteil die Genauigkeit der Transkription wurde gar verschlechtert (WER_{Δ} 2.18). Bemerkenswert ist, dass Dolby.io meistens das beste Speech Enhancement Verfahren ist. Es ist zu vermuten, dass dies auf die erweiterten Funktionalitäten des Dienstes zurückzuführen ist. Die Lautstärkenormalisierung konnte im Beispiel RT6Y8NC7[1] viele Löschungen verhindern (WER_{Δ} -0.42). MAXINE dereverb war nur in 47 Fällen die beste Wahl als Präprozessor, da es nur den Nachhall aus den Aufnahmen entfernt. In den Fällen wo MAXINE dereverb die beste Wahl war, ist aber auch immer Nachhall in den Audiodateien hörbar, z.B. bei TixnNjiz[1] (WER_{Δ} -0.15) oder 036JDvdK[1] (WER_{Δ} -0.09).
Im Falle eines Transkriptionsdienstes ist der Referenztext nicht verfügbar. Daher kann der beste Präprozessor nicht mit absoluter Sicherheit bestimmt werden. Aus diesem Grund wurde in den folgenden Experimenten versucht, den besten Präprozessor vorauszusagen, um die WER-Reduktion 7.1% anzunähern.

5.6 Vorhersage des besten Prozessors durch SVM

5.6.1 Ziel

SVM_BestProc+NoProc_X

Durch Charakteristika/Features einer Aufnahme sollte das beste Speech Enhancement Verfahren bestimmt werden, sofern ein solches existiert. Eine Support Vector Machine (SVM) sollte die Indikatoren auf dem Rev-Korpus lernen, die auf den besten Präprozessor für DeepSpeech hinweisen, oder keine Verfahren unterbreiten, wenn es kein solches existiert.

SVM_BestProc_X

Angenommen man weiss, dass ein Audioclip verbessert werden kann, sollte man in der Lage sein, mit Hilfe eines SVM vorherzusagen, welcher Präprozessor der Beste ist.

SVM_ImproveYesNo_X

Es soll erkannt werden, wann ein beliebiger Präprozessor angewendet werden kann, ohne die Audioqualität zu verschlechtern, bzw. wann der Präprozessor Dolby.IO mobile_phone angewendet werden kann, ohne die Audioqualität um mehr als 20% zu verschlechtern. Damit im Durchschnitt über alle Audioclips man trotzdem eine Verbesserung erzielt werden kann.

[1] Name des Audioclips - Datei ist im Rev-Korpus oder unter `https://audio-denoising.netlify.app` zu finden

5.6.2 Aufbau

Audio Features

Um die Audioclips zu charakterisieren, wurden verschiedene Merkmale als Input für die SVM berechnet. Lautstärke und Noise-Eigenschaften [53] wurden mithilfe des Dolby.io-Dienstes berechnet. Die Media Analyze API bietet Einblicke in die Audioqualität [54] einer Aufnahme. Die Tabelle 5.4 zeigt und erklärt die verwendeten Metriken.

Name & Beschreibung	Hypothese
bitrate Audio Bitrate (Bits pro Sekunde)	Kann Aufschluss darüber geben, wie gut die Auflösung einer Aufnahme ist, und weist möglicherweise auf einen guten Präprozessor hin.
loudness_measured gemessene Lautstärke nach ITU-R BS.1770	Gibt Auskunft darüber, wie "lautëine Aufnahme ist und verweist möglicherweise auf einen Präprozessor wie Dolby, der die Lautstärke korrigieren könnte.
loudness_range Spannweite der Lautstärke gemessen in LU	Zeigt an, wie weit die Bandbreite der Lautstärke reicht und könnte daher auf bestimmte störende Geräusche, wie z. B. Sirenen, hinweisen.
sample_peak Abgetastete maximal Lautstärke	Die abgetastete maximale Lautstärke könnte auf lauten Lärm hinweisen.
true_peak Effektive Maximal Lautstärke	Die effektive Maximallautstärke könnte auf lauten Lärm hinweisen.
bandwidth Bandbreite	Die Audiobandbreite könnte Auskunft darüber geben, um welche Art von Audio es sich handelt. Z.B. Telefon, Studio, Interview-Aufnahme, etc.
snr_average stationäres Rauschen, gemessen als Signal-Rausch-Verhältnis	Ermittelt, wie viel Hintergrund-/ stationäre Störgeräusche vorhanden sind. Siehe 2.4
level_average Pegel des stationären Rauschens über die gesamte Länge	Zeigt an, wie laut das Grundrauschen ist und könnte möglicherweise die Umgebung charakterisieren.
speech_percentage Sprachanteil in Prozent	Zeigt an, wie viel Sprache in einer Aufnahme vorhanden ist und könnte zusammen mit der Cross STT WER die Transkriptionsschwierigkeit widerspiegeln.
speech_num_sections Anzahl Sprachabschnitte	Gibt Auskunft darüber, wie viele Sprachsegmente vorhanden sind und könnte den Inhaltstyp charakterisieren, z. B. Interview, öffentliche Rede usw.
silence_percentage Anteil an Stille in Prozent	Der Anteil der Stille könnte auch den Audiotyp und/oder die Umgebung widerspiegeln.
silence_num_sections Anzahl Stille Abschnitte	Die Anzahl Still-Segmente könnte auch den Audiotyp und/oder die Umgebung widerspiegeln.

Tabelle 5.4.: Dolby.io Metriken Übersicht und Beschreibungen

Um Eigenschaften zu erfassen, die den Inhalt beschreiben, wurde die librosa-Bibliothek zur Berechnung von Spektral-Features eines Audioclips verwendet. Diese Features werden gerne verwendet um Musikgenre zu klassifizieren [55] [56]. In der Tabelle 5.5 sind die verwendeten Spektral-Features zu entnehmen.

Verwendete librosa Features	
chroma_stft	Chromagramm aus dem Signal
spec_cent	spektraler Schwerpunkt
spec_bw	spektrale Bandbreite
rolloff	Abklingfrequenz
zcr	Nulldurchgangsrate

Tabelle 5.5.: librosa Feature Übersicht und Beschreibungen

Des Weiteren wurden mit Hilfe des Moduls speechmetrics [57] MOS Annäherungswerte (mos_average) und das Speech-to-Reverberation Modulation Energy Ratios (srmr_average) berechnet. mos_average ist ein Inferenzwert eines DNNs, der zur Vorhersage des MOS-Wertes trainiert wurde [58]. srmr_average ist eine Metrik zur Schätzung der Verständlichkeit bei Störgeräuschen und Nachhall, wobei 0 für schlechte Qualität und 1 für gute Qualität steht [59]. Auch wurde die im Kapitel 2.3.3 erklärte Cross WER mit DeepSpeech und Apple berechnet. Benannt wurden die Features ”apple_vs_deepspeech” oder ”deepspeech_vs_apple”.

Datensatz

Die Trainingsdaten wurden für die Modelle immer gleich initialisiert. Der bereinigte Rev-Korpus wurde genommen und in 20% Testdaten und 80% Trainingsdaten geteilt.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.20,
    random_state=42,
    stratify=y)
```

Die Zufallszustandsvariable wurde immer auf 42 gesetzt, und durch Stratifizierung wurde sichergestellt, dass die kleinen Klassen in beiden Test- und Trainingssplits vertreten sind.

Dummy Classifier

Es wurde ein ”uniform” Dummy Classifier verwendet, welcher die Vorhersagen auf Basis der Verteilung der Klassen in der Testmenge trifft [60]. Die erzielten Werte dienen als Referenzwerte für den Vergleich der SVMs.

SVM_BestProc+NoProc_AllFeat

Die erste SVM_BestProc+NoProc_AllFeat wurde mit allen Features trainiert, mit dem Ziel das beste Verfahren (”dolby_mobile-phone”, ”maxine_effect=dereverb&intensity_ratio=0.373”, ”maxine_effect=dereverb_denoiser&intensity_ratio=0.618”, ”maxine_effect=denoiser&intensity_ratio=0.618”, ”poconet_”, ”deepxi_” oder ”no-processing_”) vorauszusagen. Mittels GridSearchCV und StratifiedKFold wurden die besten Parameter ermittelt, die den f1_macro maximieren. Der f1_macro-Score wird aus dem arithmetischen Mittel aller F1-Scores pro Klasse berechnet und wird verwendet, wenn alle Klassen gleich wichtig sind. Um die Werte der Features zu standardisieren, wurde der StandardScaler aus der Scikit Bibliothek verwendet. Damit wird jedes Feature skaliert, indem der Mittelwert subtrahiert und durch dessen Standardabweichung dividiert wird, um die Verteilung so zu verschieben, dass sie einen Mittelwert von Null und eine Standardabweichung von Eins aufweist [61].

SVM_BestProc+NoProc_FeatSel

Feature Selection wird verwendet, um die Anzahl der Features zu reduzieren. Da Features mit geringer Auswirkung entfernt werden, wird das Training der SVM beschleunigt und kann dazu beitragen, Overfitting zu verhindern. Die SVM_BestProc+NoProc_FeatSel verfolgt das gleiche Ziel wie die SVM_BestProc+NoProc_AllFeat, allerdings wurde die Anzahl der Merkmale reduziert. In einem ersten Schritt wurden die Werte des Pearson-Korrelationskoeffizienten berechnet.

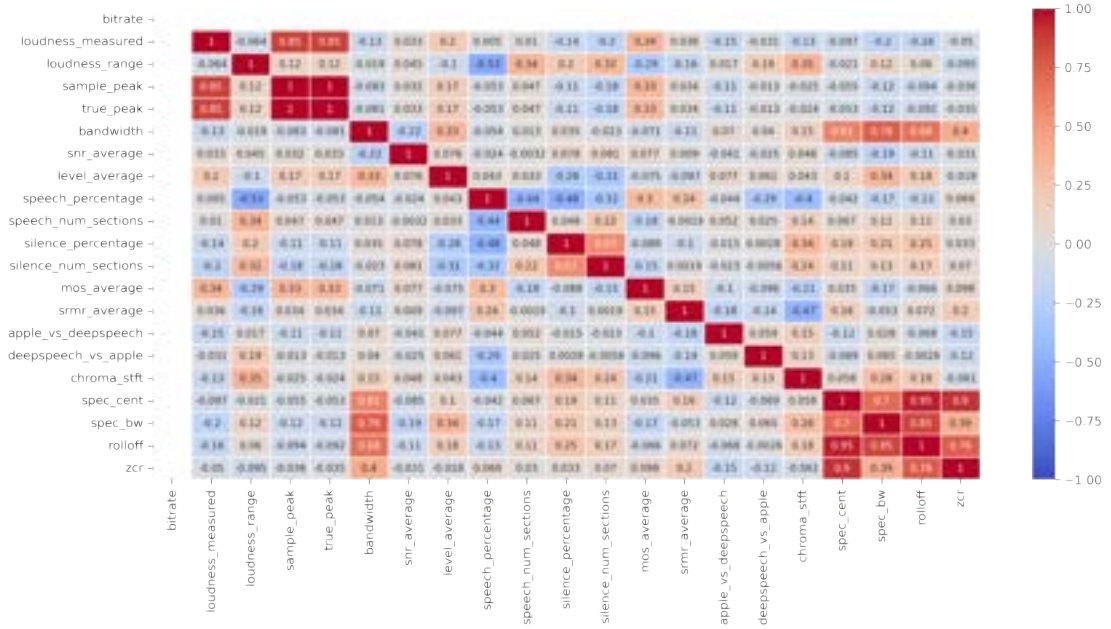

Abbildung 5.6.: Korrelationsmatrix, die folgenden Features entfernt wurden entfernt: ”sample_peak”, ”true_peak”, ”true_peak”, ”spec_bw”, ”onPrem_vs_google”, ”deepspeech_vs_apple”, ”spec_cent”, ”rolloff”

Korrelationswerte mit $r > 0.75$ und $r < -0.75$ wurden bereinigt, indem nur ein Wert aus diesem Paar verwendet wurde, da sie dasselbe aussagen. In der Abbildung 5.6 sind die entfernten Features ersichtlich.
In einem zweiten Schritt wird ein Random Forest verwendet, um die Wichtigkeit der Features zu bestimmen. Mit n_estimators=100 wird die Feature Importance aller Features berechnet. Die Feature Importance wird berechnet als die Abnahme der Impurity eines Nodes, gewichtet mit der Wahrscheinlichkeit, diesen Node zu erreichen.

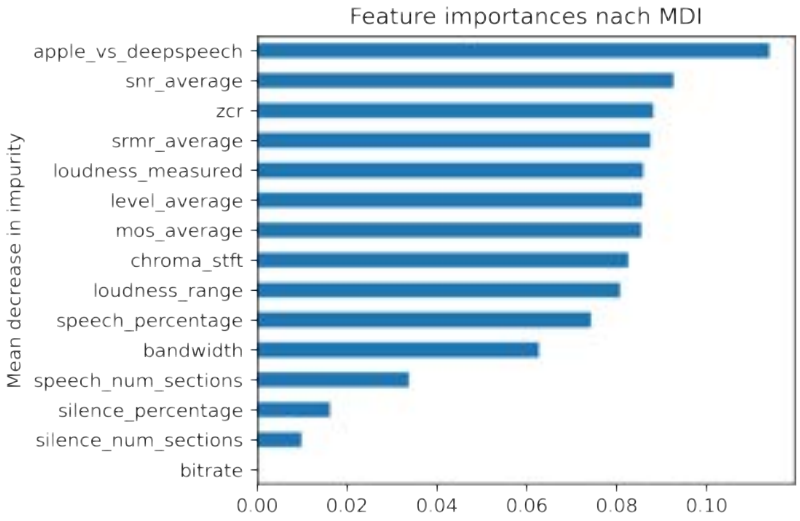

Abbildung 5.7.: Die Features ”loudness_measured”, ”loudness_range”, ”snr_average”, ”level_average”, ”speech_percentage”, ”mos_average”, ”srmr_average”, ”apple_vs_deepspeech”, ”chroma_stft”, ”zcr” wurden verwendet.

Die Node-Wahrscheinlichkeit kann durch die Anzahl der Proben, die den Node erreichen, geteilt durch die Gesamtzahl der Proben, berechnet werden [62]. Je höher der Wert, desto wichtiger ist das Feature. Aus diesen Ergebnissen ergibt sich der Plot, an dem man die Wichtigkeit der Features ablesen kann. Im Diagramm 5.7 sind die verwendeten Features ersichtlich.

SVM_BestProc+NoProc_AllSel+NoCorr

Wie bei SVM_BestProc+NoProc_FeatSel werden bei SVM_BestProc+NoProc_AllSel+NoCorr alle korrelierende Features entfernt. Die Wichtigkeit der Features wurde jedoch nicht ausgewertet.

SVM_BestProc_AllFeat

Die SVM_BestProc_AllFeat arbeitet nur mit Audiodaten aus dem Rev-Korpus, die sich verbessern lassen. Das bedeutet, dass die Kategorien wie folgt lauten: ”dolby_mobile-phone”, ”maxine_effect=deverb&intensity_ratio=0.373”, ”maxine_effect=deeerb_denoiser&intensity_ratio=0.618”, ”maxine_effect=denoiser&intensity_ratio=0.618”, ”poconet_” und ”deepxi_”. Es wurden alle Features verwendet. Diese SVM soll zeigen, ob Präprozessoren sich überhaupt vorhersagen lassen.

SVM_ImproveYesNo_AnyProc

SVM_ImproveYesNo_AnyProc ist ein binärer Klassifikator und wurde erstellt, um vorherzusagen, ob eine Preprocessing angewandt werden sollte oder nicht. Alle Features wurden übergeben und wie zuvor wurden die optimalen SVM Parameter bestimmt.

SVM_ImproveYesNo_Dolby

SVM_ImproveYesNo_Dolby zielt darauf ab, Verschlechterungen von mehr als 20% durch den Dolby.IO-Enhancer zu erkennen. Auch hier wurden alle Features geprüft und die optimalen SVM-Parameter bestimmt.

5.6.3 Ergebnisse

Vorhersage des besten Verfahrens

Die Tabelle 5.6 zeigt die Ergebnisse der SVMs, um das beste Verfahren vorherzusagen. Der Best Case auf dem verwendeten Testset wäre eine WER-Verbesserung von -0.08916 im Mittel und -0.03184 im Median, wenn immer das beste Verfahren zum Einsatz käme.

				Classifier WER_{Δ}	
Name	**Parameter**	**accuracy**	**f1_macro**	**Durchschnitt**	**Median**
Dummy Classifier	uniform	0.16	0.13	0.00196	0
SVM_BestProc+NoProc_AllFeat	'C': 1000, 'kernel': 'linear'	0.28	0.18	-0.01215	0
SVM_BestProc+NoProc_FeatSel	'C': 10, 'kernel': 'linear'	0.29	0.13	-0.01542	0
SVM_BestProc+NoProc_AllSel+NoCorr	'C': 100, 'kernel': 'linear'	0.31	0.19	-0.01634	0

Tabelle 5.6.: SVM Ergebnisse bei der Vorhersage des besten Verfahrens

Würde man den Vorhersagen des Dummy Classifiers folgen, erhielte man ein WER_{Δ} von 0.0196 gegenüber der Baseline. Dies würde bedeuten, dass es zu einer Verschlechterung führte und man ein besseres Ergebnis ohne Präprozessoren erhalten würde. Alle drei SVMs erzielen ein negatives WER_{Δ}, jedoch bewegen sie sich um die ein Prozent. Der Median ist immer bei 0, da sich bei einer häufigen Vorhersage von no-processing_ viele WER_{Δ} von 0 ergeben. Für genauere Informationen zu den Prediction, sind im Anhang A.2 sind die Wahrheitsmatrizen / Confusion Matrices zu finden.

Vorhersage des besten Präprozessors

SVM_BestProc_AllFeat wurde für die Anwendung auf einen theoretischen Korpus trainiert, bei dem alle Audioclips mit einem bestimmten Präprozessor verbessert werden können. Im Best-Case Szenario könnte die WER um -0.10681 im Durchschnitt und -0.04384 im Median im Vergleich zur Baseline reduziert werden.
Die Vorhersagen sind im Vergleich zu den letzten SVMs besser und erreichte ein WER_{Δ} von 5%.

				Classifier WER_{Δ}	
Name	**Parameter**	**accuracy**	**f1_macro**	**Durchschnitt**	**Median**
SVM_BestProc_AllFeat	'C': 10, 'kernel': 'linear'	0.39	0.21	-0.05055	-0.00456

Tabelle 5.7.: SVM Ergebnisse bei der Vorhersage des besten Präprozessors

SVM - Binary Classifier: Präprozessor Anwenden

SVM_ImproveYesNo_AnyProc konnte no_processing nur in 2 Fällen korrekt erkennen.

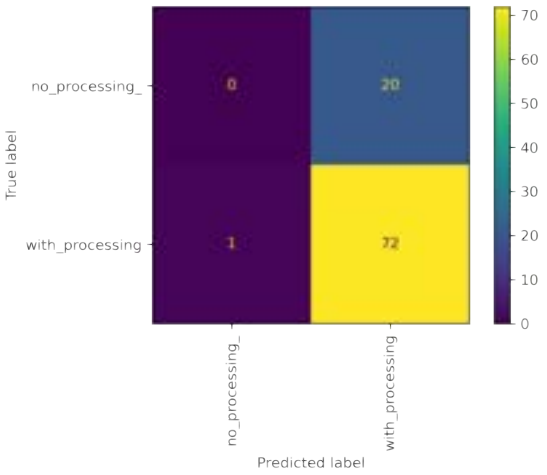

Abbildung 5.8.: Confusion Matrix vom SVM_ImproveYesNo_AnyProc

Es wurde eine Accuracy von 0.77 erreicht, was jedoch nicht die Leistungsfähigkeit der SVM widerspiegelt, da die Kategorien nicht gleich gross sind. Der F1-Makro berücksichtigt dieses Ungleichgewicht und liefert einen Wert von 0.44.

SVM - Binary Classifier: Dolby Anwenden

Die SVM konnte keine Indikatoren lernen, die darauf hinweisen, wann Dolby eine Verschlechterung von mehr als 20% verursacht. Die Klasse "false" enthält die Audioclips, bei denen die Verarbeitung eine WER-Verschlechterung von mehr als 20% verursacht.

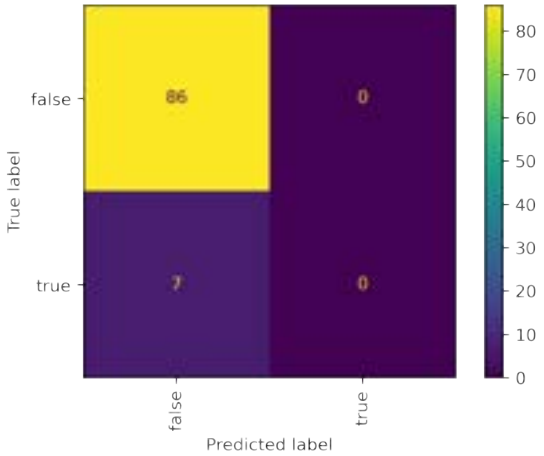

Abbildung 5.9.: Confusion Matrix vom SVM_ImproveYesNo_Dolby

”true” enthält die Audioclips, die verbessert werden können oder die keine Verschlechterung von mehr als 20% verursachen. Die SVM hat im Testset ständig ”falsch” vorhergesagt.

5.6.4 Diskussion

Die Ergebnisse zeigen, dass die Features nicht informativ genug sind, um den besten Präprozessor vorherzusagen. Die SVMs SVM_BestProc+NoProc_X bringen unbedeutende Verbesserungen im Bereich von einem Prozent. SVM_BestProc_AllFeat würde gut funktionieren und könnte durchaus verwendet werden, um WER zu reduzieren. Dies setzt jedoch voraus, dass bekannt ist, welche Aufnahmen verbessert werden können. Die SVMs 3.X würden ein solches Ergebnis liefern, wenn sie dies lernen könnten, aber die Wahrheitsmatrizen zeigen, dass dies nicht möglich ist. Die verwendeten Features sind nicht aussagekräftig genug, um die Audioclips zu klassifizieren.

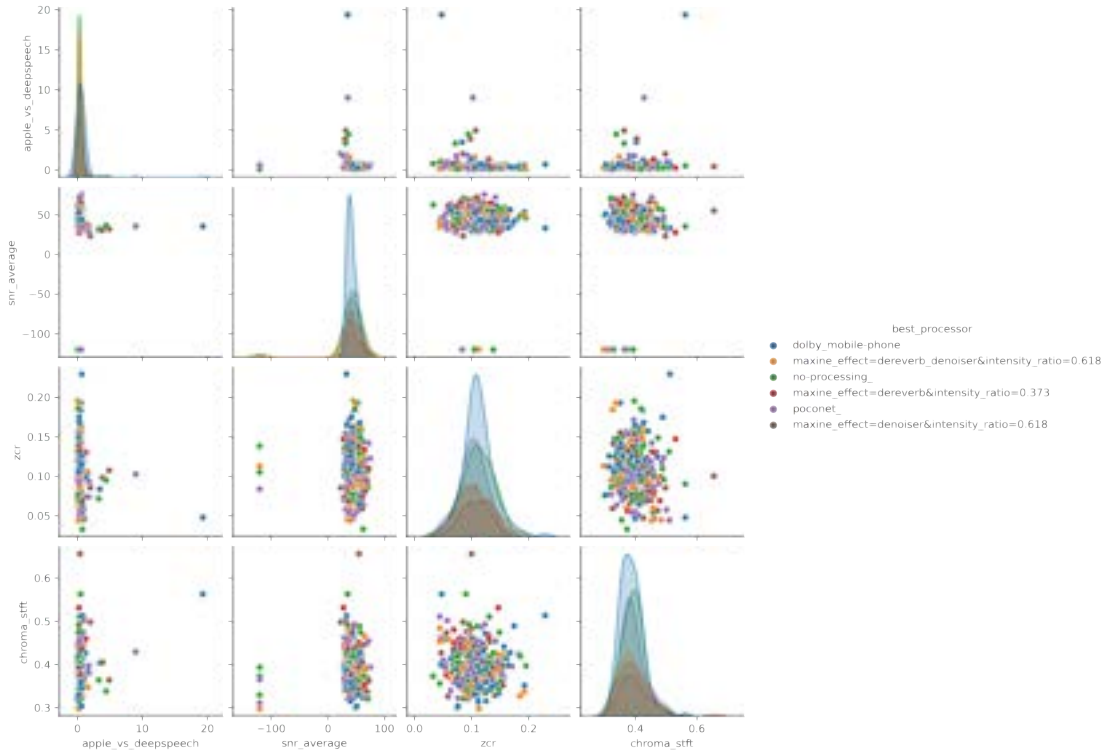

Abbildung 5.10.: Pairplot mit den vier besten Features.

Im Pairplot 5.10 ist zu erkennen, dass die Klassen nicht geclustert erscheinen und keinem Muster folgen. Interessanterweise ist das Merkmal ”apple_vs_deepspeech” das mit Abstand das beste Feature zur Trennung der Klassen. Dies könnte darauf hindeuten, dass mehr Eigenschaften der ASR-Systeme und der Präprozessoren als Eigenschaften eingebettet werden sollten. Wenn die Cross STT WER nach der Verbesserung aufgeschlüsselt wird, kann ein Histogramm 5.11 erstellt werden.

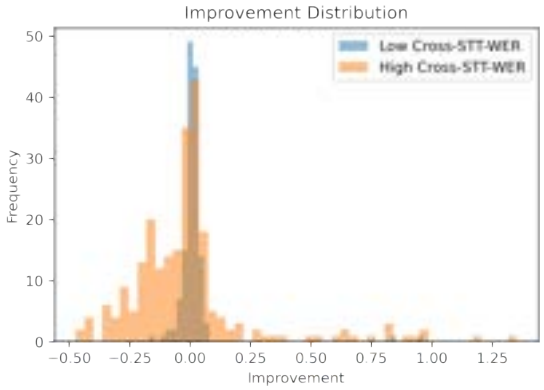

Abbildung 5.11.: Audioclips mit einer Cross-STT höher als 0.3 gelten als ”hoch” bzw. ”high”

Dies zeigt, dass grundsätzlich die schwierigeren Transkripte mehr verbessert werden können als einfache Transkripte. Allerdings können schwierige Transkripte auch stark verschlechtert werden.

5.7 Klassierung mit Dolby und librosa Features durch Neuronales Netz

5.7.1 Ziel

Wie schon bei der SVM soll eine Abbildung von den Dolby und librosa Features zur optimalen Speech Enhancement Implementierung gefunden werden, um die dort erzielten Ergebnisse zu bestätigen. Falls die Ergebnisse bestätigt werden, wird der Verdacht, dass die Features nicht repräsentativ sind, erhärtet. Da dieser Verdacht bereits sehr stark ist, wurde schon nach alternativen Features gesucht. Um aber nicht die Features und gleichzeitig die verwendete Technologie zu ersetzen, wurde entschieden dieses Experiment als Übergang durchzuführen.

5.7.2 Aufbau

Mit Tensorflow wurde ein Neuronales Netz erstellt, dass auf den betreffenden Daten trainiert werden kann. Als Basis diente ein Artikel von Mauro Di Pietro der einen Überblick über den Aufbau eines Neuronalen Netzes bietet [63]. Dabei wurden Teile des Codes des Artikels übernommen und angepasst.
Das Training wurde mit einer Batchsize von 128 und 500 Epochen durchgeführt.
Der untenstehende Code zeigt den Aufbau des Neuronalen Netzes.

```
model = models.Sequential()
model.add(layers.Dense(32, activation='softplus', input_shape
   =(X_train.shape[1],)))
model.add(layers.Dropout(rate=0.2))
model.add(layers.Dense(32, activation='softplus'))
model.add(layers.Dropout(rate=0.2))
model.add(layers.Dense(32, activation='softplus'))
model.add(layers.Dropout(rate=0.2))
model.add(layers.Dense(32, activation='softplus'))
model.add(layers.Dropout(rate=0.2))
model.add(layers.Dense(16, activation='softplus'))
model.add(layers.Dropout(rate=0.2))
model.add(layers.Dense(7, activation='softmax'))
model.compile(optimizer='adam', loss='
   sparse_categorical_crossentropy', metrics=['accuracy'])
```

Struktur des ersten Neuronalen Netzes

5.7.3 Ergebnisse

Insbesondere bei der Validierung ist eine äusserst ungewöhnliche Entwicklung der Accuracy ersichtlich für die keine Erklärung gefunden werden konnte. Um sicher zu gehen, dass diese unübliche Form keine einmalige Anomalie ist, wurde das Training insgesamt dreimal durchgeführt. Dabei zeigte sich auch, dass die Accuracy je nach Training erheblich unterscheidet.

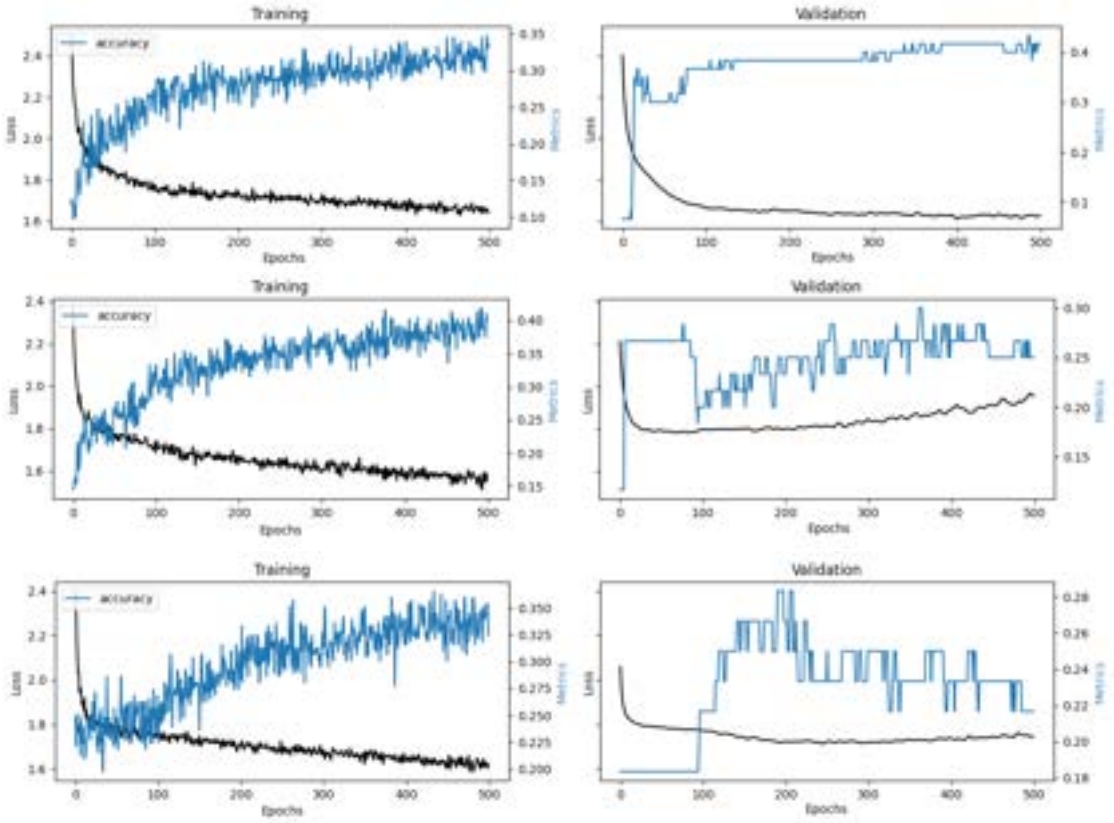

Abbildung 5.12.: Learning Curves dreier Trainingsläufe

5.7.4 Diskussion

Der Verlauf der Learning Curves ist insbesondere bei der Validierung äusserst ungewöhnlich. Wie diese kantigen Verläufe zustande gekommen sind, ist unklar. Es wird angenommen, dass dies ein Hinweis darauf ist, dass die gewünschte Abbildung von den Dolby und librosa Features zum optimalen Präprozessor nicht gefunden werden kann. Auch die tiefen Accuracy Werte deuten darauf hin. Des Weiteren ist gut ersichtlich, dass die erreichten Werte sich von Trainingslauf zu Trainingslauf deutlich unterscheiden. Der einzige Unterschied zwischen den Trainingsläufen war, welche Daten für das Training und welche für die Validierung genutzt wurden. Dies wird dahingehend interpretiert,

dass entweder zu wenige Daten zur Verfügung standen oder als ein weiterer Hinweis auf nicht repräsentative Daten.

5.8 Klassierung mit MFCCs durch Neuronales Netz

5.8.1 Ziel

Wie bei den vorherigen beiden Versuchen wurde auch hier das Ziel verfolgt, die optimale Speech Enhancement Implementation für eine gegebene Audiodatei vorherzusagen. Im Gegensatz zu den anderen beiden Experimenten aber nicht durch die Dolby und librosa Features sondern anhand der MFCCs. So soll der zentrale Schwachpunkt der vorherigen Experimente eliminiert werden.

5.8.2 Aufbau

Der Aufbau ist stark an einen Blogpost von Ketan Doshi [64] angelehnt, bei dem illustriert wird, wie sich "Urban Sounds" in Audiodateien klassieren lassen. Daher wurde auch das Framework von Tensorflow zu PyTorch gewechselt.

Die Audiodateien werden analog zum Blogpost aufbereitet, jedoch wurden einige Konfigurationen angepasst. Die Sample Rate wurde auf 16000 Hz gesetzt, da diese von den Speech Enhancement Implementierungen so gefordert wurde. Auch wird das Audio auf einen Channel reduziert, da die verschiedenen Channels sehr ähnlich zueinander sein würden und so der Speicherverbrauch halbiert werden kann. Schliesslich wurde auch die Länge aller 522 Audiodateien auf zwei Minuten gesetzt, da alle Dateien bereits diese Länge haben und es sehr schwierig sein würde anhand eines kurzen Ausschnittes auf die Charakteristiken des ganzen Audios zu schliessen.
Die Implementation des "Time Shifts"wurde übernommen, da der Zeitpunkt des Auftretens eines bestimmten Musters in den Daten eine untergeordnete Rolle spielen muss. Auch die Implementation von SpecAugment wurde übernommen, da diese der Beschreibung aus dem entsprechenden Paper [65] nahe ist.
Das Training wurde über 100 Epochen durchgeführt.
Der untenstehende Code zeigt den Aufbau des Neuronalen Netzes.

```
self.conv1 = nn.Conv2d(2, 243, kernel_size=(3, 3), stride=(2,
    2), padding=(1, 1))
self.relu1 = nn.ReLU()
self.bn1 = nn.BatchNorm2d(243)
init.kaiming_normal_(self.conv1.weight, a=0.1)
self.conv1.bias.data.zero_()
conv_layers += [self.conv1, self.relu1, self.bn1]
```

```
self.conv13 = nn.Conv2d(243, 81, kernel_size=(3, 3), stride
   =(2, 2), padding=(1, 1))
self.relu13 = nn.ReLU()
self.bn13 = nn.BatchNorm2d(81)
init.kaiming_normal_(self.conv13.weight, a=0.1)
self.conv13.bias.data.zero_()
conv_layers += [self.conv13, self.relu13, self.bn13]

self.conv13 = nn.Conv2d(81, 27, kernel_size=(3, 3), stride
   =(2, 2), padding=(1, 1))
self.relu13 = nn.ReLU()
self.bn13 = nn.BatchNorm2d(27)
init.kaiming_normal_(self.conv13.weight, a=0.1)
self.conv13.bias.data.zero_()
conv_layers += [self.conv13, self.relu13, self.bn13]

self.conv14 = nn.Conv2d(27, 9, kernel_size=(3, 3), stride=(2,
    2), padding=(1, 1))
self.relu14 = nn.ReLU()
self.bn14 = nn.BatchNorm2d(9)
init.kaiming_normal_(self.conv14.weight, a=0.1)
self.conv14.bias.data.zero_()
conv_layers += [self.conv14, self.relu14, self.bn14]

# Linear Classifier
self.ap = nn.AdaptiveAvgPool2d(output_size=1)
self.lin = nn.Linear(in_features=9, out_features=7)

# Wrap the Convolutional Blocks
self.conv = nn.Sequential(*conv_layers)
```

Dabei wurden auch mehrere verschiedene Aktivierungsfunktionen getestet, sowie die Anzahl Layer deutlich variiert. Der oben gelistete Code stellt dabei die finale Version dieses Experimentes dar.

5.8.3 Ergebnisse

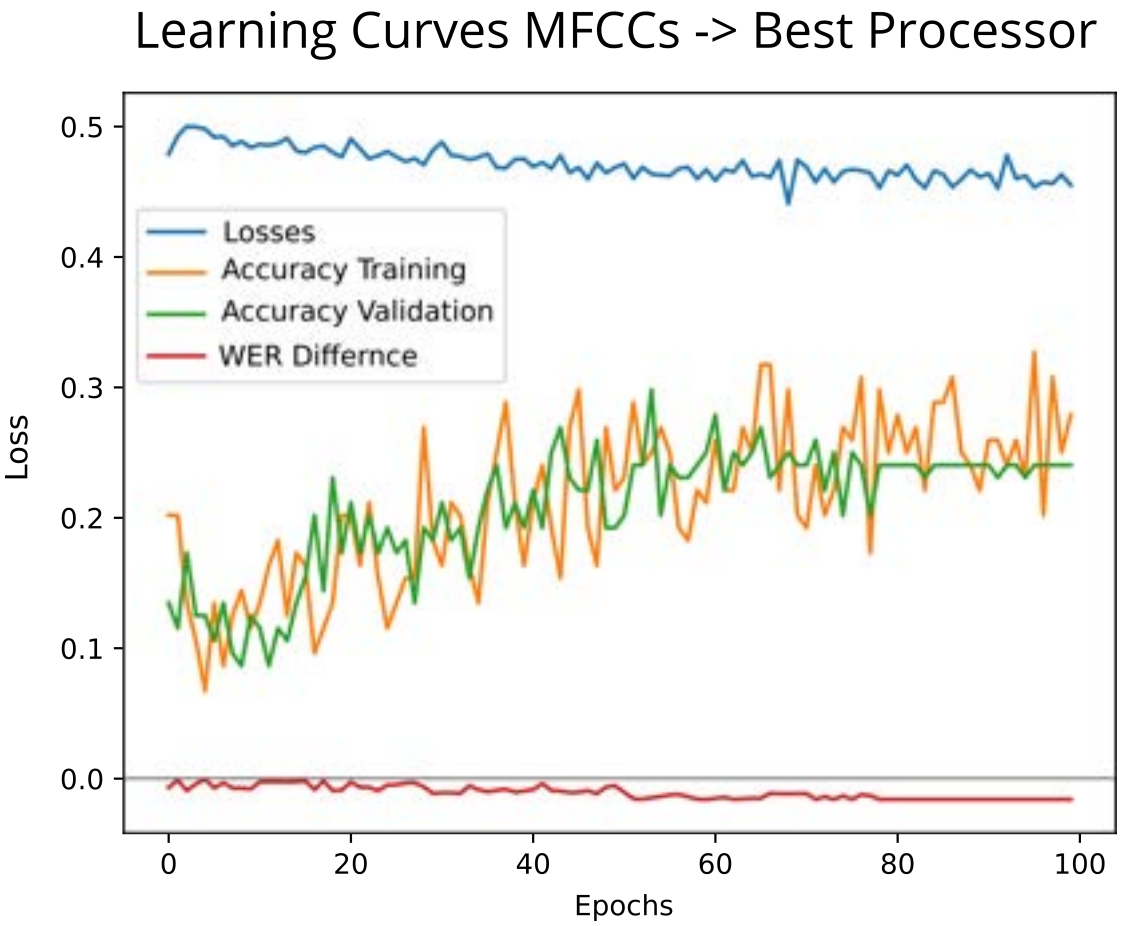

Abbildung 5.13.: Learning Curves mit MFCCs

Wie in Abbildung 5.13 ersichtlich ist, wurde durch das Training erneut nur ein bescheidener Fortschritt erzielt durch das Training. Die Accuracy ist mit 22% nur unwesentlich besser als ein Zufallsentscheid, der $\frac{1}{Anz.Prprozessors} = \frac{1}{7} = 14.2\%$ ist. Zudem sind die relativ starken Schwankungen ein Indikator für eine mangelhafte Vorhersage.

5.9 Vorhersage der Word Error Rate durch CNN

5.9.1 Ziel

Da das Vorhersagen des optimalen Präprozessors bisher nicht gelang, wurde dazu übergegangen, die Word Error Rate einer Audiodatei vorherzusagen bzw. abzuschätzen. Dies geschah wohlweislich, dass die Vorhersage nicht genau genug sein wird, um aus einer Menge von Audiodateien, die auswählen zu können, die das beste Transkript liefern

wird. Es wurde hingegen erwartet, dass sich so eine der besten Audiodateien finden lässt.

5.9.2 Aufbau

Als Vorlage diente der Code des vorherigen Experimentes. Dieser wurde dahingehend angepasst, dass anstelle einer Klassierung eine positive Fliesskommazahl vorhergesagt wird. Des Weiteren wurden nun alle Audiodateien aus dem Rev Korpus herangezogen, die entweder heruntergeladen wurden oder durch das Anwenden von Speech Enhancement entstanden. So wuchs die Anzahl der Audiodateien auf 4008 erhöhte. Zudem wurde beim Training die Länge der Audios von zwei Minuten auf eine Minute gekürzt. Dies geschah in der Erwartung, dass so ein Vielfaches der Daten zum Training bereitsteht und die Ergebnisse möglicherweise genauer werden, wobei das Kürzen der Audiodateien etwas Ressourcen einsparen sollte.

Das Training wurde mit einer Batchsize von 8 und 100 Epochen auf Single-Channel Audiodateien durchgeführt.
Der unten stehende Code zeigt den Aufbau des Neuronalen Netzes.

```
self.conv1 = nn.Conv2d(channels, 243, kernel_size=(3, 3),
   stride=(2, 2), padding=(1, 1))
self.relu1 = nn.ReLU()
self.bn1 = nn.BatchNorm2d(243)
init.kaiming_normal_(self.conv1.weight, a=0.1)
self.conv1.bias.data.zero_()
conv_layers += [self.conv1, self.relu1, self.bn1]

self.conv13 = nn.Conv2d(243, 81, kernel_size=(3, 3), stride
   =(2, 2), padding=(1, 1))
self.relu13 = nn.ReLU()
self.bn13 = nn.BatchNorm2d(81)
init.kaiming_normal_(self.conv13.weight, a=0.1)
self.conv13.bias.data.zero_()
conv_layers += [self.conv13, self.relu13, self.bn13]

self.conv12 = nn.Conv2d(81, 27, kernel_size=(3, 3), stride
   =(2, 2), padding=(1, 1))
self.relu12 = nn.ReLU()
self.bn12 = nn.BatchNorm2d(27)
init.kaiming_normal_(self.conv12.weight, a=0.1)
self.conv12.bias.data.zero_()
conv_layers += [self.conv12, self.relu12, self.bn12]

self.conv14 = nn.Conv2d(27, 9, kernel_size=(3, 3), stride=(2,
```

```
    2), padding=(1, 1))
self.relu14 = nn.ReLU()
self.bn14 = nn.BatchNorm2d(9)
init.kaiming_normal_(self.conv14.weight, a=0.1)
self.conv14.bias.data.zero_()
conv_layers += [self.conv14, self.relu14, self.bn14]

# Linear Classifier
self.ap = nn.AdaptiveAvgPool2d(output_size=1)
self.lin = nn.Linear(in_features=9, out_features=1)

# Wrap the Convolutional Blocks
self.conv = nn.Sequential(*conv_layers)
```

5.9.3 Ergebnisse

Der Verlauf der Learning Curves zeigen, dass diesmal sichtbare Fortschritte während des Trainings erzielt werden konnten. Am Ende des Trainings zeigte sich, dass die WER mit einer mittleren Abweichung von ca. 11% vorhergesagt werden kann. Dennoch ist der Verlauf der Learning Curves etwas unüblich. Es wäre zu erwarten gewesen, dass in den ersten Epochen höhere Abweichungen auftreten und diese innerhalb weniger Epochen stark sinkt und sich schliesslich stabilisiert. Eine potenzielle Erklärung für diese Unregelmässigkeit ist, dass das Neuronale Netz unvorteilhaft designet wurde. Aufgrund eines Denkfehlers wurden die Grössenverhältnisse der Layer des CNN ungeschickt gewählt. Gegebenenfalls liesse sich die Performance verbessern, wenn diese korrigiert würden.

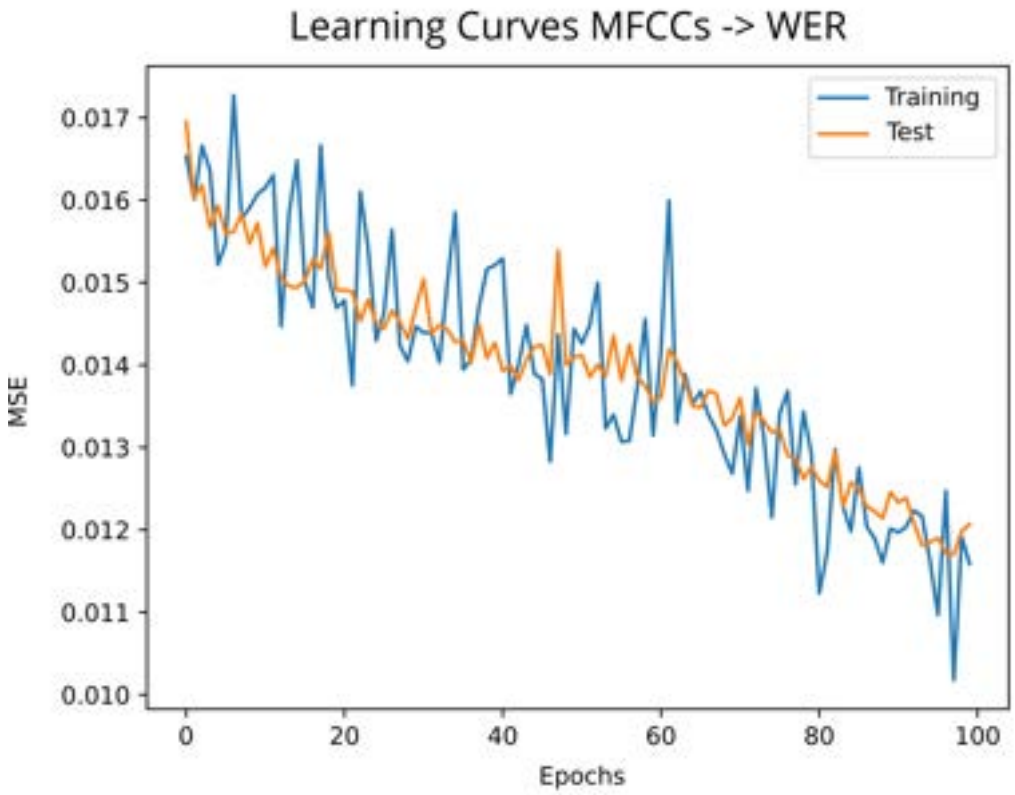

Abbildung 5.14.: Learning Curves mit MFCCs (WER)

Damit besteht nach wie vor eine Unsicherheit hinsichtlich der Verlässlichkeit der Vorhersage. Daher wurde entschieden eine weitere Validierung durchzuführen.

Vorhersage von Dolby.io Verschlechterungen

Um die Einsatzmöglichkeiten eines solchen WER-Vorhersagemodells zu untersuchen, wurden die WER-Schätzungen verwendet, um vorherzusagen, ob eine bestimmte Dolby.io bearbeitete Audioaufnahme, zu einer WER Verschlechterung gegenüber der Originalaufnahme führt. Die WER-Vorhersagen der Baseline ergaben einen MSE von 0,0316 und die Vorhersagen von Dolby.io Audio WERs einen MSE von 0.0337. Folgt man nun den geschätzten WERs und wendet Dolby.io nur an, wenn WER-Predicted$_{\Delta}$ grösser als 0 ist, erhält man einen WER-True$_{\Delta}$ von -0.0316%.

5.9.4 Validierung

Da das Training mit Daten aus dem Rev Korpus durchgeführt wurde, bestand das Risiko, dass das Neuronale Netz Charakteristiken gelernt haben könnte, die lediglich in diesem Korpus auftreten. Um diese Vermutung bestätigen bzw. widerlegen zu können, wurde

eine Validierung mit Teilen des Ceasr Korpus durchgeführt. Der erste Teil entspricht der ”Phase 1” aus dem ”DARPA Switchboard Telephone Conversation Corpus”. Dies umfasst eine Sammlung von englischen Telefongesprächen, die 1997 aufgenommen wurden.

```
...
predicted 0.96  instead of 0.992 diff 0.0322
predicted 0.856 instead of 0.98  diff 0.124
predicted 0.988 instead of 0.983 diff 0.0055
predicted 0.647 instead of 0.948 diff 0.301
predicted 0.999 instead of 0.964 diff 0.0342
predicted 0.992 instead of 0.995 diff 0.00307
predicted 0.772 instead of 0.963 diff 0.191
predicted 0.993 instead of 0.974 diff 0.0195
predicted 0.85  instead of 0.939 diff 0.0891
predicted 0.999 instead of 0.95  diff 0.0485
...
AVG: 0.12370157429078436
MSE: 0.03631070761934362
MED: 0.04465856848950367
```

Vorhersage auf Phase 1 DARPA Switchboard Telephone Conversation Corpus

```
...
predicted 0.526 instead of 0.222 diff 0.303
predicted 0.41  instead of 0.0   diff 0.41
predicted 0.474 instead of 0.0   diff 0.474
predicted 0.487 instead of 0.636 diff 0.149
predicted 0.395 instead of 0.0   diff 0.395
predicted 0.488 instead of 0.294 diff 0.194
predicted 0.471 instead of 0.308 diff 0.164
predicted 0.534 instead of 0.231 diff 0.303
predicted 0.424 instead of 0.0   diff 0.424
predicted 0.48  instead of 0.333 diff 0.147
...
AVG: 0.335509011725293
MSE: 0.141728760687437
MED: 0.37
```

Vorhersage auf Common Voice des Types ”valid” aus der Gruppe ”Test”

Dabei ist deutlich sichtbar, dass die vorhergesagten WER Werte genauer sind, wenn die WER nahe bei Eins ist. Somit kann davon ausgegangen werden, dass das CNN durch Charakteristiken des Rev Korpus ein Bias zu hohen Fehlerraten in den Transkripten hat.

5.9.5 Diskussion

Die wesentlichen gefundenen Schwächen dieses Ansatzes dürften sich bei einem weiteren Versuch eliminieren lassen. Konkret müssten Trainingsdaten verwendet werden, die nicht ausschliesslich hohe WER Werte aufweisen, sondern auch solche, die sich nahezu fehlerfrei transkribieren lassen und die Struktur des Neuronalen Netzes überarbeitet werden.

5.10 Transkriptselektion durch Sentence Scoring

5.10.1 Ziel

Die Vorhersage der Wortfehlerrate ohne Referenztext ist eine schwierige Aufgabe. Ein transkribierter Text hat jedoch die Eigenschaft, dass er von Menschen gesprochen und aufgezeichnet wurde, um Informationen zu vermitteln. Daher liegt es in der Natur von Transkriptionen, dass sie einem Muster der gesprochenen Sprache folgen, z. B. Grammatik und Wortwahl im Kontext. In der Tabelle 5.8 wird ersichtlich, dass sich dies durch die mit einem Language Modell berechnete Wahrscheinlichkeit des Satzes messen lässt.

Text	**WER**	**ØP(W)**	**Output-wdiff**
Referenztext	-	**0.0315**	i believe we ve had a series of very productive meetings in the past few days and i m looking forward to continuing to make progress on critical global issues
Apple	**0.033**	0.0308	i believe we {+ve+} had a series of very productive meetings in the past few days and i m looking forward to continuing to make progress on critical global issues
DeepSpeech	0.10	0.0151	i believe we ve had a series of [-reproductive-] {+very productive+} meetings in the past few days and i m looking forward to continuing to make progress [-in-] {+on+} critical global issues

Tabelle 5.8.: Apples Transkript hat eine tiefere WER und hohe $P(W)$ für die Wahrscheinlichkeit von Wörtern im geometrischen Mittel. Ausserdem liest es sich für Menschen ”logischer” als das von DeepSpeech. Blaue Wörter fehlen im Transkript und rote Wörter sind im Referenztext nicht vorhanden, bzw. folgt auf eine Deletion eine Insertion, so handelt es sich um eine Substitution.

Ziel dieses Experiments ist es, das Transkript mit der niedrigsten Wortfehlerrate unter allen Transkripten anhand der inhaltlichen Wahrscheinlichkeit zu identifizieren. So könnte man eine Audioaufnahme durch alle Präprozessoren laufen lassen, dann aus allen Audiodateien Transkripte erstellen und schliesslich das beste Transkript mit der niedrigsten Wortfehlerrate anhand der Wahrscheinlichkeit des Textes ermitteln.

5.10.2 Aufbau

Sentence Scoring

Ein Language-Model lernt die Wahrscheinlichkeitsverteilung von Wörtern aus einer Menge von Texten [66]. So können Texte mit Modellen wie z.B. GPT2 generiert werden. Aus den bedingten Wahrscheinlichkeiten lassen sich anhand von gegebenen Wörtern nachfolgende Wörter generieren. Dabei ist aber auch das Gegenteil möglich: Gegeben ein Satz mit einer endlichen Anzahl Wörtern, kann die Wahrscheinlichkeit einer Wortfolge ermittelt werden.
Sei W ein Satz bestehend aus $w_1, w_2, \dots, w_n$ Wörtern und $P(w_n)$ die Wahrscheinlichkeit für ein Wort und $P(w_m|w_n)$ die bedingte Wahrscheinlichkeit. Dann lässt sich die Wahrscheinlichkeit für einen Satz mit $P(W)$ wie folgt berechnen.
Beispiel aus [66]:

$$P(W) = P(w_1) \quad *P(w_2|w_1) \quad *P(w_3|P(w_1 \oplus w_2)) \tag{5.1}$$

$$P(\text{"a red fox"}) = P(\text{"a"}) \quad *P(\text{"red"}|\text{"a"}) \quad *P(\text{"fox"}|\text{"a red"}) \tag{5.2}$$

Da sich die Wahrscheinlichkeit der Sätze aus Produkten von Werten kleiner als eins zusammensetzt, sinkt die Wahrscheinlichkeit mit jedem zusätzlichen Wort. Zudem gibt es in den Transkriptionen nicht immer eindeutige Punkt-Satzzeichen, die einen vollständigen Satz kennzeichnen. Daher wurden Texte mit einem "Sliding Windows" der Grösse zehn aufgeteilt. Auf den Fragmenten, wurden dann die Wahrscheinlichkeiten der Wortfolge berechnet und über alle Fragmente gemittelt. Die Berechnung erfolgte mit der Python-Bibliothek "lm-scorer" [67] und dem Language Modell "gpt2-large".

"cats are very cool pets"		**"cats random very arbitrary pets"**	
('cats', 'are', 'very')	1.41e-14	('cats', 'random', 'very')	1.35e-19
('are', 'very', 'cool')	3.53e-13	('random', 'very', 'arbitrary')	1.60e-17
('very', 'cool', 'pets')	4.83e-15	('very', 'arbitrary', 'pets')	7.12e-19
Mean:	**1.24e-13**	**Mean:**	**5.62E-18**

Tabelle 5.9.: Die linke Seite stellt einen korrekten Satz dar. Die rechte Seite repräsentiert einen falschen und "unlogischen" Satz. Jede Zeile zeigt ein Sliding Window (n=3) und der Wahrscheinlichkeit die Wortfolge.

SVM

Die Wahrscheinlichkeit für eine Wortfolge kann Auskunft darüber geben, wie gebräuchlich ein Satz ist. Sie kann jedoch nicht erkennen, ob bei der Transkription viele Löschungen auftraten. In einer ungünstigen Situation geht bei der Transkription genau ein kompletter Satz verloren, ohne dass die inhaltliche Logik beeinträchtigt wird. Um diesem Problem zu begegnen, wird auch die Anzahl der Sliding Windows berücksichtigt, die im Vergleich zu den Daten der anderen Transkriptionen Aufschluss darüber geben kann, ob Inhalte verloren gegangen sind. Die Anzahl der Schiebefenster wurde zusammen mit der Wahrscheinlichkeit für die Texte an eine SVM übergeben.

audio_name	ppl_prod-maxine_dr-dn	split_length-maxine_dr-dn	...	split_length-no_processing
RT6Y8NC7	1.96e-25	66	...	88
ZqmM73iz	3.17e-16	192	...	187
FAngT3Uh	2.08e-21	249	...	258
...	...	...	...	...
n3y6JFYM	1.11e-24	280	...	274
AE4efEKc	6.12e-23	50	...	51
FxTcCkYb	3.54e-21	366	...	362

Tabelle 5.10.: Ausschnitt vom Dataframe für die SVM

Wie bei den vorherigen SVM-Experimenten wurde das Korpus in Trainings- und Testdaten aufgeteilt und die optimalen Parameter mit GridSearchCV auf einem StratifiedKFold gefunden. Die Merkmale wurden ebenfalls wieder mit dem StandardScaler standardisiert.

5.10.3 Ergebnisse

Bei 406 Audioclips wurden die Transkripte in Sliding Windows zerlegt und die durchschnittliche Wahrscheinlichkeit berechnet. Die bestmögliche Verbesserung auf diesem Datensatz liegt bei 9.84% WER-Reduktion. Wählt man nun immer das Transkript (mit Präprozessor oder unbearbeitet) mit der höchsten Wahrscheinlichkeit aus, erhält man ein WER_{Δ} von 0.6%. Berücksichtigt man aber nun noch die Anzahl Sliding Windows und über übergibt alle Werte einer SVM, erzielt man eine WER Reduktion von 7.01%.

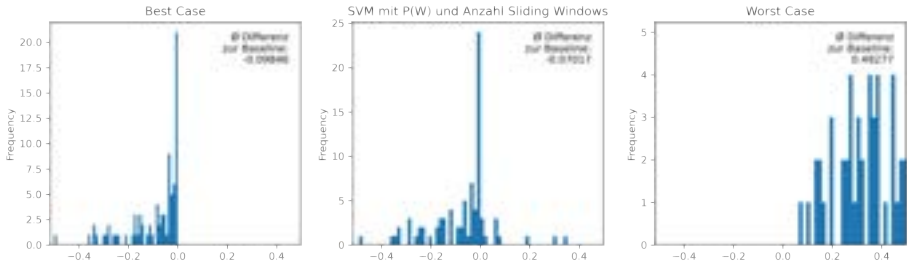

Abbildung 5.15.: Die Verteilung der WER Deltas im Best-Case, mit der SVM Vorhersagen und Worst-Case.

In der Grafik 5.15 ist ersichtlich, dass Transkripte, welche schlechter als die Baseline sind, sehr gut erkannt werden.

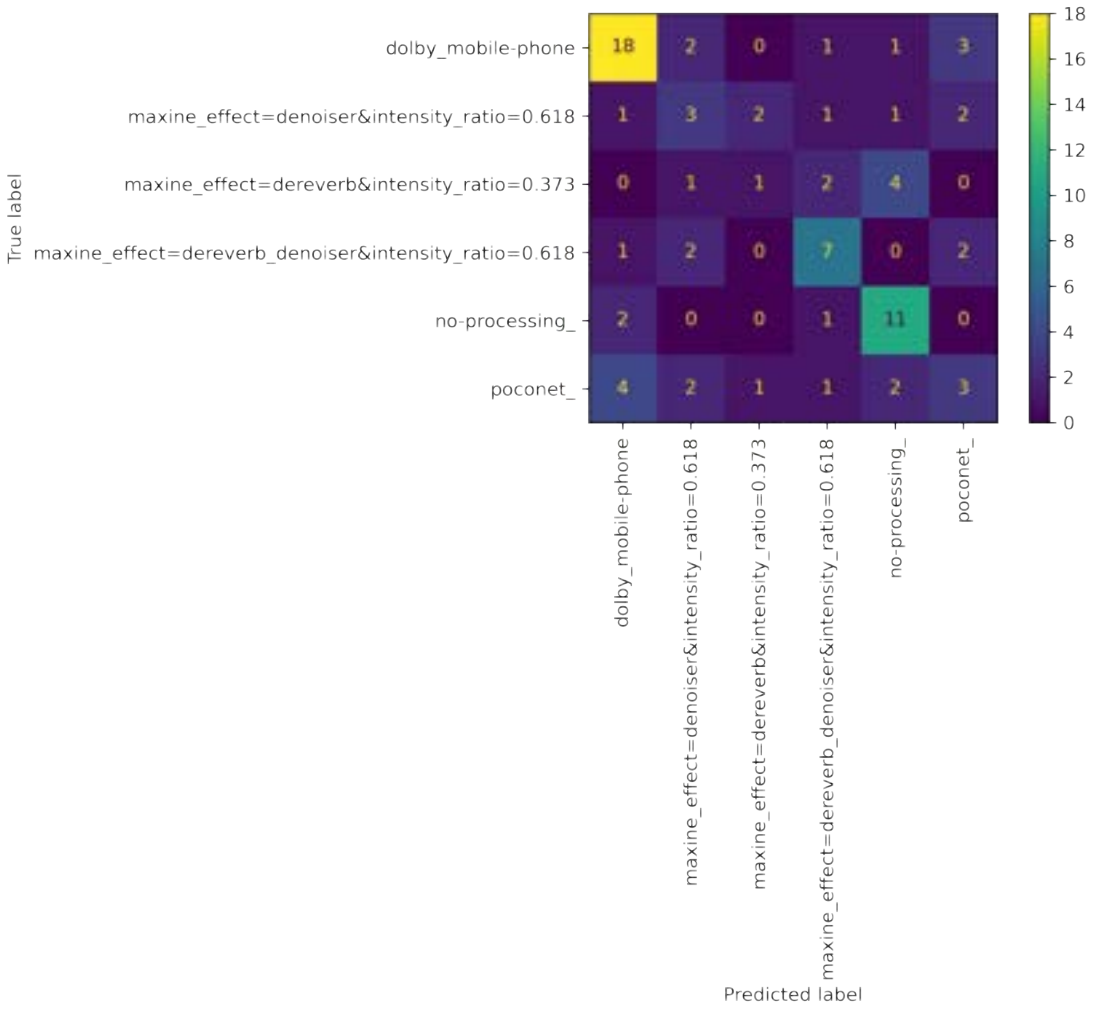

Abbildung 5.16.: Confusion Matrix mit den gewählten Präprozessoren vs. bester Präprozessoren

Die Confusion Matrix 5.16 weist eine ausgeprägtere Diagonale auf als die SVMs im Kapitel 5.6.

5.10.4 Diskussion

Das Sentence Scoring ist eine bewährte Methode, um ASR-Hypothesen zu vergleichen und die Hypothese mit den wenigsten Fehlern zu identifizieren. Dies widerspiegelt sich auch in unserem letzten Experiment, das die beste WER-Reduktion liefert. Da für alle Präprozessoren Transkripte zur Verfügung stehen, kann die Qualität der Transkripte viel zuverlässiger bestimmt werden. Wie in der Confusion Matrix 5.16 und im Histogramm 5.15 zu sehen ist, ist es nicht immer notwendig, den besten Präprozessor zu bestimmen, da es auch ausreicht, Verschlechterungen zu vermeiden und vielleicht einfach den

zweitbesten Präprozessor zu wählen.
Allerdings bringt diese Methode auch Nachteile mit sich. Die Berechnung der Wahrscheinlichkeiten mit einem Language Modell setzt voraus, dass man alle Audioaufnahmen mit den Speech Enhancement Verfahren bearbeitet und transkribiert. Darüber hinaus muss in einem zweiten Schritt das Scoring für alle Transkripte mit einem Language Modell wie GPT2 berechnet werden. Dies erfordert grosse Mengen an Rechenressourcen.
Auch ist das Ergebnis dieser Methode nicht auf andere Speech-to-Text-Systeme übertragbar, da in diesem Experiment nur DeepSpeech-Transkripte verwendet wurden. Es ist möglich, dass Unternehmen wie Google oder Microsoft bereits Verfahren wie das ASR-Rescoring implementiert haben. Zukünftige Arbeiten sollten das Sentence Scoring mit anderen Speech-to-Text-Systemen untersuchen. Darüber hinaus werden Sprachmodelle in der Regel trainiert, um die Wahrscheinlichkeit korrekter Wortfolgen zu maximieren, und nicht, um ASR-Fehler zu erkennen. Es existieren weitere Arbeiten zu diesem Thema mit angepassten Methoden zur Erkennung von ASR-Fehlern [68], die sicherlich ebenfalls untersucht werden sollten.

6 Diskussion und Ausblick

Speech Enhancement Verfahren als Preprocessing für automatische Spracherkennung, wurde in dieser Arbeit umfassend untersucht. Mit dem Experiment im Kapitel 5.1 wurde ersichtlich, dass verschiedene Audioaufnahmen, Präprozessoren und STT Engines sich verschieden verhalten und dass es zu WER Verschlechterungen aber auch Verbesserungen kommen kann. Eine Ausnahme war Noisereduce, denn das Signaltechnikbasierende Speech Enhancement Verfahren Noisereduce konnte nie zu einer Verbesserung der WER beitragen. Dies kann möglicherweise darauf zurückgeführt werden, dass für die ASR-Systeme durch Noisereduce die Inkongruenz-Faktoren gegenüber den Trainingsdaten zu gross wird. Im Kapitel 5.5 wurde die effektive WER der präprozessierten Audiodateien mit dem Rev Korpus quantifiziert. Der Rev Korpus entspricht den im Buch «Spoken language processing: a guide to theory, algorithm, and system development» erwähnten Kriterien, für eine Aussagekräftige Auswertung der WERs [42]. Der Rev-Korpus erforderte jedoch das automatische Kürzen der Referenztexte und stellt eine Einschränkung unserer Arbeit dar. Da nach einem Ankerpunkt im Referenztext, die Anzahl der fehlenden Wörter im Transkript, ergänzt wird, kann keine hundertprozentige Garantie gegeben werden, dass der Referenztext an der richtigen Stelle gekürzt wurde. Allerdings wird der Referenztext einer Audiodatei für alle Präprozessor-WER-Berechnungen verwendet, so dass sie alle die gleichen Voraussetzungen haben und somit trotzdem ein aussagekräftiger Vergleich möglich ist. Dabei zeigte sich, dass kein allgemeines bestes Speech Enhancement Verfahren existiert. Wird Dolby.io mit dem Preset «mobile-phone» immer angewendet, kommt es zu einem WER-Delta von 0.8%. Wird der schlechteste Präprozessor immer angewendet, erreicht man mit Deep Xi ein WER-Delta von 6.4%, was in Widerspruch mit dem Paper von Nicolson et. al steht [18]. Im Paper wurde ein WER-Delta von über 20% erreichen. Eine mögliche Erklärung für diese unterschiedlichen Ergebnisse könnte der Audiokorpus sein, denn in dieser Arbeit wurden echte Aufnahmen mit Hintergrundgeräuschen verwendet. In der Arbeit von Nicolson et. al. wurden die Störgeräusche künstlich erzeugt und zum Training und zur Inferenz verwendet. Auf dem ganzen Rev-Korpus wurde im Best-Case Szenario ein WER-Delta von -7.1% erzielt. Dieses Szenario erfordert aber, dass immer der beste Präprozessor bekannt ist. So wurde in der zweiten Hälfte dieser Arbeit, der Fokus auf die Vorhersage des besten Speech Enhancement Verfahrens gesetzt, um den Best-Case von -7.1% anzunähern.
Mit der Python-Bibliothek librosa und der Dolby.io Analyze API wurden Metriken für

alle Audioclips berechnet, um diese als Features in supervised Machine-Learning Modellen zu verwenden und trainieren. Die SVMs im Kapitel 5.6 erzielten zwar bessere WER-Deltas gegenüber dem Dummy-Classifier, bringen aber mit den Vorhersagen insignifikante WER-Verbesserungen von rund -1.5%. Auch das DNN aus Kapitel 5.7 zeigte, dass sich mit den Dolby.io und Librosa Features, nicht die besten Präprozessoren lernen lässt. Die Ergebnisse lassen vermuten, dass es keine Zusammenhänge zwischen den Dolby.io / librosa Features und der Transkriptionsqualität gibt. Abbildung 5.10 zeigt einige Features als Pairplot, dabei sind keine Muster zu erkennen, die die Klassen separieren könnten. So wurden in den nächsten Experimenten neue Ansätze gestartet, um die Audioaufnahmen besser zu charakterisieren.
MFCCs geben Auskunft über «was gesagt wurde» anstelle von «in welcher Tonlage» und wird erfolgreich in verschiedene Soundklassifizierungsaufgaben angewendet. Im Kontext unsere Bachelorarbeit konnte aber eine MFCC basierende Präprozessor-Klassifizierer nicht die Ergebnisse der vorherigen Modelle übertreffen, sondern erreichte vergleichbare WER-Deltas. Wurde jedoch eine MFCC basierende CNN darauf trainiert, die WERs zu vorhersagen, konnte diese verwendet werden, um vorauszusagen, ob auf einem gegebenen Audio Dolby.io Enhancer angewendet werden sollte oder nicht bzw. eine Verbesserung bringt oder nicht. Dies brachte ein WER-Delta von -3.1% auf dem ganzen Rev-Korpus. Der Validierungsschritt zeigte aber, dass sich dieses Ergebnis nicht auf die anderen Korpora übertragen lässt, da die WER Vorhersage auf Teilen des Ceasr Korpus weniger zuverlässig funktionierten. Die -3.1% WER Verbesserung deuten aber daraufhin, dass eine WER Vorhersage durchaus ein möglicher Ansatz ist, um den Präprozessor zu bestimmen. Eine Folgearbeit könnte, die WER Vorhersage mit mehr Daten optimieren.
Das beste Ergebnis erzielten wir mit einem Sentence Scoring Ansatz. Angenommen, eine Audioaufnahme wurde durch alle Speech Enhancement Verfahren bearbeitet und alle resultierenden Audiodateien wurden transkribiert, dann lässt sich das beste Transkript mit Sentence Scoring und einer SVM ermitteln. Auf dem verwendeten Testdatensatz liegt das Best-Case WER-Delta bei -9.8%, mit dem Sentence Scoring und der SVM erreichten wir ein WER-Delta von -7.01%. DeepSpeech erzeugt gelegentlich für den Menschen offensichtliche Fehler. Mit einem Language Modell wie GPT2 lässt sich durch die Wahrscheinlichkeit eines Textes ermitteln, wie üblich ein Transkript ist. Zusammen mit der Anzahl Sliding Windows, was indirekt die Textlänge repräsentiert, können zuverlässig gute und schlechte Transkripte unterschieden werden. Das Experiment 5.10 bietet Erkenntnisse für mögliche Folgearbeiten, die sich mit ASR Rescoring und Ensemble Learning befassen. So existieren z.B. bereits Language Modelle, die sich bereits auf ASR Korrekturen spezialisiert haben. Oder man könnte Audioaufnahmen von zwei verschieden STT Engines vergleichen und an den Stellen, wo sich die STT Engines nicht einig sind, mit einem Language Modell die Lücke, durch die Wahrscheinlichkeit der Wortfolge, schliessen. Das Sentence Scoring bringt im Zusammenhang mit ASR auch Nachteile. Das Berechnen der Wahrscheinlichkeiten ist an hohe Computerressourcenverbrauch gebunden, da eine Aufnahme von allen Präprozessoren bearbeitet und dann alle resultierenden Audiodateien transkribiert und gescored werden müssen. Auch gilt das Resultat von unserem Experiment nur für DeepSpeech, wie im Experiment 5.1 ersichtlich

war, verhalten sich verschieden ASR Systeme im Zusammenhang mit Präprozessoren verschieden. Es wird empfohlen das Sentence Scoring auch mit anderen ASR System zu untersuchen.

7 Verzeichnisse

Literaturverzeichnis

[1] „Speech-to-Text Benchmark", Mai 2022. [Online]. URL: https://github.com/Picovoice/speech-to-text-benchmark [Stand: 04-06-2022].

[2] M. Delcroix, Y. Kubo, T. Nakatani und A. Nakamura, „Is speech enhancement pre-processing still relevant when using deep neural networks for acoustic modeling?" 2013.

[3] R. Errattahi, A. E. Hannani und H. Ouahmane, „Automatic Speech Recognition Errors Detection and Correction: A Review", *Procedia Computer Science*, Bd. 128, S. 32–37, Jan. 2018, Publisher: Elsevier.

[4] T. Virtanen, R. Singh und B. Raj, *Techniques for noise robustness in automatic speech recognition*, 2012. [Online]. URL: https://books.google.ch/books?hl=en&lr=&id=EjMxpiT38owC&oi=fnd&pg=PT14&dq=Technique+for+noise+robustness+in+automatic+speech+recognition&ots=ibj-t48Cm2&sig=g8O-QFPBzNbDWuoQEzkCgR__SJA

[5] J. Benesty, S. Makino und J. Chen, *Speech Enhancement*. Springer, 2005.

[6] T. Sainburg, M. Thielk und T. Q. Gentner, „Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires", *PLOS Computational Biology*, Bd. 16, Nr. 10, S. e1008228, Okt. 2020. [Online]. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008228 Publisher: Public Library of Science ISBN: 1111111111.

[7] A. W. Rix, J. G. Beerends, M. P. Hollier und A. P. Hekstra, „Perceptual evaluation of speech quality (PESQ) - A new method for speech quality assessment of telephone networks and codecs", *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Bd. 2, S. 749–752, 2001.

[8] J. G. Beerends, M. Obermann, R. Ullmann, J. Pomy und M. Keyhl, „Perceptual Objective Listening Quality Assessment (POLQA), The Third Generation ITU-T Standard for End-to-End Speech Quality Measurement Part I–Temporal Alignment", *J. Audio Eng. Soc.*, Bd. 61, Nr. 6, S. 19, 2013.

[9] B. Naderi und R. Cutler, „An Open source Implementation of ITU-T Recommendation P.808 with Validation", *Interspeech 2020*, S. 2862–2866, Okt. 2020. [Online]. URL: http://arxiv.org/abs/2005.08138 arXiv: 2005.08138.

[10] T. S. S. of ITU, „SERIES P: TERMINALS AND SUBJECTIVE AND OBJECTIVE ASSESSMENT METHODS", Juli 2016. [Online]. URL: https://www.itu.int/rec/T-REC-P.800.1-201607-I/en [Stand: 02-05-2022].

[11] C. K. Reddy, H. Dubey, K. Koishida, A. Nair, V. Gopal, R. Cutler, S. Braun, H. Gamper, R. Aichner und S. Srinivasan, „Interspeech 2021 Deep Noise Suppression Challenge", *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, Bd. 2, S. 801–805, Jan. 2021. [Online]. URL: https://arxiv.org/abs/2101.01902v3 arXiv: 2101.01902 Publisher: International Speech Communication Association ISBN: 9781713836902.

[12] C. K. Reddy, V. Gopal, R. Cutler, E. Beyrami, R. Cheng, H. Dubey, S. Matusevych, R. Aichner, A. Aazami, S. Braun, P. Rana, S. Srinivasan und J. Gehrke, „The INTERSPEECH 2020 Deep Noise Suppression Challenge: Datasets, Subjective Testing Framework, and Challenge Results", *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, Bd. 2020-October, S. 2492–2496, Mai 2020. [Online]. URL: https://arxiv.org/abs/2005.13981v3 arXiv: 2005.13981 Publisher: International Speech Communication Association.

[13] T. Nakatani, S. Araki, M. Delcroix, T. Yoshioka und M. Fujimoto, „Reduction of highly nonstationary ambient noise by integrating spectral and locational characteristics of speech and noise for robust ASR", in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[14] J. Barker, E. Vincent, N. Ma, H. Christensen und P. Green, „The PASCAL CHiME Speech Separation and Recognition Challenge", *Computer Speech and Language*, Bd. 27, Nr. 3, 2013. [Online]. URL: https://hal.inria.fr/hal-00743529 Publisher: Elsevier.

[15] „The 4th CHiME Speech Separation and Recognition Challenge". [Online]. URL: http://spandh.dcs.shef.ac.uk/chime_challenge/CHiME4/results.html [Stand: 09-05-2022].

[16] nvidia, „What Is NVIDIA Maxine?". [Online]. URL: https://developer.nvidia.com/maxine#ae-sdk [Stand: 11-05-2022].

[17] U. Isik, R. Giri, N. Phansalkar, J.-M. Valin, K. Helwani und A. Krishnaswamy, „PoCoNet: Better Speech Enhancement with Frequency-Positional Embeddings, Semi-Supervised Conversational Data, and Biased Loss", *arXiv:2008.04470 [cs, eess, stat]*, Aug. 2020. [Online]. URL: http://arxiv.org/abs/2008.04470 arXiv: 2008.04470.

[18] A. Nicolson und K. K. Paliwal, „Deep Xi as a Front-End for Robust Automatic Speech Recognition", in *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, Dez. 2020, S. 1–6.

[19] T. Sainburg, „timsainb/noisereduce: v1.0". [Online]. URL: https://github.com/timsainb/noisereduce [Stand: 18-03-2022].

[20] „Enhance | Media APIs | Dolby.io". [Online]. URL: https://docs.dolby.io/media-apis/docs/enhance-api-guide [Stand: 01-05-2022].

[21] nvidia, „Audio Effects SDK (Linux)". [Online]. URL: https://docs.nvidia.com/deeplearning/maxine/audio-effects-sdk-linux/index.html [Stand: 18-03-2022].

[22] nvidia, „Introduction to the NVIDIA Audio Effects SDK". [Online]. URL: https://docs.nvidia.com/deeplearning/maxine/audio-effects-sdk/index.html#afx-intro [Stand: 04-06-2022].

[23] „How-To Improve Audio Quality by Content Type | Media APIs | Dolby.io". [Online]. URL: https://docs.dolby.io/media-apis/docs/how-to-improve-audio-by-content-type [Stand: 02-05-2022].

[24] „Start Enhancing". [Online]. URL: https://docs.dolby.io/media-apis/reference/media-enhance-post

[25] „OpenVINO™ Toolkit - Open Model Zoo repository", Mai 2022. [Online]. URL: https://github.com/openvinotoolkit/open_model_zoo/blob/e313674d35050d2a4721bbccd9bd4c404f1ba7f8/models/public/index.md [Stand: 02-05-2022].

[26] A. Nicolson, „Deep Xi: A Deep Learning Approach to A Priori SNR Estimation for speech enhancement." Mai 2022. [Online]. URL: https://github.com/anicolson/DeepXi original-date: 2018-05-25T01:30:27Z.

[27] „Speechmatics | Homepage". [Online]. URL: https://www.speechmatics.com/index [Stand: 04-06-2022].

[28] „Dragon Speech Recognition - Get More Done by Voice | Nuance". [Online]. URL: https://www.nuance.com/dragon.html [Stand: 04-06-2022].

[29] „Amazon Transcribe – Sprache-zu-Text – AWS". [Online]. URL: https://aws.amazon.com/de/transcribe/ [Stand: 04-06-2022].

[30] Google, „Speech-to-Text". [Online]. URL: https://cloud.google.com/speech-to-text [Stand: 05-05-2022].

[31] Microsoft, „Speech to text". [Online]. URL: https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/#overview [Stand: 05-05-2022].

[32] „Speech | Apple Developer Documentation". [Online]. URL: https://developer.apple.com/documentation/speech

[33] „Project DeepSpeech", Juni 2022. [Online]. URL: https://github.com/mozilla/DeepSpeech [Stand: 04-06-2022].

[34] „wav2letter++", Juni 2022. [Online]. URL: https://github.com/flashlight/wav2letter [Stand: 04-06-2022].

[35] „TensorFlowASR", Juni 2022. [Online]. URL: https://github.com/TensorSpeech/TensorFlowASR [Stand: 04-06-2022].

[36] Gartner, „Market Guide for Speech-to-Text Solutions". [Online]. URL: https://www.gartner.com/document/3983881 [Stand: 04-06-2022].

[37] Google, „Best practices | Cloud Speech-to-Text Documentation". [Online]. URL: https://cloud.google.com/speech-to-text/docs/best-practices [Stand: 18-03-2022].

[38] Mozilla, „Project DeepSpeech". [Online]. URL: https://github.com/mozilla/DeepSpeech [Stand: 05-05-2022].

[39] Mozilla, „Welcome to DeepSpeech's documentation!". [Online]. URL: https://deepspeech.readthedocs.io/en/ro.9/?badge=latest [Stand: 05-05-2022].

[40] S. Thordarson, „hear", Mai 2022. [Online]. URL: https://github.com/sveinbjornt/hear [Stand: 14-05-2022].

[41] A. C. Morris, V. Maier und P. Green, „From WER and RIL to MER and WIL: Improved evaluation measures for connected speech recognition", in *8th International Conference on Spoken Language Processing, ICSLP 2004*. International Speech Communication Association, 2004, S. 2765–2768.

[42] X. Huang, A. Acero und H.-W. Hon, *Spoken language processing: a guide to theory, algorithm, and system development*. Upper Saddle River, NJ: Prentice Hall PTR, 2001.

[43] „JiWER: Similarity measures for automatic speech recognition evaluation", Juni 2022. [Online]. URL: https://github.com/jitsi/jiwer [Stand: 04-06-2022].

[44] „Signal to Noise Ratio". [Online]. URL: https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/snr.htm [Stand: 04-06-2022].

[45] S. Morita, X. Lu und M. Unoki, „Signal to noise ratio estimation based on an optimal design of subband voice activity detection", in *The 9th International Symposium on Chinese Spoken Language Processing*, Sep. 2014, S. 560–564.

[46] M. Benzeghiba, R. D. Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris, R. Rose, V. Tyagi und C. Wellekens, „Automatic speech recognition and speech variability: A review". [Online]. URL: https://www.sciencedirect.com/science/article/pii/S0167639307000404 [Stand: 17-05-2022].

[47] M. Sahidullah und G. Saha, „Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition". [Online].

URL: https://www.sciencedirect.com/science/article/pii/S0167639311001622 [Stand: 17-05-2022].

[48] P. Pedersen, „The Mel Scale". [Online]. URL: https://www.jstor.org/stable/843164 [Stand: 17-05-2022].

[49] srf, „Heute, 14. Mai 2022". [Online]. URL: https://www.srf.ch/play/tv/programm [Stand: 14-05-2022].

[50] E. Parton, „All Transcripts", Jan. 2019. [Online]. URL: https://www.rev.com/blog/transcripts/all-transcripts [Stand: 15-05-2022].

[51] M. A. Ulasik, M. Hürlimann, F. Germann, E. Gedik, F. Benites und M. Cieliebak, „CEASR: a corpus for evaluating automatic speech recognition", in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, S. 6477–6485.

[52] J. W. Hunt und M. D. McIlroy, „An Algorithm for Differential File Comparison", S. 9.

[53] „Measure Noise | Media APIs | Dolby.io". [Online]. URL: https://docs.dolby.io/media-apis/docs/how-to-measure-noise

[54] „Analyze | Media APIs | Dolby.io". [Online]. URL: https://docs.dolby.io/media-apis/docs/analyze-api-guide [Stand: 20-05-2022].

[55] „Notebooks". [Online]. URL: https://notebooks.githubusercontent.com/view/ipynb?browser=chrome&color_mode=auto&commit=9f637b48fc50ff6e76483c21cd64fae9d0b3f65d&device=unknown&enc_url=68747470733a2f2f7261772e676974687562757365726e74656e742e636f6d2f6a6b6f7472612f4d757
logged_in=false&nwo=jkotra%2FMusicGenreClassification&path=MusicGenreClassification_FeatureEnsemble.ipynb&platform=android&repository_id=231607935&repository_type=Repository&version=99 [Stand: 20-05-2022].

[56] P. Pandey, „Music Genre Classification with Python", Mai 2021. [Online]. URL: https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8 [Stand: 20-05-2022].

[57] A. Liutkus, „speechmetrics", Mai 2022. [Online]. URL: https://github.com/aliutkus/speechmetrics [Stand: 20-05-2022].

[58] C.-C. Lo, S.-W. Fu, W.-C. Huang, X. Wang, J. Yamagishi, Y. Tsao und H.-M. Wang, „MOSNet: Deep Learning based Objective Assessment for Voice Conversion", *Interspeech 2019*, S. 1541–1545, Sep. 2019. [Online]. URL: http://arxiv.org/abs/1904.08352 arXiv: 1904.08352.

[59] T. H. Falk, C. Zheng und W.-Y. Chan, „A Non-Intrusive Quality and Intelligibility Measure of Reverberant and Dereverberated Speech", *IEEE Transactions on Audio, Speech, and Language Processing*, Bd. 18, Nr. 7, S. 1766–1774, Sep. 2010, Conference Name: IEEE Transactions on Audio, Speech, and Language Processing.

[60] „sklearn.dummy.DummyClassifier". [Online]. URL: https://scikit-learn/stable/modules/generated/sklearn.dummy.DummyClassifier.html [Stand: 21-05-2022].

[61] „sklearn.preprocessing.StandardScaler — scikit-learn 1.1.1 documentation". [Online]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html [Stand: 22-05-2022].

[62] S. Ronaghan, „The Mathematics of Decision Trees, Random Forest and Feature Importance in Scikit-learn and Spark", Nov. 2019. [Online]. URL: https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark [Stand: 22-05-2022].

[63] Mauro Di Pietro, „Deep Learning with Python: Neural Networks (complete tutorial)". [Online]. URL: https://towardsdatascience.com/deep-learning-with-python-neural-networks-complete-tutorial-6b53c0b06af0 [Stand: 02-06-2022].

[64] Ketan Doshi, „Audio Deep Learning Made Simple: Sound Classification, Step-by-Step". [Online]. URL: https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5 [Stand: 18-05-2022].

[65] Daniel S. Park and William Chan and Yu Zhang and Chung-Cheng Chiu and Barret Zoph and Ekin D. Cubuk and Quoc V. Le, „SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". [Online]. URL: https://arxiv.org/pdf/1904.08779.pdf [Stand: 19-05-2022].

[66] F. Chiusano, „Two minutes NLP — Perplexity explained with simple probabilities", Jan. 2022. [Online]. URL: https://medium.com/nlplanet/two-minutes-nlp-perplexity-explained-with-simple-probabilities-6cdc46884584 [Stand: 25-05-2022].

[67] „Issues · simonepri/lm-scorer". [Online]. URL: https://github.com/simonepri/lm-scorer [Stand: 27-05-2022].

[68] H. Futami, H. Inaguma, M. Mimura, S. Sakai und T. Kawahara, „ASR Rescoring and Confidence Estimation with ELECTRA", arXiv, Tech. Rep. arXiv:2110.01857, Okt. 2021, arXiv:2110.01857 [cs, eess] type: article. [Online]. URL: http://arxiv.org/abs/2110.01857

Glossar

API Application Programming Interface.

ASR Automatic Speech Recognition.

Automatic Speech Recognition, Speech-to-Text Die automatische Spracherkennung ist eine Technik, die ein Sprachsignal nutzt, um um Wörter oder Phrasen zu erkennen. .

CNN Convolutional Neural Network.

DNN Deep Neural Network.

GMM Gaussian Mixture Model.

GPT2 GPT2 ist ein Deep Learning-Modell für die Sprachgenerierung. .

GPU Graphics Processing Unit.

HDD Hard Disk Drive.

HTTP Hypertext Transfer Protocol.

Interspeech Conference Die Interspeech ist eine Konferenz, mit dem Schwerpunkt auf Wissenschaft und Technologie zur Verarbeitung gesprochener Sprache. .

Librosa Librosa ist ein Python-Paket zur Bearbeitung von Audio-Dateien. .

Machine Learning Machine Learning ist das Fachgebiet der Algorithmen, die verwendet werden können, um automatisch aus Daten zu lernen. .

MFCC Mel Frequency Cepstral Coefficients.

MOS Mean Opinion Score.

NAS Network Attached Storage.

Neural Network Ein Computersystem nach dem Vorbild des menschlichen Gehirns und Nervensystems, um eine Funktion zu lernen. .

PESQ Perceptual Evaluation of Speech Quality.

POLQA Perceptual Objective Listening Quality Analysis.

Präprozessor, Prozessor, Preprocessor, Processor Im Kontext unserer Arbeit werden diese Begriffe als Speech Enhancement Verfahren verstanden, die vor der Transkription eines Sprachsignals eine Vorbereitung durchführen. .

PyTorch PyTorch ist ein Deep Learning Python-Framework. .

REST API REST API ist eine Schnittstelle zwischen einem Programm und einem Web-Server. .

SDK Software Development Kit.

SSD Solid State Drive.

STT Speech-to-Text.

SWAP Swap Space.

TensorFlow TensorFlow ist ein Deep Learning Python-Framework. .

USB Universal Serial Bus.

WER Word Error Rate.

ZHAW Zürcher Hochschule für Angewandte Wissenschaften.

Abbildungsverzeichnis

Tabellenverzeichnis

A Anhang

A.1 Aufgabenstellung

School of Engineering

[Online Administration Praktische Arbeiten]

[Dept. T Admin Tools]

zurück Logout

Bachelorarbeit 2022 - FS: BA22_ciel_08

Allgemeines:

Titel:	Elimination of Noise in Audio-Aufnahmen
Anzahl Studierende:	2
Durchführung in Englisch möglich:	Ja, die Arbeit kann vollständig in Englisch durchgeführt werden und ist auch für Incomings geeignet.

Betreuer:

HauptbetreuerIn: Mark Cieliebak, ciel

Zugeteilte Studenten:

Diese Arbeit ist vereinbart mit:
- Manuel Berweger, berweman (IT)
- Marvin Tseng, tsengmar (IT)

Fachgebiet:

DSV	Digitale Signalverarbeitung
ML	Machine Learning

Studiengänge:

ET	Elektrotechnik
IT	Informatik

Zuordnung der Arbeit :

CAI	Centre for Artificial Intelligence

Infrastruktur:

benötigt keinen zugeteilten Arbeitsplatz an der ZHAW

Interne Partner :

Es wurde kein interner Partner definiert!

Industriepartner:

Es wurden keine Industriepartner definiert!

Beschreibung:

Bei Speech-to-Text (STT) wird eine Audioaufnahme von einem Gespräch automatisch in Text transkribiert. Dieser Prozess funktioniert sehr gut, wenn die Audioaufnahmen eine hohe Qualität aufweisen.

Sobald das nicht der Fall ist, zB. bei schlechter Raumakkustik, Hintergrundgeräuschen oder weil eine Person zu weit vom Micro entfernt ist, hat der generierte Text sehr viele Fehler.

Das Ziel dieser Arbeit ist, potentielle Störungen im Audiosignal zu erkennen und diese, soweit möglich, zu eleminieren.

Hier ein paar potentielle Ansätze:

- Bereits bei der Aufnahme die Qualität prüfen und Vorschläge unterbreiten wie das Setting (Richtung vom Microfon, Position der Sprecher, Lautstärke) angepasst werden kann
- Hintergrundgeräusche als künstlichen "Sprecher" betrachten und dann mit Speech Separation diesen Sprecher herausfiltern (Speech Separation wird normalerweise verwendet, um parallele Sprecher zu separieren).
- Die Lautstärke von einzelnen Sprechern automatisch angleichen (zB falls diese unterschiedlich weit vom Micro entfernt waren)

Je nach Interesse und Vorwissen kann der Focus dieser Arbeit im Bereich Signalverarbeitung oder Machine Learning liegen - oder sogar in beiden.

Voraussetzungen:

Falls Sie Interesse an diesem spannenden Thema haben, können wir gern einen Termin abmachen und die konkrete Aufgabenstellung besprechen. Email: ciel@zhaw.ch oder Telefon: 058 934 72 39.

zurück Logout

A.2 SVM Confusion Matrix

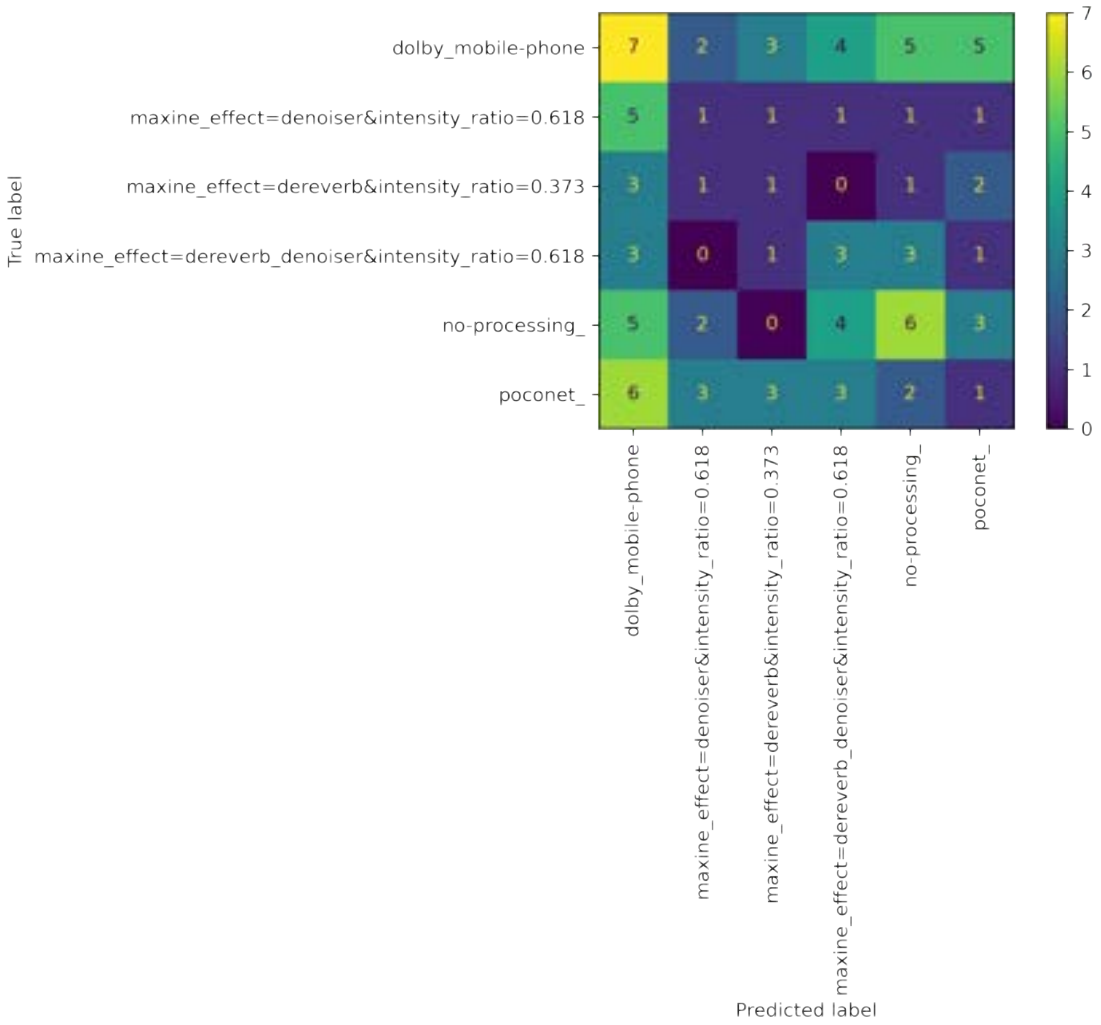

Abbildung A.1.: Confusion Matrix vom Dummy Classifier

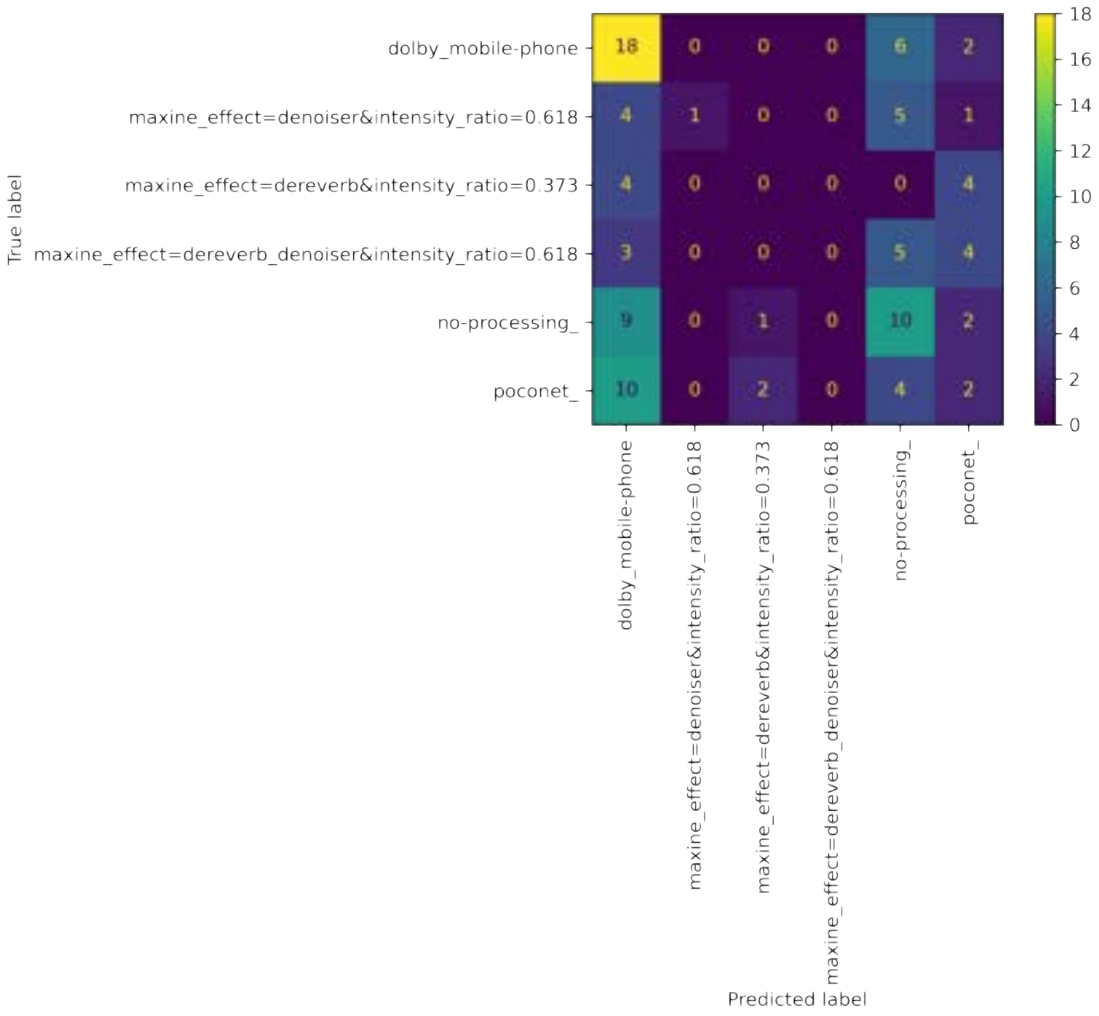

Abbildung A.2.: Confusion Matrix vom SVM_BestProc+NoProc_AllFeat

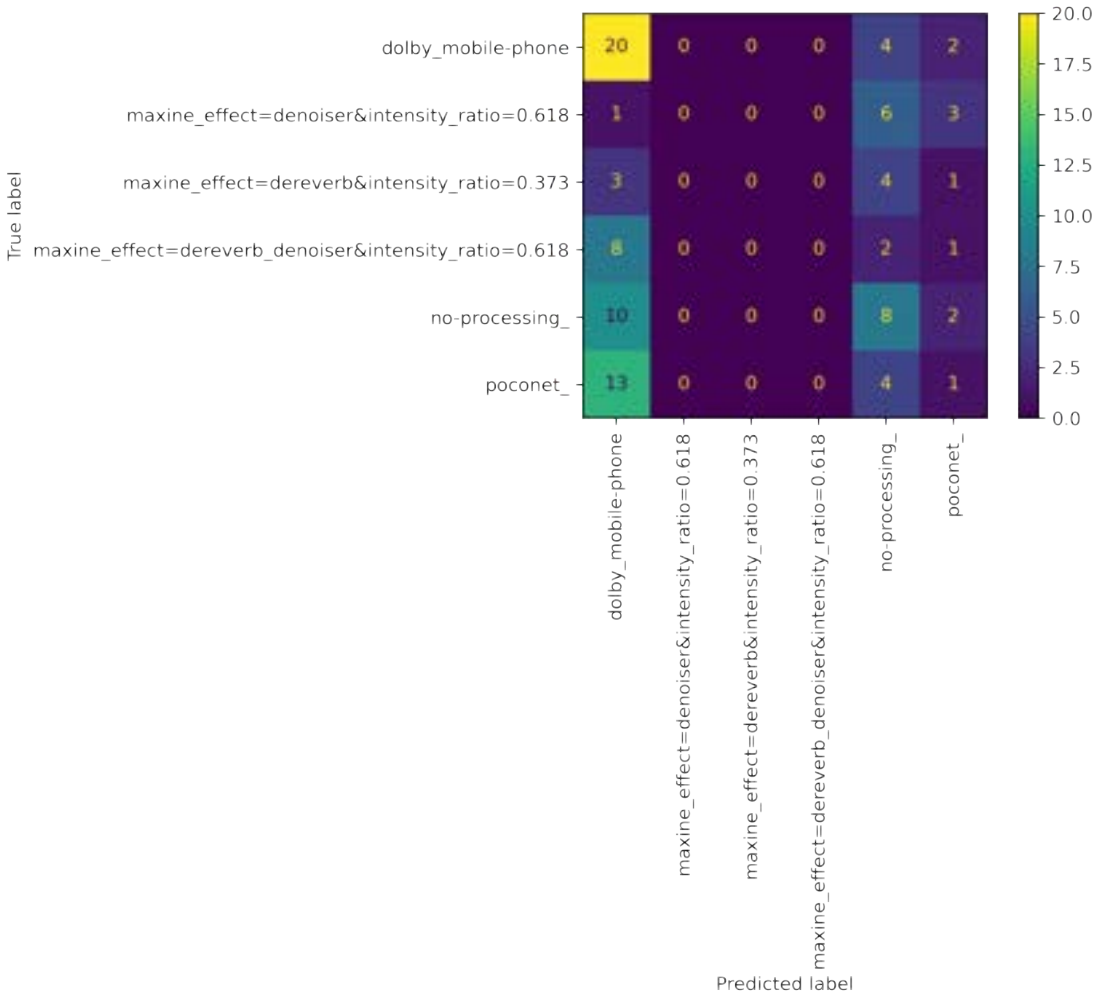

Abbildung A.3.: Confusion Matrix vom SVM_BestProc+NoProc_FeatSel

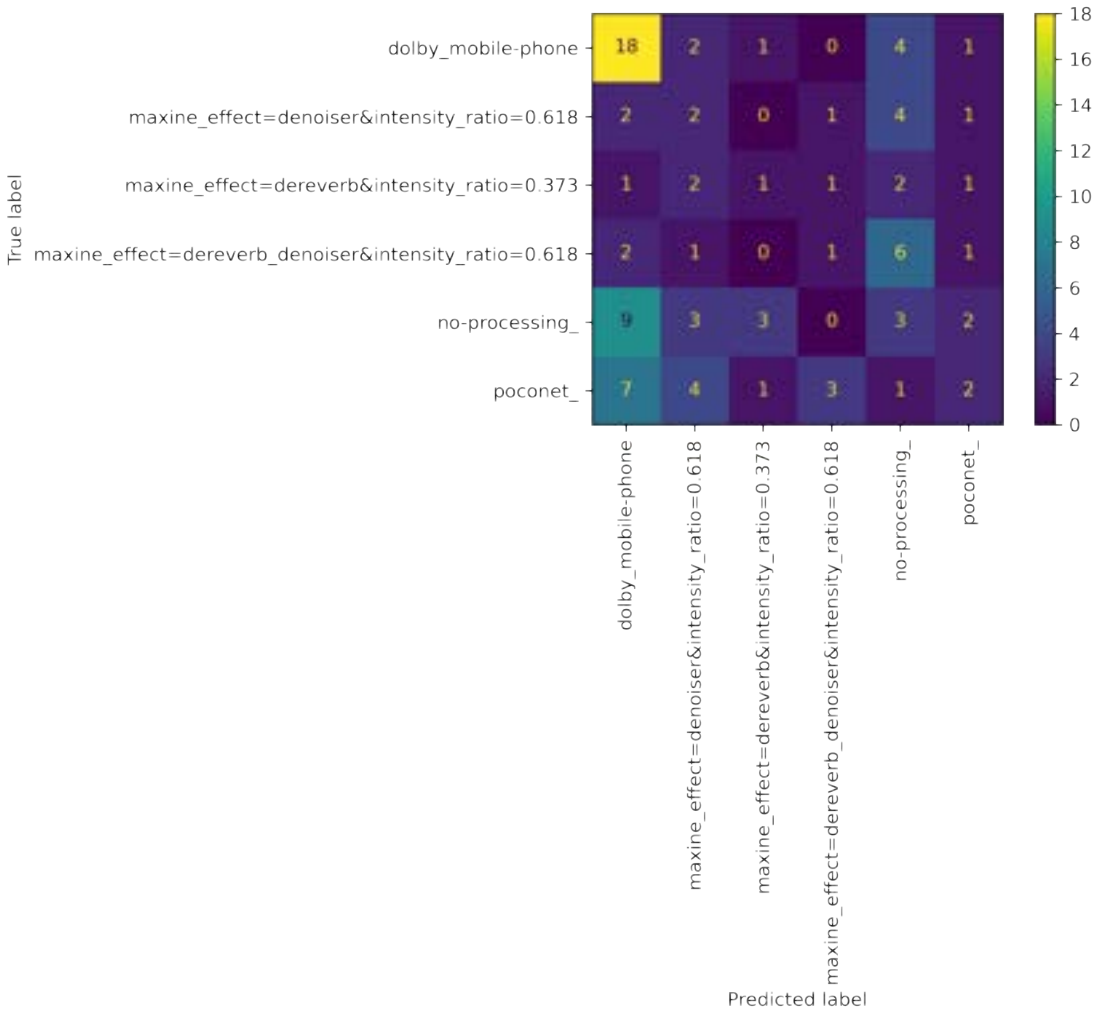

Abbildung A.4.: Confusion Matrix vom SVM_BestProc+NoProc_AllSel+NoCorr

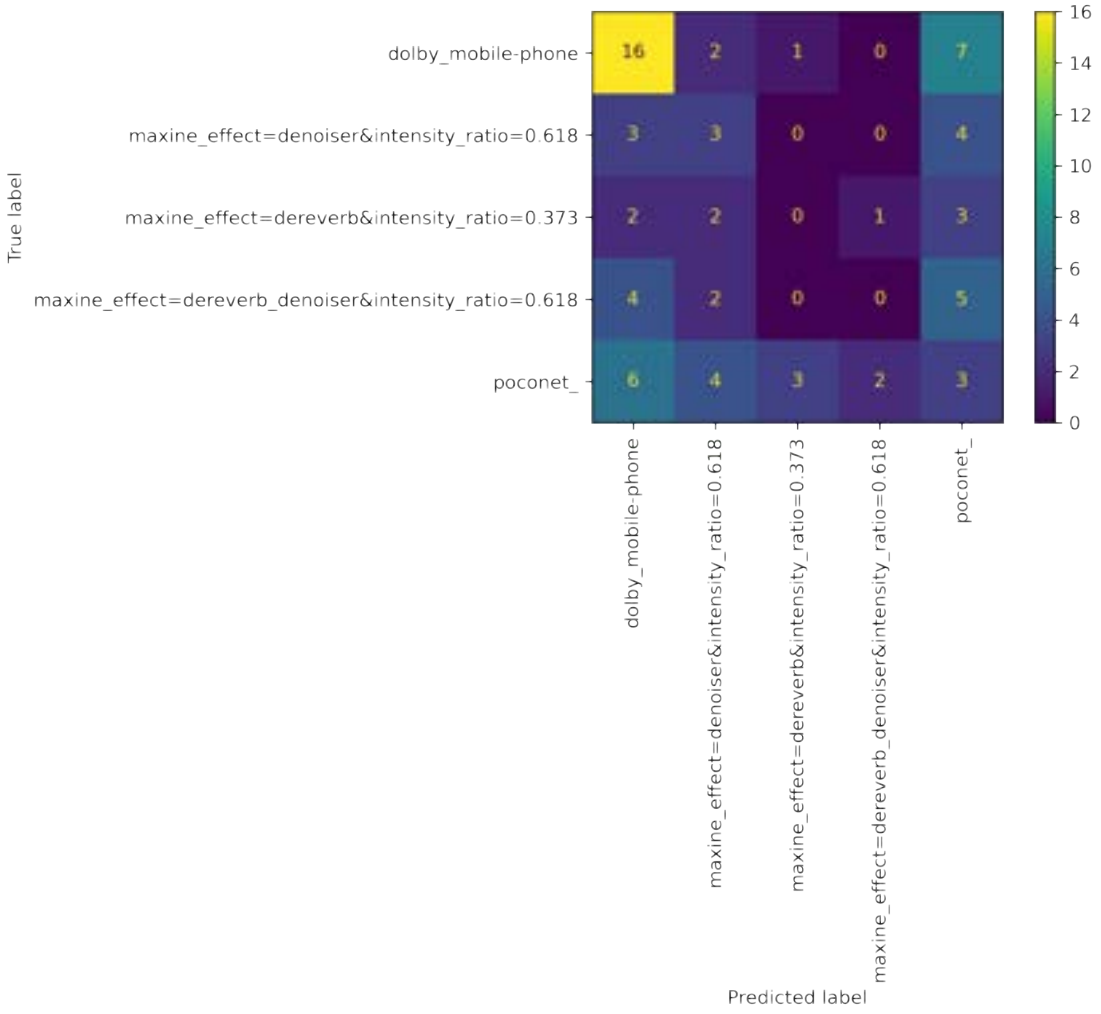

Abbildung A.5.: Confusion Matrix vom SVM_BestProc_AllFeat

A.3 Technische Dokumentation

Der Audiokorpus und die Softwarekomponenten sind auf der Github-Instanz der ZHAW verfügbar. Um die Experimente zu reproduzieren, empfiehlt es sich, alle Repositories in das gleiche Verzeichnis zu klonen. Die einzelnen Repositories sind in der Github-Organisation BA-berweman-tsengmar unter `https://github.zhaw.ch/BA-berweman-tsengmar` zu finden.

A.3.1 Audio Korpus

Der Audiokorpus kann unter `https://github.zhaw.ch/BA-berweman-tsengmar/data` abgerufen werden. Aufgrund des Umfangs der Audioaufnahmen enthält dieser nur die

Transkripte, WERs, Metadaten und Plots. Die Audiodateien wurden den Betreuern zur Verfügung gestellt und können auf Anfrage angefordert werden.

Transcript Aligner

Zum automatischen Kürzen der Transkripte kann TranscriptAligner.py verwendet werden. Dieses ist im Pipeline-Repository unter `./Utils` zu finden. Beim Aufruf des Skripts wird ein Ordner als Parameter übergeben. Dieser Ordner muss die Transkripte der gekürzten Audios und die vollständigen Referenztexte enthalten. Zur Identifizierung werden auch die Endungen der Dateinamen als Parameter übergeben. Wenn das Skript erfolgreich läuft, werden Textdateien mit der Endung _trimmend.txt erstellt.

A.3.2 Audio Speech Enhancement

Docker Umgebung

Fast alle Speech Enhancement Module (ausser Dolby.io siehe unten) wurden containerisiert. Diese können einfach in einer Docker-Umgebung gestartet werden.

1. Sicherstellen, dass Docker installiert ist.
2. `git clone git@github.zhaw.ch:BA-berweman-tsengmar/audio-denoiser-microservices.git`
3. In der docker-compose.yml Datei die entsprechenden Ports anpassen.
4. In der docker-compose.yml Datei eine Umgebungsvariable `APIKEY` setzen, z.B. `APIKEY=JDBR6cGsNVq4LaSTIHQF`
5. Alle Images builden:
   a) `docker build --tag poconet-wrapper poconet-wrapper/`
   b) `docker build --tag noisereduce-wrapper noisereduce-wrapper/`
   c) `docker build --tag nvidia-maxine-wrapper`
   d) `docker build --tag deep-xi-wrapper deep-xi-wrapper/`
   e) `docker build --tag deep-xi-wrapper deep-xi-wrapper/`
6. Die Images mit Docker-Compose starten `docker-compose up -d`

Nun können die Services mit HTTP aufgerufen werden. Zu Testzwecken können Audiodateien über den Browser unter `https://160.85.252.169/maxine/api?apikey=JDBR6cGsNVq4LaSTIHQF` hochgeladen werden. Parameter werden über HTTP Query Parameter übergeben, im Skript `pipeline/main.py` sind die Beispiele zu finden.

Dolby.io Enhance

Der Dolby.io Enhance-Client für das Processing ist im Pipeline Repository unter "./Speech_Enhancement/DolbyIO" zu finden. Dieser kann mit Python gestartet werden, wobei das Verzeichnis mit den Audiodateien und das gewünschte Preset über Parameter angegeben werden können.

```
DolbyIO git:(main) python DolbyIO-Enhance.py
usage: DolbyIO-Enhance.py [-h] input_directory {
   conference,interview,lecture,meeting,mobile_phone,
   music,podcast,studio,voice_over}
DolbyIO-Enhance.py: error: the following arguments are
   required: input_directory, content_type
```

Dolby.io Analyse

Der Dolby.io Analyse-Client für die Metadaten ist auch im Pipeline Repository unter "./Speech_Enhancement/DolbyIO" zu finden. Dieser wird auch mit Python gestartet, wobei das Verzeichnis mit den Audiodateien Parameter angegeben wird.

A.3.3 Audio Processing Client

Ebenfalls im Pipeline Repository befindet sich im Ordner "./Speech_Enhancement die Datei "main.py". Diese nimmt als Argument ein Verzeichnis entgegen in dem sich Audiodateien befinden und wendet auf diese die verschiedenen Speech Enhancements an.

A.3.4 Speech-to-Text Clients

Azure OnPrem

Der Client für Azure onPrem Speech to text ist ebenfalls im Pipeline Repository abgelegt. Er befindet sich unter "./STT/zhawSTT.py". Als Parameter kann ein Verzeichnis und oder eine einzelne Audiodatei angegeben werden, das transkribiert werden soll. Für einzelne Dateien wird das erstellte Transkript direkt neben der angegebenen Datei abgelegt. Bei Verzeichnissen hingegen wird neben dem angegeben Verzeichnis ein neues Verzeichnis mit dem Namen "txt" angelegt, dass eine Kopie der Struktur des angegeben Verzeichnisses beinhaltet. Jedoch befinden sich dort anstelle der Audios dann die Transkripte.

Google Cloud

Um Audioaufnahmen mit Google zu transkribieren, kann das Skript STT/Google/GoogleSTT.py im Pipeline Repository verwendet werden. API Keys und Credentials müs-

sen vorhanden und in der Shell exportiert `export GOOGLE_APPLICATION_CREDENTIALS ="/X/Y/Z/GoogleCreds.json"` sein. Danach kann das Skript einfach mit dem Audioordner als Parameter gestartet werden.

Apple

Apple Transkripte wurden mit hear erstellt. Hear kann hier heruntergeladen werden: `https://github.com/sveinbjornt/hear` Mit dem folgenden Bash-Befehl lassen sich dann die WAV-Dateien transkribieren.

```
ls *.wav | parallel --jobs 8 "[ ! -f {.}apple.txt ] &&
   echo {} && hear -i {} > {.}apple.txt"
```

`parallel` ermöglicht das parallele Ausführen von Befehlen und kann auf macOS mit `brew install parallel` installiert werden.

DeepSpeech

Der Aufruf und das Verhalten dieses Clients sind analog zu Azure OnPrem. Der Client erwartet jedoch, dass unter der hart-codierten IP-Adresse der DeepSpeech Server läuft.

A.3.5 DeepSpeech Server

Analog zu dem Speech Enhancement Containern kann auch ein DeepSpeech Container deployt werden. Aus organisatorischen Gründen befindet sich dieser im Repository `audio-denoiser-microservices`. Dieser Server basiert auf der GPU beschleunigten Variante von DeepSpeech und kann daher nur mit einer solchen gestartet werden.

A.3.6 SVM

Die SVM-Experimente (inkl. Sentence Scoring) sind im Pipeline-Repository zu finden. Die Experimente können einfach über die Jupyter-Notebooks ausgeführt werden, vorausgesetzt, die Audiokorps wurden ebenfalls geklont.

A.3.7 Neuronale Netze

Die Experimente mit den Neuronalen Netzen sind ebenfalls im Pipeline Repository abgelegt. Sie befinden sich im Unterordner `NN`.

A.3.8 Sentence Scoring

Das Sentence Scoring wurde bereits für die Transkripte durchgeführt und kann im Corps unter Metadata gefunden werden. Wenn man weitere Audiodateien scoren möchte,

kann man das Skript `my.py` im Pipeline Repository unter `mlm-scoring/src/my.py` starten. Im Skript müssen audio_dir und metadata_dir angepasst werden, damit die Transkripte geladen werden können und die Werte gespeichert werden können.

A.4 Besprechungsprotokolle

Datum	Beschreibung
22.02.22	Unsere Bachelorarbeit könnte sich in zwei verschiedene Richtungen entwickeln. Erste Möglichkeit: Benutzerunterstützung zur Verbesserung der Aufnahmen. Wie kann ich dem Nutzer zuverlässig mitteilen, wie gut die Qualität der Transkription ist? Zum Beispiel, indem alle Teilnehmer einen Testsatz sprechen, um zu sehen, wie gut die Aufnahme ist. Zweite Möglichkeit: Es gibt bereits Aufnahmen und man versucht, diese zu verbessern. Durch Rauschunterdrückung usw. Wir haben uns für die zweite Möglichkeit entschieden. Die Idee einer Pipeline ist gut. Am besten ist es, selbst Audiodateien zu sammeln, damit die Ergebnisse nicht verfälscht werden. Alle vortrainierten Modelle wurden höchstwahrscheinlich bereits z. B. mit dem Mozilla Common Voice Korpus trainiert. Wir könnten auch untersuchen, ob es in Audiodateien Merkmale gibt, die auf den besten Filter hinweisen. Wo können wir das nachlesen? Interspeech Konferenz, paperswithcode.org, SpecAugment ist ein sehr guter Prozess, um neue/mehr Daten zu generieren.
01.03.22	Wir könnten selbst eine Pipeline aufbauen oder eine externe Pipeline bei uns integrieren, da eine solche bereits existiert. Wir sind noch dabei, den Status der externen Pipeline abzuklären. Im Moment haben wir zu wenig Audiomaterial, vielleicht weiss Flurin Gishammer, wie man auf sauberem Audio künstlich Rauschen erzeugen kann. Die Idee, zwei Präprozessoren auf ein Audio (bzw. alle Kombinationen) anzuwenden, ist noch aufgekommen. Dies benötigt aber viele STT-Stunden.
08.03.22	Der NVIDIA Maxine Speech Enhancer konnte bereits als Microservice implementiert werden. Weitere werden folgen. Der Status der externen Pipeline ist noch unbekannt.

Tabelle A.1.: Besprechungsprotokolle 1 von 2

Datum	Beschreibung
15.03.22	Alle Präprozessoren für das Speech Enhancement wurden containerisiert und sind als Microservices verfügbar. Zudem wurde ein Preprocessing-Skript erstellt, um alle Audiodateien unter Verwendung aller Konfigurationen automatisch zu verarbeiten. Wir haben derzeit 40 Stunden SRF-Audio und 30 GB in englischer Sprache von Rev.com. Wie besprochen werden wir SRF nicht mehr weiterverfolgen, da die Aufnahmen zu wenig Hintergrundgeräusche haben und die Untertitel keine 1:1-Transkriptionen sind.
29.03.22	Wir haben zusätzlich den Deco Korpus erhalten und die ersten Tests durchgeführt. Die Audiodateien klingen beeindruckend. Jetzt müssen wir sie transkribieren und auswerten.
05.04.22	Die WER API hat die schwäche, dass nur eine begrenzte Zahl von Wörtern verwendet werden kann. Wir sollten die Audioaufnahmen und Transkripte kürzen. Dies ist evtl. mit einem Matching eines Transkripts möglich.
12.04.22	Plots vorgestellt, Azure ist besser als Google was nicht sein kann. Bei Google sollte man noch mit dem Modell "video"transkribieren.
29.04.22	Wir haben die SVM vorgestellt. Vielleicht sollten wir andere Klassen klassifizieren, z.B. Verbesserung vs. keine Verbesserung. Für den Dummy-Klassifikator sollten wir nicht die häufigste Klasse nehmen, sondern die Verteilung der Klassen. Bei den Diagrammen könnten wir die Ausreisser entfernen, um die Skalierung deutlicher zu machen. Eine Demo wäre gut für die Verteidigung des Bachelors.
03.05.22	Pre-Prozessor-Auswahl durch Inhaltswahrscheinlichkeit und NNs vorgestellt. SVMs: Prüfen Sie, ob Sie eine Dolby-Verschlechterung (von z.B. 20%) erkennen können. Dolby-Histogramm erstellen. NNs können weiterverfolgt werden, könnten etwas bringen.
10.05.22	Da die SVMs die Eigenschaften nicht lernen können, kann man eventuell Heuristiken von Hand erstellen, z. B. mit dem Cross STT WER. Das Ceaser-Korpus sollte heruntergeladen werden, damit das NN für die WER-Vorhersage noch validiert werden kann.
17.05.22	Die Kapitel im Bericht können frei gewählt werden, wichtig ist nur, dass es einen roten Faden durch die Experimente gibt. Zu Beginn geben Sie einen Literaturüberblick und beschreiben die Ausgangssituation. Dann gibt es eine technische Einführung zum Verständnis der Experimente. Auch Experimente, die nicht funktioniert haben, sollten dokumentiert werden. Der Anhang sollte eine technische Dokumentation enthalten, damit Folgearbeiten durchgeführt werden können. Eine englische und deutsche Zusammenfassung sollte zwei Tage vor der BA-Einreichung an Mark geschickt werden.

Tabelle A.2.: Besprechungsprotokolle 2 von 2

A.5 Zeitplan

Woche	Datum	Arbeiten
1	22.02.22	Start
2	01.03.22	Recherche über Speech Enhancement
3	08.03.22	Speech Enhancement Microservices + Preprocessing Pipeline
4	15.03.22	Download von Audio & Transcripts
5	22.03.22	Bereinigung von Audio & Transcripts
6	29.03.22	Pipeline aufbauen
7	05.04.22	Audio Daten durch Pipeline laufen lassen
8	12.04.22	Analyse der Scores
9	19.04.22	Analyse der Scores
10	26.04.22	Erkenntnisse Aufbereiten
11	03.05.22	Erkenntnisse Aufbereiten / Prediction best Processor
12	10.05.22	Prediction best Processor
13	17.05.22	Prediction best Processor
14	24.05.22	Dokumentation schreiben

Tabelle A.3.: Zeitplan unserer Bachelorarbeit