Zurich University
of Applied Sciences

School of
Engineering
CAI Centre for
Artificial Intelligence

Bachelor’s Thesis in Computer Science

Automatic Identification of Swiss German Dialects using Large Language Models

Authors	Claudio Frei Philippe Schneider
Supervisors	Prof. Dr. Mark Cieliebak Dr. Jasmina Bogojeska
Date	08.06.2023

Abstract

The publication of pre-trained language models enabled the development of various speech technologies for low-resource languages such as Swiss German. The training data required for this has become available with the creation of the SDS-200 corpus and the recent finalisation of the STT4SG-350 corpus. While Swiss German speech-to-text systems are the main research area, the ability to automatically identify Swiss German dialects can help to further improve the performance of such systems. In previous work, Swiss German dialect identification systems were already developed, but recent advancements such as the finalisation of the STT4SG-350 corpus and the publication of the Whisper model provided new resources with the potential to significantly increase the performance in this area. This thesis evaluated how the newly available resources can be leveraged to build the best-performing model for Swiss German dialect identification by training and validating models in various configurations. It has been found that mixing the SDS-200 and STT4SG-350 corpora can achieve promising results, but that the variety of speakers is an important factor to reach good generalisation. Speech augmentation has been found as a promising technique that can help to further increase the performance by artificially increasing the number of samples and the variety of speakers. Additionally, a representation learning approach was evaluated, which has not proven satisfactory. Finally, the newly available resources combined with the gained knowledge enabled an increase of the macro F1 score from 45.95% on classification to four canton groups to 62.76% on classification to seven subregions, an even harder task, thereby setting a new baseline for future systems.

Zusammenfassung

Die Veröffentlichung von vortrainierten Sprachmodellen ermöglichte die Entwicklung verschiedener Sprachtechnologien für ressourcenarme Sprachen wie Schweizerdeutsch. Die dafür notwendigen Trainingsdaten stehen durch die Erstellung des SDS-200 Korpus und die kürzlich erfolgte Fertigstellung des STT4SG-350 Korpus zur Verfügung. Während schweizerdeutsche Sprache-zu-Text-Systeme das Hauptforschungsgebiet sind, kann die Fähigkeit, schweizerdeutsche Dialekte automatisch zu identifizieren, dazu beitragen die Leistung von solchen Systemen weiter zu verbessern. In früheren Arbeiten wurden bereits Systeme zur Identifizierung von schweizerdeutschen Dialekten entwickelt, aber jüngste Fortschritte wie die Fertigstellung des STT4SG-350 Korpus und die Veröffentlichung des Whisper Modells stellten neue Mittel bereit, die das Potenzial haben, die Leistung in diesem Bereich deutlich zu steigern. In dieser Arbeit wurde evaluiert, wie die neu verfügbaren Mittel genutzt werden können, um das leistungsfähigste Modell für die Identifizierung von schweizerdeutschen Dialekten zu erstellen, indem Modelle in verschiedenen Konfigurationen trainiert und validiert wurden. Es wurde festgestellt, dass die Mischung der SDS-200 und STT4SG-350 Korpora vielversprechende Ergebnisse erzielen kann, dass aber die Vielfalt der Sprecher ein wichtiger Faktor ist, um eine gute Generalisierung zu erreichen. Eine weitere vielversprechende Technik ist die Transformierung von Sprachaufnahmen, wodurch sich die Leistung weiter steigern lässt, indem die Anzahl der Sprachaufnahmen und die Vielfalt der Sprecher künstlich erhöht wird. Zusätzlich wurde ein Verfahren zum Lernen von Repräsentationen evaluiert, welches sich aber als nicht zufriedenstellend erwiesen hat. Schlussendlich ermöglichten die neu verfügbaren Mittel in Kombination mit dem gewonnenen Wissen eine Steigerung des Makro-F1-Mass von 45.95% bei der Klassifizierung in vier Kantonsgruppen auf 62.76% bei der Klassifizierung in sieben Dialektregionen, was eine schwierigere Aufgabe darstellt, und setzt damit einen neuen Massstab für zukünftige Systeme.

Preface

Since we are both native Swiss German speakers, the advancement in Swiss German language processing is of particular interest to us. We are grateful for the opportunity to contribute to this research area and we hope that our work will assist in advancing Swiss German language processing and generating greater interest in this field. A special thanks goes to our supervisors Prof. Dr. Mark Cieliebak and Dr. Jasmina Bogojeska, both of whom supported us by sharing their expertise and providing valuable advice throughout the creation of this thesis. Additional thanks goes to the Institute of Applied Information Technology at ZHAW for providing us with the necessary computing capabilities and to everyone involved in the creation of the SDS-200 and STT4SG-350 corpora at FHNW and ZHAW without whom the creation of this thesis would not have been possible.

Contents

1 Introduction

In recent years, numerous efforts have been made to improve speech technologies for low-resource languages. The main problem with such languages is the lack of sufficient supervised data that can be used for training models. As of now, the predominant solution for this problem is the use of unsupervised data for training speech representations as well as using pre-trained cross-lingual models. Examples of such models are the wav2vec family developed by Facebook AI [1]–[4] as well as the recently published Whisper model by Open AI [5].

Following those advancements, it became possible to develop various speech technologies for low-resource languages such as Swiss German, a group of Alemannic dialects spoken in the German-speaking part of Switzerland and Liechtenstein. To collect the necessary supervised data, the FHNW and ZHAW started the "Schweizer Dialektsammlung" project, which resulted in the creation of the SDS-200 corpus that contains spoken sentences in various Swiss German dialects, including the corresponding transcriptions in Standard High German [6]. To further increase the performance of Swiss German speech-to-text systems, the recently finalised STT4SG-350 corpus was created [7].

This thesis focuses on automatically identifying Swiss German dialects using large language models. One of the main uses of such a system is the improvement of regular speech-to-text systems. The assumption is that the performance of speech-to-text systems can be improved by training separate models for each dialect. This makes it necessary to select the correct model for the spoken dialect, a task that can be done using a dialect identification system.

In previous work, systems for Swiss German dialect identification were already developed, but recent advancements have the potential to significantly improve the performance of such systems [8], [9]. For example, the recently finalised STT4SG-350 corpus has become available as an additional source for high-quality audio recordings that can be used for training. Additionally, newer pre-trained models, as well as libraries and frameworks, have been published. Therefore, the aim of this thesis is to evaluate how the newly available resources can be leveraged to build the best-performing model for Swiss German dialect identification.

1.1 Definitions of Terms

Accent refers to a particular pronunciation of words. The Cambridge Dictionary defines accent as “the way in which people in a particular area, country, or social group pronounce words” [10].

Dialect refers to a particular form of language. The Cambridge Dictionary defines dialect as “a form of a language that people speak in a particular part of a country, containing some different words and grammar, etc.” [10].

1.2 Related Work

This chapter will provide an overview of existing literature and previous work around Swiss German dialect identification. While dialect identification strictly speaking is not the same as language and accent identification, the borders between those tasks are not clearly defined and similar methods are used to solve them. However, it is always important to distinguish between identification tasks performed on speech and on written text. While this thesis focuses on spoken dialect identification, written dialect identification is closely related and sometimes even combined to increase performance.

In early works on spoken dialect identification, a Gaussian Mixture Model in combination with a Support Vector Machine was used. In 2008, this approach was used to identify four main Chinese dialects, which reached an accuracy of 92.5% by using a corpus containing 40 speakers per dialect [11]. In 2015, a similar approach reached a precision of 80.49% on spoken Arabic dialect identification on a corpus containing five Arabic Maghreb dialects with a total of 525 speakers [12].

Later work mainly shifts toward the usage of neural networks, especially convolutional neural networks. In 2018, an accuracy of 65.55% was reached on the MGB-3 corpus containing five Arabic dialects by using a convolutional neural network on acoustic features and by using speech augmentation the accuracy was even increased to 68.82% [13]. In 2019, a convolutional neural network with variable filter sizes was used to achieve a classification accuracy of 90% on native English accents, 71% on Arabic accents and 50% on Mandarin accents [14].

One of the newest works in the field of dialect identification is the paper “Arabic Speech Dialect classification using Deep Learning” [15], which was presented at the 1st International Conference on Advanced Innovations in Smart Cities and published in April 2023. The paper described how eight Arabic dialects were identified with an accuracy of 83% by using a convolutional neural network on spectrogram images. For the experiment, only 84 audio files per dialect with a duration of around 5 to 20 seconds were used.

In previous work, various large language models have already been proven to be capable of identifying speakers, languages and accents. In 2020, various deep learning methods, including the wav2vec model, were compared in language identification on the Mozilla's Common Voice corpus [16]. It was found that while wav2vec did not perform best, it still achieved an F1 score of 79% and outperformed the convolutional neural network, which achieved an F1 score of 75%. In the same comparison, the SpecAugment augmentation method was tested, which raised the F1 score of the convolutional neural network from 75% to 85% [17].

In a bachelor's thesis from 2021, the wav2vec2-large-xlsr-53 model was tested on various speech classification tasks such as English and Spanish accent classification as well as age and sex classification in German [18]. The models were trained on 1.5 to 8 hours of speech per class extracted from Mozilla's Common Voice corpus. It was found that while it was possible to use wav2vec models for classification, the scores did not yet correspond to the desired values and more samples are needed to further increase the performance.

When it comes to the more recently published Whisper model, not yet many papers using it for speech classification tasks have been published [5]. However, from the experiments done in the Whisper paper, it is known that Whisper outperforms wav2vec when it comes to speech-to-text tasks.

Most studies have been conducted on non-Germanic languages such as Arabic, Chinese, English, Finnish or Spanish. Only a limited number of studies regarding spoken dialect identification are available on Swiss German dialects. The reason for this is that Swiss German is not a widely spoken language and therefore only a limited number of audio corpora are available, which could be used for dialect identification.

ArchiMob is a corpus that contains continuous speech with around 555 recordings from various places in Switzerland [19]. The Swiss Parliaments Corpus is a collection of publicly available audio files from the Bernese cantonal parliament [20]. The corpus consists of 293 hours of audio in mainly Bernese spoken dialect. Another corpus which only covers the region of Valais, is the Radio Rottu Oberwallis corpus [21]. It is therefore limited to dialects spoken in upper Valais. The ETH Zürich released with the SwissDial a corpus with high quality recordings split into 8 different dialect regions [22]. A newer corpus is the SDS-200 corpus which contains 200 hours of spoken sentences recorded in Swiss German by nearly 4,000 participants [6]. The STT4SG-350 corpus is the newest available corpus, which contains 343 hours of spoken Swiss German dialects from 316 speakers [7].

For written Swiss German dialect identification, three shared tasks were organised as part of the VarDial Evaluation Campaign in the years 2017, 2018 and 2019 [23]–[25]. In all the three shared tasks, manually created speech transcriptions in the writing system "Schwyzertutschi Dialäktschrift" [26] from the four dialect regions Basel, Bern, Lucerne and Zurich were used, which have been extracted from the ArchiMob corpus. In 2017, the best team achieved a weighted F1 score of 66.2% [27]. A year later in 2018, the transcriptions were improved, which allowed the best team to achieve a macro F1 score of 68.6% [28]. In the third and last shared task in 2019, the participants were additionally allowed to make use of acoustic features, which resulted in the best team achieving a macro F1 score of 75.93% [29].

Another shared task on written Swiss German but on language identification was organised as part of the GermEval in 2020 with the goal of creating a binary classifier to decide between tweets written in Swiss German and tweets not written in Swiss German [30]. The best team was able to achieve an F1 score of 98.2% [31].

For spoken Swiss German dialect identification, which is the topic of this thesis, two direct predecessor theses using wav2vec2-xls-r-300m were already performed at the Centre of Artificial Intelligence at the ZHAW. The first one is a project thesis done in 2021 on the SDS-200 corpus in which the capabilities of wav2vec for Swiss German dialect identification were tested [8]. While it was possible to achieve a macro F1 score of 45.95% on classification to four canton groups, it was found that there is a lack of reproducibility and a certain degree of uncertainty in the results. The second one is a bachelor's thesis done in 2022 about pre-training wav2vec using unsupervised Swiss German speech [9]. While it was found that the unsupervised pre-training was not beneficial for speech translation, it might have a positive impact on dialect identification. Both of those theses did find that the SDS-200 corpus was of insufficient quality in terms of the amount of data as well as class balance.

1.3 Outline

Following this introduction, the second chapter provides a brief overview of the theoretical framework used in this thesis. Afterwards, the used experimental setup is described, including an analysis of the used corpora, a breakdown of the performed data pre-processing as well as general information about the employed experimental environment. The next chapter then contains all the conducted experiments with each experiment having separate sections for the applied methods followed by the obtained results. The thesis then concludes with a summary of all the results of the experiments as well as a short discussion.

2 Theoretical Framework

In this chapter, the theoretical framework, including definitions of key concepts and technologies important for our research, is further explained. Readers already familiar with the content of this chapter might skip it.

2.1 Learning Methods

In machine learning, models are trained to predict output for unseen samples. To do this, the model learns from the experience provided by a set of training samples. Different learning methods are available which can be used to learn from training data.

In **Supervised Learning,** a model is presented with input samples as well as the corresponding target outputs during training [32]. With this information, the model can fit itself onto the training data. One of the challenges with supervised learning is the lack of high-quality labelled data which can be used for training.

Unsupervised Learning is a form of learning in which a model is only presented with input samples, and no target outputs are available [32]. With this form of learning, the model needs to fit itself onto the hidden patterns within the data itself.

Representation Learning, also known as feature learning, refers to the automatic discovery of useful representations from raw data such as audio, image, or text [33]. The learned representations can then be used for classification or various other tasks.

For **Contrastive Learning,** a model is given pairs of samples that are either similar or dissimilar [34]. During training, the model then learns features that are common between similar samples and features which help to differentiate between dissimilar samples. Contrastive learning has proven to be a successful learning method in computer vision [35].

Deep Learning is a special form of representation learning in which deep neural networks with multiple layers are used to learn representations of data at different levels of abstraction [36]. This technique brought a breakthrough in computer vision.

It is also possible to combine different learning methods. For example, semi-supervised learning is a combination of supervised and unsupervised learning.

2.2 Loss Functions

During training, it is necessary to specify the training objective of a model. Typically, this takes the form of a loss function. A loss function is a mathematical function which maps the produced output of a model together with some information from the input sample to a numeric value. Commonly, the objective during training is then to minimise the loss function using gradient descent. Most frameworks provide a large variety of commonly used loss functions, such as cross-entropy loss or mean square error loss. A common representation of a loss function is $L(x, y)$, where x stands for the output of the model and y stands for some information from the sample.

2.2.1 Cross-Entropy Loss

The cross-entropy loss[1] is one of the most widely used loss functions. The cross-entropy measures the difference between two probability distributions. This is useful for measuring the difference between the output probability distribution of a model to the expected probability distribution. For classification, the cross-entropy loss can be written like in the Equation 1 below.

$$L(x, y)_{CE} = -\sum_{c=1}^{C} \log(x_c)\, y_c$$

Equation 1: Cross-Entropy Loss Function

In the Equation 1 above, C stands for the number of classes, x and y are two vectors containing a probability for each class.

2.2.2 Negative Likelihood Loss

Another loss functions commonly used for multi-class classification tasks is the negative likelihood loss[2]. With this loss function, the model can be trained to maximise the likelihood that the correct class is predicted by minimising the negative likelihood. The Equation 2 below shows the function of the negative likelihood loss.

$$L(x, y)_{NL} = -\log(x_y)$$

Equation 2: Negative Likelihood Loss Function

In the Equation 2 above, x is a vector containing a probability for each class and y is the index of the correct class. The negative likelihood loss function is the same as the cross-entropy loss function in case samples are assigned to only one correct class.

[1] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

[2] https://pytorch.org/docs/stable/generated/torch.nn.NLLLoss.html#torch.nn.NLLLoss

2.2.3 Cosine Embedding Loss

Models which produce embeddings can be trained using the cosine embedding loss[3]. With this loss function, the cosine similarity of embeddings from similar samples is increased while the cosine similarity of embeddings from dissimilar samples is decreased. This is mainly useful when using contrastive learning to learn meaningful embeddings. The Equation 3 below shows the function of the cosine embedding loss.

$$L(x,y)_{CE} = \begin{cases} 1 - cos(x_1, x_2), & y = 1 \\ cos(x_1, x_2), & y = -1 \end{cases}$$

Equation 3: Cosine Embedding Loss Function

In the Equation 3 above, x_1 and x_2 are two embedding vectors and y is either 1 in case the embeddings are supposed to be similar or -1 in case the embeddings are supposed to be dissimilar.

2.3 Large Language Models

While no formal definition of large language models exists, they are typically large and deep neural networks with millions of parameters trained on large quantities of data. A key aspect of large language models is that they are not limited to a specific task but rather can be used for multiple different tasks in the field of natural language processing.

2.3.1 Attention

A key concept introduced by Bahdanau in 2016 is the concept of attention in recurrent neural networks [37]. Attention allows a recurrent neural network to automatically highlight parts of an embedding that are more relevant for producing the output. In practice, this works much like cognitive attention. Initially, attention was proposed as a solution to improve the performance in translation tasks when using recurrent neural networks, but since then has been applied much more broadly.

2.3.2 Transformer Architecture

In the Google-published paper "Attention Is All You Need" [38] the usage of attention was proposed as a replacement for recurrent neural networks, taking the form of an encoder-decoder architecture based on self-attention in a multi-head setup. This transformer architecture became the state-of-the-art standard in the field of natural language processing, especially in sequence-to-sequence tasks such as text-to-text or speech-to-text. The strength of the transformer architecture mainly lies within the self-attention mechanism, which allows the model to better capture the context when encoding an input. Another strength of the transformer architecture is that it does not process the input sequentially and therefore can take more advantage of

[3] https://pytorch.org/docs/stable/generated/torch.nn.CosineEmbeddingLoss.html

parallelism during training and inference. This allows transformer models to train much faster and scale to large numbers of parameters and to large quantities of data.

The transformer architecture is split into an encoder and a decoder. The former encodes an input sequence into hidden states before the latter is repeatedly invoked to decode the hidden state and the previous output, thus producing the next output.

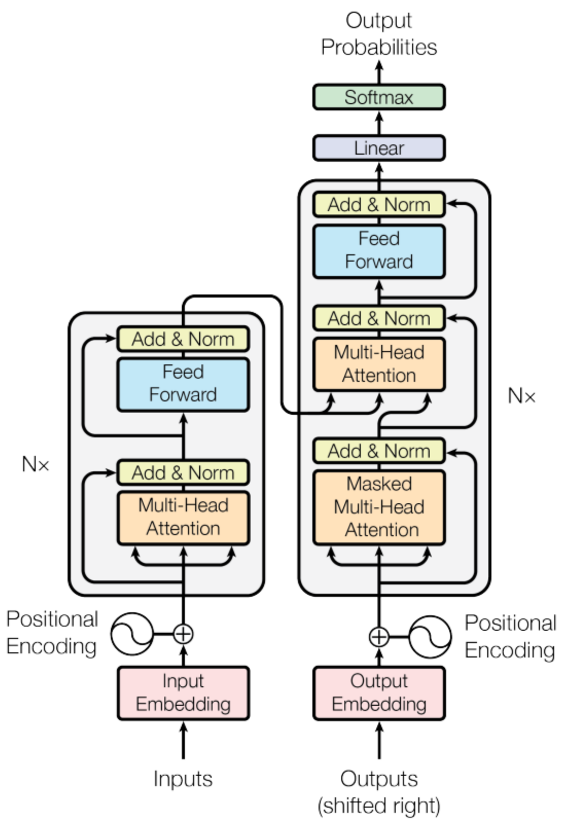


Figure 1: Transformer Architecture [38]

The Figure 1 above, from the original "Attention Is All You Need" [38] paper, visualises the transformer architecture. The input is fed into the encoder network on the left to produce an embedding. The decoder network on the right receives the previous output as well as the embedding from the encoder and produces an output.

An additional benefit of the transformer architecture is that the encoder produces embeddings that can also be used for other downstream tasks such as classification. Because of that only the encoder part is used in this thesis.

2.3.3 wav2vec

The wav2vec model was first introduced in the paper "wav2vec: Unsupervised Pre-Training For Speech Recognition" [2] by Facebook AI in 2019. A year later in 2020, a major revision was published with the paper "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations" [1] which integrates parts of the transformer architecture. Throughout this thesis, wav2vec always refers to wav2vec 2.0.

The wav2vec models consume raw waveforms and produce an embedding. Internally, a convolutional neural network is used to extract features from the raw waveforms for each timestep. The extracted features are then fed through a context network which produces contextualised representations using self-attention.

Wav2vec models are pre-trained on large amounts of unsupervised data using contrastive learning and can then be fine-tuned for various speech tasks such as speech recognition. Most published wav2vec models are pre-trained on English corpora. For example, the wav2vec2-base-960h[4] model is pre-trained and fine-tuned on 960 hours from the LibriSpeech corpus [1].

With the publication of the paper "Unsupervised Cross-lingual Representation Learning for Speech Recognition" [3] in 2020, a multilingual model called wav2vec2-large-xlsr-53[5] trained on 56 thousand hours of speech in 53 different languages was released. A year later in 2021, with the publication of the paper "XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale" [4], even larger multilingual models have become available in different sizes, which were pre-trained on 436 thousand hours in 53 different languages. One of them called wav2vec2-xls-r-300m[6] with 300 million parameters was used throughout this thesis.

Because it is possible to fine-tune wav2vec models on different languages to those on which they have been pre-trained, the models can be used for solving various speech tasks in multiple languages. For example, in this thesis the wav2vec2-xls-r-300m model will be fine-tuned on Swiss German, a language on which the model has not been pre-trained.

2.3.4 Whisper

The Whisper model is a recently published transformer-based neural network featuring both an encoder and a decoder [5]. Instead of using unsupervised learning like wav2vec models, Whisper was trained to transcribe speech on large amounts of supervised data from the web. In total 680k hours of labelled audio data was used to train the model. From this 680k hours of data, 117k hours are from 96 languages other than English. This allows the model to perform on other languages and to translate to English.

Instead of raw waveforms, Whisper consumes spectrograms with a fixed length of 30 seconds in which the frequencies have been converted to the mel scale. Internally, Whisper uses two convolutional layers followed by multiple layers of transformer encoders using multi-head self-attention to produce embeddings which are

[4] https://huggingface.co/facebook/wav2vec2-base-960h

[5] https://huggingface.co/facebook/wav2vec2-large-xlsr-53

[6] https://huggingface.co/facebook/wav2vec2-xls-r-300m

then decoded by multiple layers of transformer decoders using multi-head self-attention and cross-attention. The Whisper model is available in different sizes as shown in the Table 1 below.

Model Name	Layers	Width	Heads	Parameters
whisper-tiny[7]	4	384	6	39 million
whisper-base[8]	6	512	8	74 million
whisper-small[9]	12	768	12	244 million
whisper-medium[10]	24	1,024	16	769 million
whisper-large[11]	32	1,280	20	1,550 million

Table 1: Whisper Model Sizes [5]

In the Table 1 above, the layers column refers to the number of transformer encoder and decoder layers, the width column refers to the size of the embeddings produced by the encoders and the heads column refers to the number of self-attention and cross-attention heads.

While Whisper has been trained to transcribe audio to text, the created embeddings of the encoder can be used for various other speech tasks as well. In this thesis, the encoder part of Whisper is used as a drop-in replacement for wav2vec to train downstream classification tasks.

2.4 Hierarchical Data Format

The hierarchical data format[12] (HDF5) is a binary data format designed to store large amounts of complex data. Originally the format was developed by the National Center for Supercomputing Applications at the University of Illinois Urbana-Champaign. The high performance, portability and versatility of the file format make it a common choice for storing scientific data in academic research or industries such as aerospace, biotech, financial services and automotive. In machine learning the format is especially useful when working with large quantities of image or audio files, because the data can be stored and accessed in a raw format which avoids the additional decoding overhead during training.

[7] https://huggingface.co/openai/whisper-tiny

[8] https://huggingface.co/openai/whisper-base

[9] https://huggingface.co/openai/whisper-small

[10] https://huggingface.co/openai/whisper-medium

[11] https://huggingface.co/openai/whisper-large-v2

[12] https://www.hdfgroup.org/solutions/hdf5

2.5 Classification Metrics

For classification tasks, the commonly used metrics are accuracy, precision, recall and F1 score. These classification metrics are based on the true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) measures [39].

Accuracy is the ratio of true positives and true negatives to the total number of samples expressed as a percentage [39]. It measures how accurately a model makes predications. A high accuracy indicates that the model mostly makes correct predictions.

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

Equation 4: Accuracy Formula [39]

Precision is the ratio of true positives to the true and false positives expressed as a percentage [39]. It measures how precisely a model predicts the true positives. A high precision indicates that most positives were predicted correctly and therefore is especially important when it is undesired to predict false positives.

$$Precision = \frac{TP}{TP + FP}$$

Equation 5: Precision Formula [39]

Recall is the ratio of true positives to the true positives and false negatives. It measures the percentage of the true positives to the actual positives [39]. A high recall indicates that most actual positives were predicted correctly and is therefore especially important when it is undesired to predict false negatives.

$$Recall = \frac{TP}{TP + FN}$$

Equation 6: Recall Formula [39]

The **F1 score** is the harmonic mean of precision and recall [39]. When precision and recall are equally important, the F1 score should be used as it combines both metrics into one.

$$F_1 = 2\,\frac{Precision * Recall}{Precision + Recall}$$

Equation 7: F1 Score Formula [39]

For multi-class classification tasks, the precision and recall metrics are calculated for each class and then averaged to compute the overall metrics for the model. Multiple averaging methods are available.

The **micro average** is computed over all samples and for multi-class classification it is equal to the overall accuracy [39].

The **weighted average** of a metric is the mean of the per-class metrics weighted by the support of each class [39].

The **macro average** of a metric is the arithmetic mean over all classes [39]. The macro average is especially important when the support for each class is not balanced but all classes should be rated equally.

3 Experimental Setup

This chapter contains general information about the experimental setup used for all the experiments. This includes an analysis of the given corpora, a description of the performed data pre-processing as well as information about used metrics, infrastructure, frameworks and training parameters.

3.1 Corpora

For this thesis, only the SDS-200 and STT4SG-350 corpora were used [6], [7]. The Swiss Parliaments Corpus contains mainly Bernese dialect and is therefore not suitable for dialect identification [20]. The same applies to the Radio Rottu Oberwallis corpus which exclusively contains dialects within Wallis [21].

Within this chapter, the SDS-200 and STT4SG-350 corpora will be introduced, analysed and compared. Both corpora are some of the newest Swiss German speech corpora and cover a wide range of Swiss German speaking regions in Switzerland.

3.1.1 SDS-200 Corpus

The SDS-200 corpus is a collection of 200 hours of Swiss German dialectal speech [6]. The corpus is publicly available on SwissNLP [40]. It contains recordings of Swiss German dialectical speech with a wide range of vocabulary. The corpus was created with an online web recording tool, which presented volunteers with a sentence in Standard High German that they then had to translate into their corresponding Swiss German dialect. These sentences were chosen from Swiss newspapers. The volunteers also provided personal information like their age and the provenance of the dialect. This corpus is well suited for dialect identification. The recordings were saved in the MP3 format and validated by other participants which increased the accuracy of the corpus. Finally, a corpus containing 142,545 samples from 3,816 speakers was made publicly available.

3.1.2 STT4SG-350 Corpus

The STT4SG-350 corpus is the newest corpus available regarding Swiss German dialectal speech [7]. It contains 343 hours of speech provided by 316 different speakers who have created 247,527 recordings.

The corpus was created similarly to the SDS-200 corpus with a web app and a validation process [6]. In phase 1 of the collection process, recordings were saved in the MP3 format, but for phase 2 the format was changed to FLAC. The recorded sentences were chosen from newspapers and parliament minutes. The recordings were enhanced with the provenance of the dialect as well as the gender and age of the speaker. Instead of creating contests or advertisements on social media like it was done for the SDS-200 corpus, participants for the STT4SG-350 corpus were paid and chosen via testingtime[13] and seniorsatwork[14].

The corpus is divided into seven subregions[15] called Basel, Bern, Grisons, Central Switzerland, Eastern Switzerland, Valais, and Zurich [7]. Through the selection of speakers during recruitment, the number of speakers per subregion was achieved to be well balanced. Finally, a well-balanced corpus of around 45 speakers and 30,000 to 40,000 recordings per subregion was made available.

3.1.3 Corpora Analysis and Comparison

To gain a comprehensive understanding of the coverage of German-speaking parts of Switzerland, the location of samples was rendered on a map. For this the zip code was used, which is present in both corpora and represents the dialectical provenance of each speaker. To render the samples on a map, we enhanced each sample with the longitude and latitude by using the official directory of cities and towns provided by the Federal Office of Topography (swisstopo) [41].

In case a zip code appeared multiple times within the directory, the latitude and longitude of the central location was used. For example, the zip code 8471 for which entries exist with the names Rutschwil, Dägerlen, Oberwil, Berg and Bänk, each having slightly different longitude and latitude.

Finally, with the help of the Python packages OSMnx[16] and Openstreetmap[17] a map of Switzerland was rendered with the location of each sample for both corpora, shown in the Figure 2 on the next page.

[13] https://www.testingtime.com

[14] https://www.seniorsatwork.ch

[15] In the latest publication of the STT4SG-350 corpus, the subregion attribute was renamed to dialect region. Whenever subregion is mentioned throughout this thesis it is referred to the dialect region.

[16] https://osmnx.readthedocs.io/en/stable

[17] https://www.openstreetmap.org

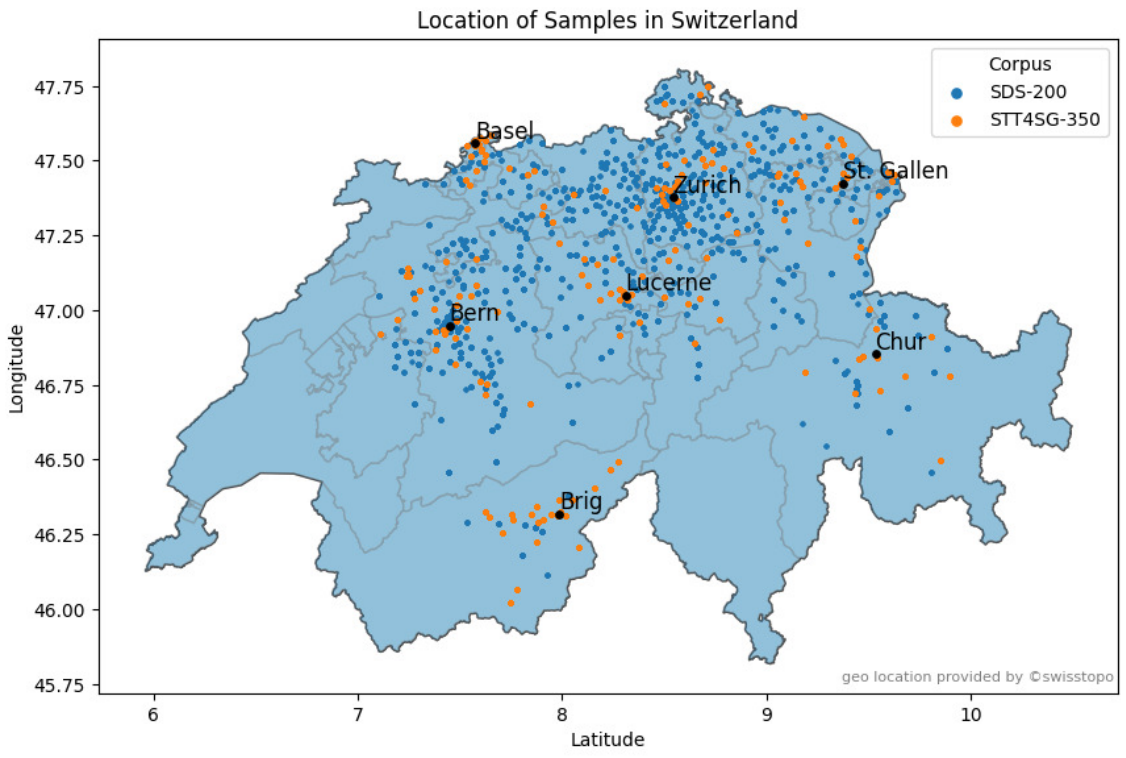
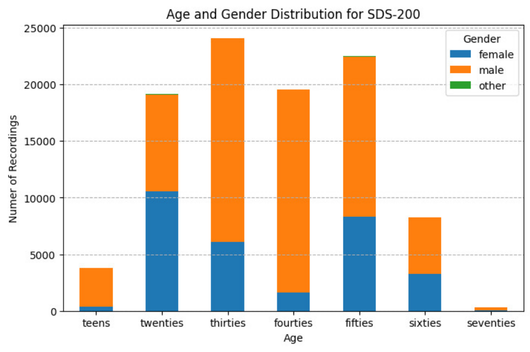


Figure 2: Location of Samples in Switzerland

As seen in the Figure 2 above, a lot of samples are located around Zurich and especially for the STT4SG-350 around Basel. Unfortunately, the regions Grisons and Valais are quite empty. The publishers of the STT4SG-350 corpus mentioned the challenges of finding enough participants for the subregion Valais [7].

Both corpora contain samples from speakers with different genders and ages, but only a few samples for teens and people in their seventies. The gender and age distribution for both corpora are shown in the Figure 3 and Figure 4 below.



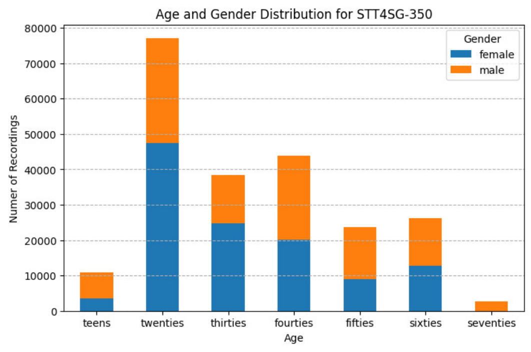


Figure 3: Age and Gender Distribution for SDS-200

Figure 4: Age and Gender Distribution for STT4SG-350

The figures above visualises that the age distribution is more balanced in the SDS-200 corpus compared to the STT4SG-350 corpus, whereas the gender distribution is more balanced in the STT4SG-350 corpus compared to the SDS-200 corpus.

Unfortunately, there is no definitive evidence to confirm whether any participants in the data collection for the SDS-200 corpus also participated in the data collection for the STT4SG-350 corpus. However, when comparing the available information such as zip codes, gender, and age across both corpora, there is a possibility of at least 65 speakers being present within both corpora. Upon manual comparison of recordings, a high likelihood was observed that at least one participant appears in both corpora. The voice of the client 84e1fca6-1a6e-4579-9688-072cc01ec094 from the SDS-200 corpus and the client c382ed83-337b-4571-baf3-008518ede862 from the STT4SG-350 corpus sound remarkably similar[18]. Any other client overlaps have been ignored for the purpose of this thesis. It is important to note that, under certain conditions, the same speaker may be present in both data corpora with different client identifiers.

When it comes to the distribution of samples per speaker, the STT4SG-350 has much more samples per speaker as shown in the Figure 5 below.

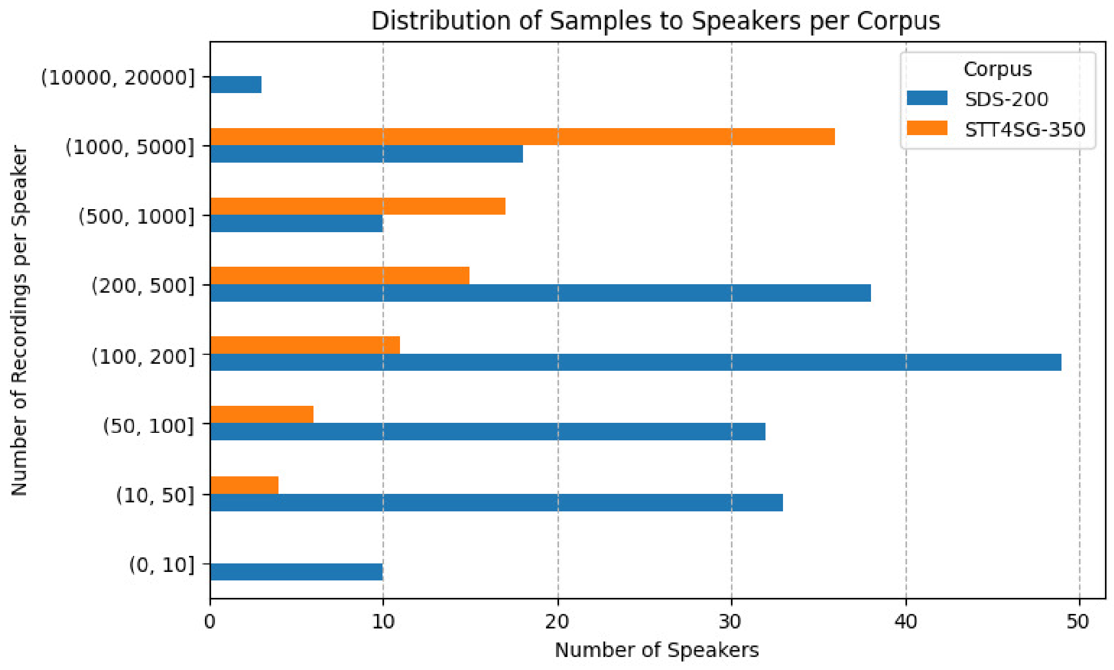


Figure 5: Distribution of Samples to Speakers per Corpus

In the SDS-200 corpus most speakers recorded between 100 and 200 samples. But overall, the number of samples per speaker has a high variance. Three speakers even have more than 10,000 samples in the SDS-200 corpus. In the STT4SG-350 the distribution of samples to speakers looks quite different, with most speakers having between 1,000 and 5,000 samples. In later experiments conducted in this thesis, the distribution of samples to speakers proved to be an important factor for dialect identification.

[18] *Through all experiments in this thesis, these clients were contained within the same train split.*

3.2 Data Pre-Processing

Before training models on the data from the corpora, certain pre-processing steps were necessary. The pre-processing includes conversion, normalisation, and enrichment of data. In the following subchapters, the performed pre-processing steps are explained in detail.

3.2.1 Re-Encoding of Audio Files

The speech recordings from the newer STT4SG-350 corpus were saved in the FLAC[19] format. With this format, no audio data is lost during encoding but with the trade-off of larger file sizes. Unfortunately, the FLAC files of the STT4SG-350 corpus were not readable by the various libraries and commonly used media players. For example, Windows Media Player and QuickTime Player had difficulties playing these files. The other files saved in the MP3 format did not have the same problems.

The root cause of those problems was missing audio duration information in the header of the FLAC files. To fix the issue and allow usage of the files, the missing header information needed to be added. To accomplish this, the FFMPEG tool[20] was used to re-encode all the affected FLAC files, without change the audio itself.

3.2.2 Decoding of Audio Files

The used neural networks expect to receive the audio data as raw waveforms sampled at 16 kHz. This makes it necessary to decode the audio files from an audio format such as MP3 or FLAC to raw waveforms sampled at 16kHz before they can be fed into the neural networks.

To avoid doing this multiple times for each experiment, the audio files were decoded and then packed into a data file during a pre-processing step. During training or testing, the data was then already prepared to be fed through the neural networks and just had to be read from the data file.

The HDF5[21] format, as described in the Chapter 2.4 above, was chosen for this as it allows fast random access to data regions and supports large amounts of data. The same format was already successfully used in a preceding thesis [9].

The final data file reached a size of over 118 gigabytes and took multiple hours to create. But it only had to be created once on each machine and was then used for all the experiments on that machine.

[19] https://xiph.org/flac

[20] https://ffmpeg.org

[21] https://www.hdfgroup.org/solutions/hdf5

3.2.3 Removal of Samples with Missing Attributes

In the SDS-200 corpus, numerous samples were deemed unusable and were consequently excluded from this thesis. A critical factor contributing to this exclusion was the lack of location metadata for the speakers who recorded these samples. The absence of location metadata such as zip code and canton rendered approximately 29,428 data samples from 2,798 speakers unusable for this thesis. Moreover, an additional 6,136 clips within the SDS-200 corpus were discarded due to their invalid status, as indicated by the "clip_is_valid" column.

Within the STT4SG-350 corpus, five speakers had missing zip code attributes. To add the missing attributes, we contacted the affected speakers and asked them whether they can provide the missing information. Unfortunately, the process of receiving their contact information, contacting them and receiving their answers took an extensive amount of time. Because at this point the first experiments were already carried out, we decided to not include these samples within the experiments of this thesis. However, the samples of those speakers can be used for future work. The Table 2 below lists the newly provided zip codes of each affected speaker.

Affected Client ID of Speaker	Newly Provided Zip Code
3ce6b6dc-6c03-48c1-83bd-024bbecfaf9a	8000
3e4bf13e-aaa4-454c-9b3e-02de4b8c7b6a	8953
4bda89af-ea8f-42aa-9f1d-b869843a42fa	3600
93ca247b-14da-4169-875a-ebb71da3e8de	7000
fd385e2c-0e92-4a23-914e-420a9106ae33	3007

Table 2: Speakers with Missing Zip Code Attribute

3.2.4 Removal of Corrupt and Long Recordings

When working with variable-length data, the batch size needs to be selected so that the batch with the longest sample fits into memory. This means that the maximum possible batch size for a given machine depends on the length of the longest sample. During the first experiments often "Out of Memory" failures occurred because of some very long samples in the train dataset. After an analysis of the clip length distribution, some outliners with a duration of more than 16 seconds were identified, as visualised in the Figure 6 on the next page.

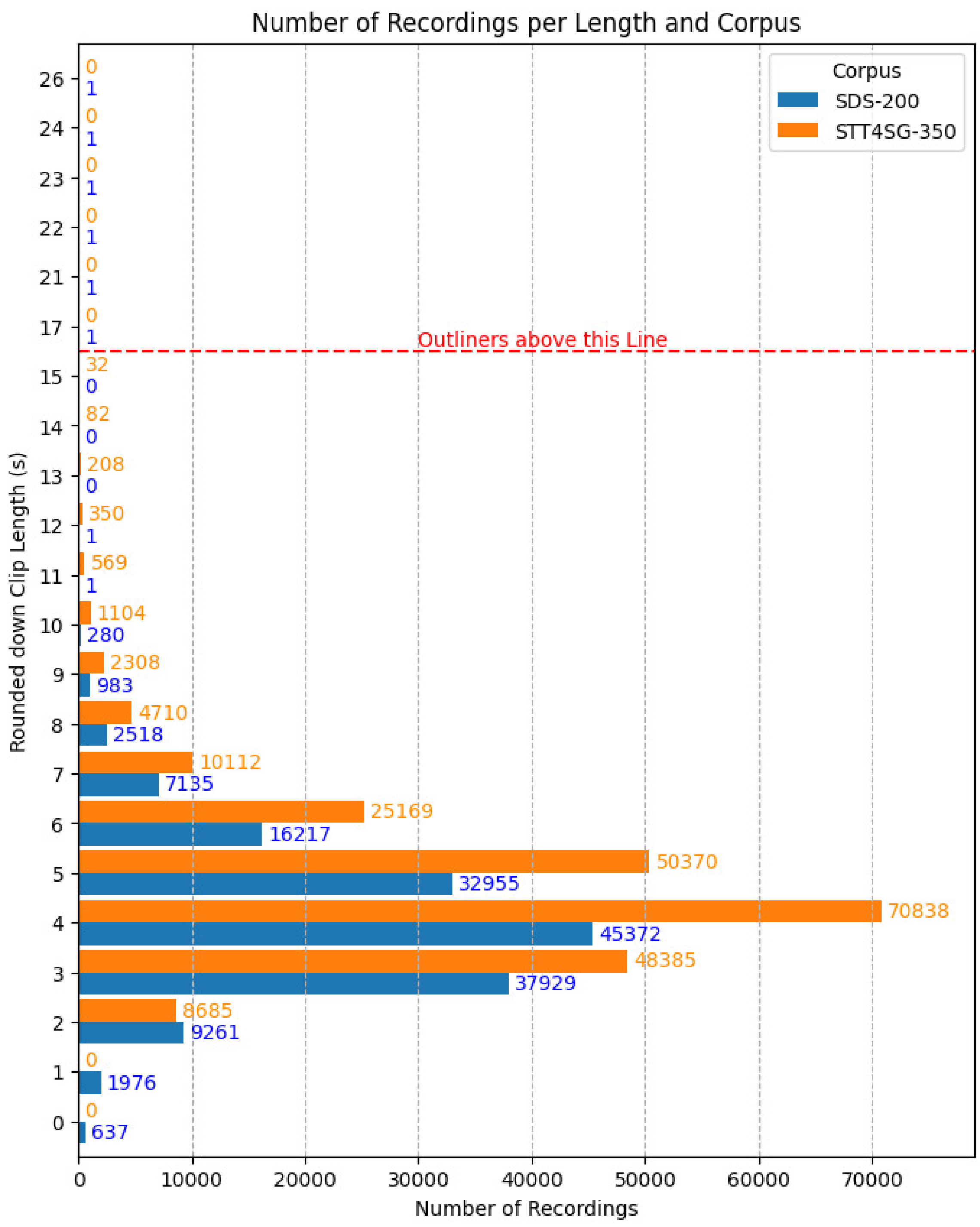


Figure 6: Number of Recordings per Length and Corpus

During a detailed analysis of the outliners, six clips were identified in the SDS-200 corpus from the same speaker, which seemed to be corrupt. When listening to the affected audio files, it appeared as if they had been stretched. Other recordings from the same speaker were not affected. The Table 3 on the next page lists the affected clips.

Client ID of Affected Speaker	Affected Clip ID
bef06101-2800-4a5f-86e4-c169edf667a6	88503
bef06101-2800-4a5f-86e4-c169edf667a6	88505
bef06101-2800-4a5f-86e4-c169edf667a6	88506
bef06101-2800-4a5f-86e4-c169edf667a6	88504
bef06101-2800-4a5f-86e4-c169edf667a6	88513
bef06101-2800-4a5f-86e4-c169edf667a6	88507

Table 3: Corrupt Recordings within SDS-200 Corpus

Because the recordings appeared to be corrupt and they prevented an increase of the batch size, it has been decided to not to use them for this thesis.

3.2.5 Addition of Canton Group Label

The project thesis "Automatic Detection of Swiss German Dialects using Wav2Vec" [8] divided the SDS-200 corpus into four canton groups, which are shown in the Table 4 below.

Canton Group Name	Cantons
Highest-Alemannic (HA)	GL, NW, OW, SZ, UR, VS
Western-High-Alemannic (WA)	BE, BS, FR, SO
Central-High-Alemannic (CA)	AG, LU, ZG, ZH
Eastern-High-Alemannic (EA)	AI, AR, GR, SG SH, TG

Table 4: Canton Groups [8]

For reproducing some of the experiments from the predecessor thesis, the canton group label was added to both corpora. The map in Figure 7 on the next page visualises the four canton groups and the location of the samples using the same approach as in the Chapter 3.1.3 above.

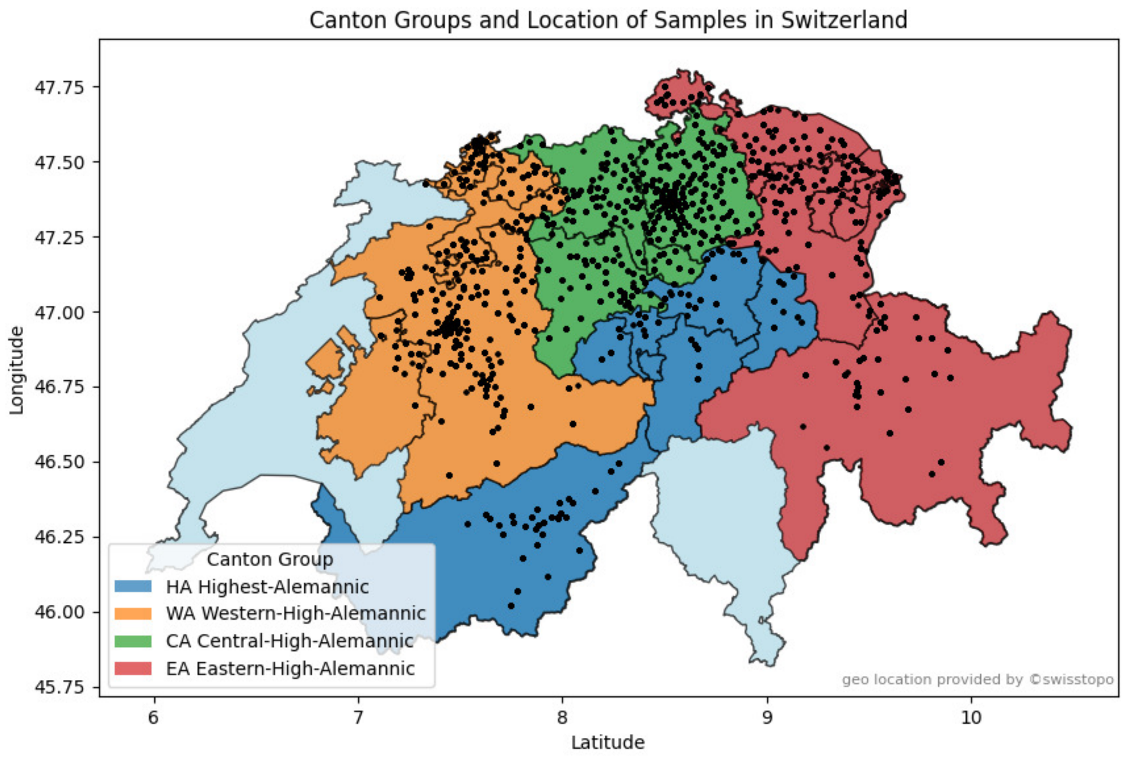

Canton Groups and Location of Samples in Switzerland
Longitude
47.75
47.50
47.25
47.00
46.75
46.50
46.25
46.00
45.75
6
7
8
9
10
Latitude
Canton Group
HA Highest-Alemannic
WA Western-High-Alemannic
CA Central-High-Alemannic
EA Eastern-High-Alemannic
geo location provided by ©swisstopo

Figure 7: Canton Groups and Location of Samples in Switzerland

Between these four canton groups the number of speakers is not balanced, as shown in the Figure 8 below.

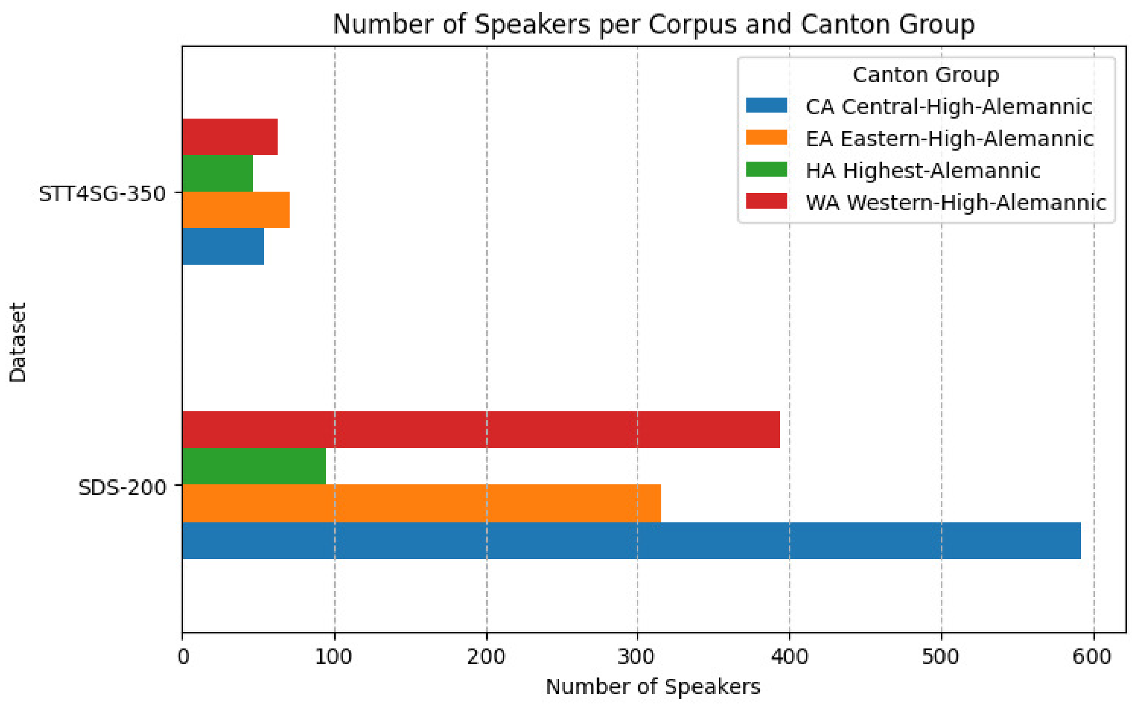


Figure 8: Number of Speakers per Corpus and Canton Group

The Figure 8 above visualises that the number of speakers per canton groups is more balanced in the STT4SG-350 than in the SDS-200 corpus, but the STT4SG-350 corpus contains less speakers compared to the SDS-200 corpus. For the Highest Alemannic (HA) group within the SDS-200 corpus, only a small number of speakers are available compared to the other canton groups.

3.2.6 Addition of Subregion Label to SDS-200 Corpus

The STT4SG-350 corpus introduced the subregion label [7]. Compared to the canton group label, which is separated into four classes, the subregion separates the samples into the seven classes Basel, Bern, Grisons, Central Switzerland, Eastern Switzerland, Valais, Zurich. The Figure 9 below visualises the distribution of the samples from the corpus SDS-200 and STT4SG-350 into the subregions.

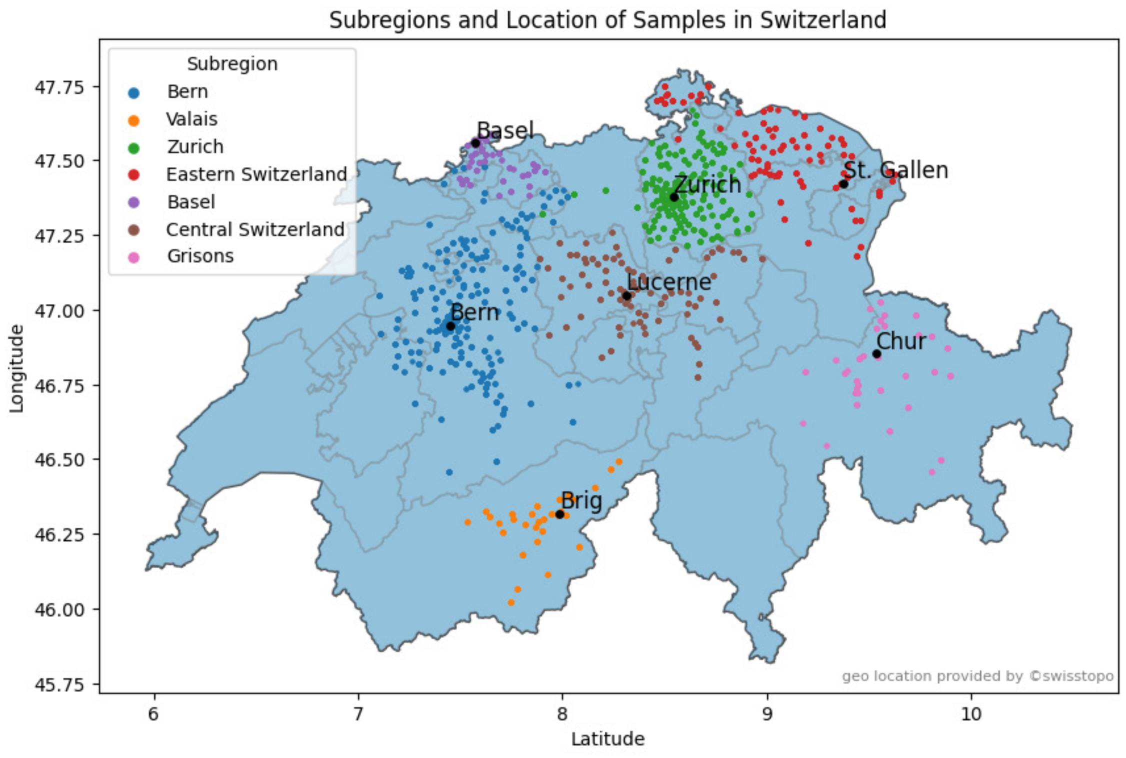


Figure 9: Subregions and Location of Samples in Switzerland

As the STT4SG-350 corpus does not divide these subregions based on fixed canton borders, it was not straightforward to map the subregion label to the SDS-200 corpus. In cases where all samples from the STT4SG-350 corpus were assigned to a single subregion within a canton, the corresponding samples from the SDS-200 corpus within the same canton were also mapped to that subregion. However, for cantons that contained samples from multiple subregions, the samples from the SDS-200 corpus within those cantons were dropped if there was no possibility of mapping them to a zip code from a sample in the STT4SG-350 corpus. As a result, a total of 14,772 samples from 256 speakers were dropped, which corresponds to approximately 4.7% of all samples in the SDS-200 corpus.

The Figure 10 below visualises the dropped samples and their location. The dropped samples are mainly from the cantons Aargau and St. Gallen.

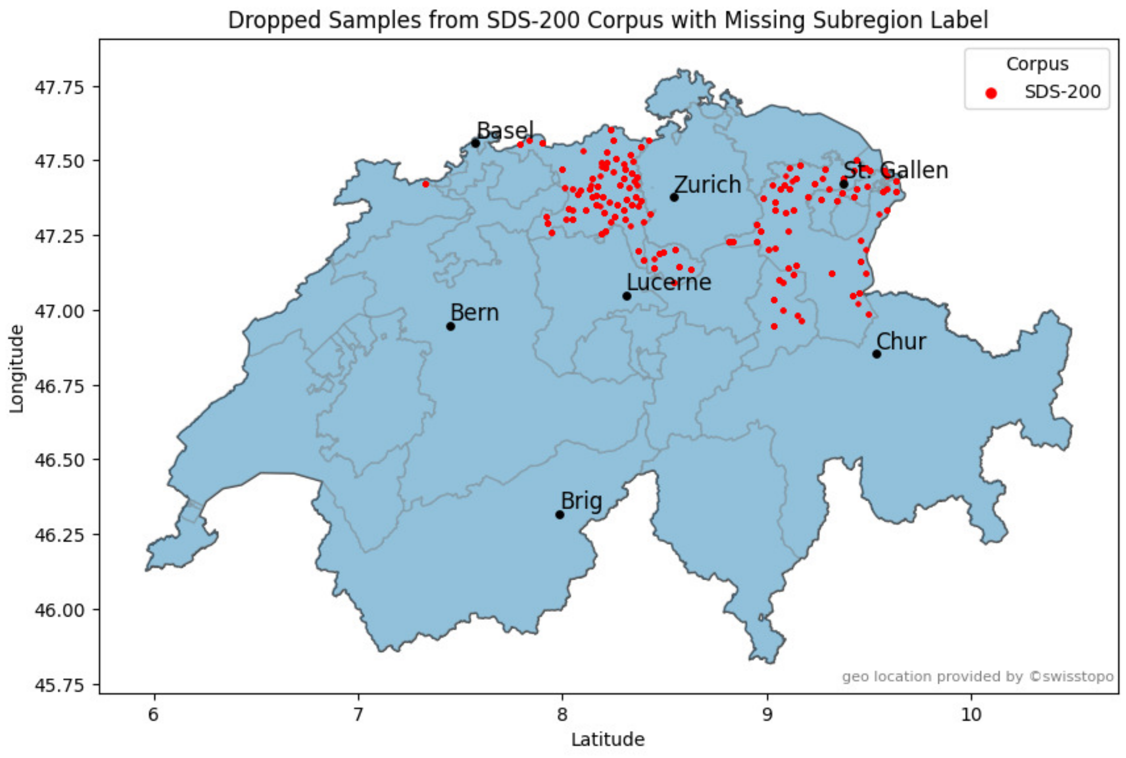


Figure 10: Dropped Samples from SDS-200 Corpus

During the initial pre-processing of the SDS-200 corpora, the geographical locations like longitude and latitude were not yet available. Consequently, it was not possible to categorise the regions based on their geographical location at that time. Upon later analysis, it became evident that a mapping based on the geographical location could have been performed. Fortunately, most of these samples would have been assigned to subregions such as Bern, Zurich and Eastern Switzerland, which have a high number of samples and speakers available compared to other subregions.

For the subregions, the number of speakers is well balanced in the STT4SG-350 corpus, but in the SDS-200 corpus, the subregions Zurich and Bern have significantly more speakers than other subregions such as Valais, Grisons and Basel. The Figure 11 on the next page visualises the number of speakers per corpus and subregion.

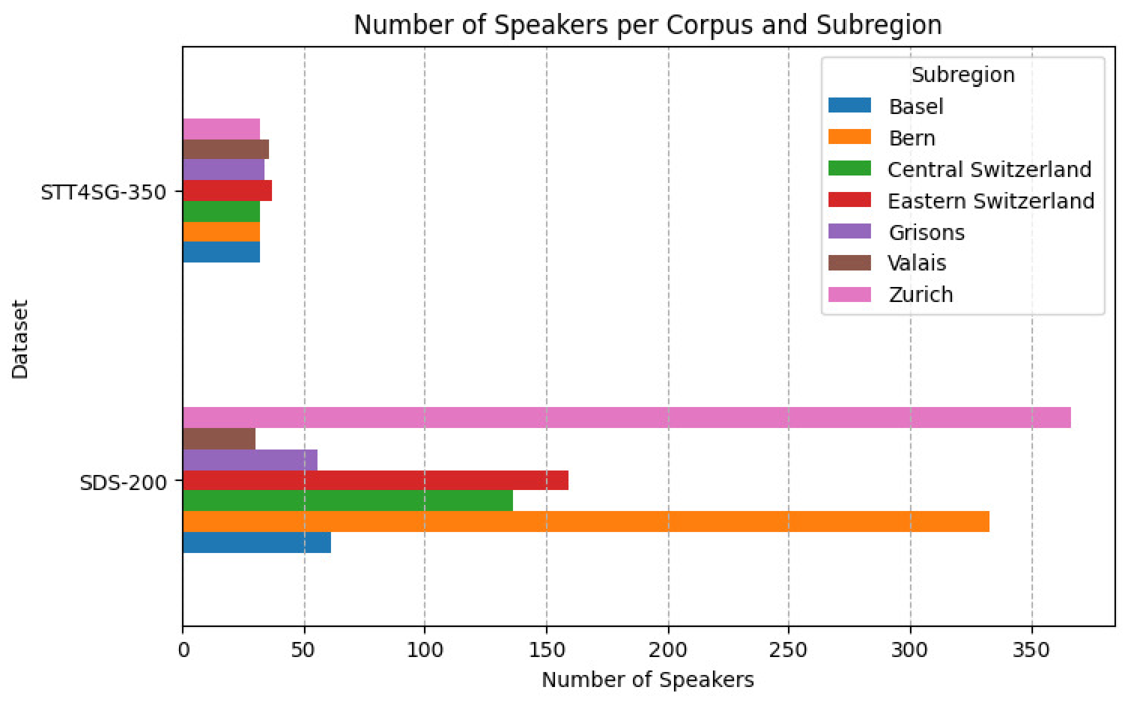


Figure 11: Number of Speakers per Corpus and Subregion

3.3 Train and Test Split

To evaluate the performance of a model it is necessary to use different samples for training and testing. To accomplish that, corpora are always split into a train and test dataset. Depending on the specific task, additional criteria need to be met. This chapter will describe how train and test splits were created for this thesis.

For classification tasks, the distribution of the samples from each class should be equal between the train and test dataset. For example, it is not desired to have 90% of the samples for one class in the test dataset and only 10% in the train dataset.

Additionally, for speech tasks it can be important to also separate the speakers. Samples of a speaker should then either be in the train or test dataset but not in both. If this separation is not done, then the performance of the model might be evaluated too high as the model learns characteristics of a speaker. In dialect identification this proved to be an important topic, which will be covered more deeply in the experiments section of this thesis.

For the STT4SG-350 corpus, a train-test split is already available which can be used [7]. In these splits the classes are distributed equally and the speakers are separated. For the SDS-200 corpus, as well as for the mixed corpora which will be introduced later in the Chapter 4.2 below, it was necessary to manually create a train and test split.

To create a train and test split which is both stratified and grouped, the samples are first grouped by the group attribute. After that, the created groups are randomly split according to a given ratio and then joined again with their samples. This results in a split where the samples in a group are guaranteed to be separated. The other criteria, such as the effective split ratio and class distribution, are then evaluated and a score is assigned to the created split. This is then repeated multiple times to create a vast variety of random splits with different scores. Finally, the best-scored split is chosen.

The above-described method was used to create all the splits used in this thesis. The client ID from the SDS-200 corpus and STT4SG-350 corpus was used as a group attribute and a split ratio of 80% for train, 10% for valid and 10% for test was used. An overview of all the created splits can be found in the Appendix 8.3 below and details like the number of samples contained in each split can be found in Appendix 8.4 below.

3.4 Model Architecture

All the used models follow the same generic architecture which was mostly inspired by previous bachelor's theses in the field of Swiss German dialect identification [9]. The main difference is that the implementation was completely revised using newer frameworks as described in Chapter 3.5 below.

Another small difference is the usage of a log-softmax activation function in the last layer of the classifier. Because of that difference the negative log-likelihood loss function was used instead of the cross-entropy loss function for training the model. For multi-class classification maximising the likelihood is the same as minimising the cross entropy and therefore those training objectives can be used interchangeably.

The model receives raw waveforms of speech (S), extracts features (F) and produces a probability distribution over the available classes to predict (C). For classification, the class with the maximum probability is selected. The Figure 12 below visualises the architecture of the models.

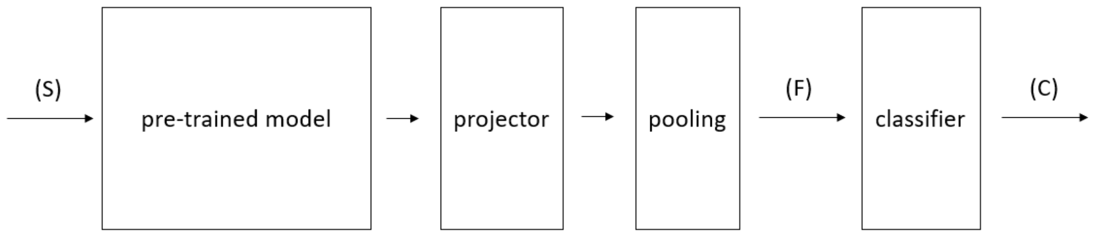


Figure 12: Model Architecture

The Figure 12 above visualises the modules from which the model is built. The first part is a pre-trained model such as wav2vec or Whisper which receives raw waveforms of speech (S). Afterwards a single-layer projector projects the output of the pre-trained models and then a mean pooling is applied to reduce the dimensionality. The result is a one-dimensional feature vector (F). This feature vector (F) is then fed into a single-layer neural classifier which uses softmax as its activation function to create a probability distribution over the classes.

3.5 Frameworks

The code for training and testing the models was written entirely in Python and uses the SpeechBrain framework which is a research-centric open-source all-in-one conversational AI toolkit based on PyTorch released in 2021 [42]. The decision was made to use the SpeechBrain framework because it provides various functionalities required for working with speech out-of-the-box. With the SpeechBrain framework the hyperparameters are conveniently managed in YAML files. This allows testing models in different configurations without any code changes. During the creation of this thesis, a major release[22] of the PyTorch framework containing the functionality to compile models to increase training and inference performance was published. By upgrading to this latest framework version and using the compilation for models, the training duration was decreased. The Weights & Biases[23] machine learning platform was used to monitor the training process and performance of the models.

3.6 Training Procedure

The used training procedure was mainly given by the SpeechBrain framework [42]. Training was done over multiple epochs. Each epoch had a training and a validation phase. During the training phase all the samples in the train dataset were fed through the neural network in batches and gradients were accumulated. After several gradients had been accumulated, the optimisers updated the parameters of the model according to the learning rate and optimiser settings. After the completion of one training epoch, the classes for all the samples in the validation dataset were predicted during a validation phase. Afterwards the performance metrics were computed and reported to the Weights & Biases machine learning platform.

[22] https://pytorch.org/blog/pytorch-2.0-release

[23] https://wandb.ai

3.7 Performance Metrics

The validated models are all multi-class classifiers, and therefore the traditional metrics for classification such as accuracy, precision, recall and F1 score, as explained in Chapter 2.4 above, were used to measure the performance of the models. For all the conducted experiments both the micro F1 and macro F1 scores were calculated using the classification report utility from the scikit-learn framework[24]. Because the aim of this thesis is to evaluate the model which performs best across all dialect regions, the macro F1 score was prioritised as it weights all classes equally. When models have been validated multiple times during training, the best performance metric was considered.

3.8 Training Parameters

While training parameters differ from experiment to experiment, certain parameters stayed mostly the same for all the conducted experiments. This chapter describes these parameters and explains how they were defined.

3.8.1 Optimiser and Learning Rate

All the experiments used the Adam optimiser [43]. While different optimisers can have a significant effect on the performance of a neural network, this effect was not explored in this thesis. The Adam optimiser was chosen as it is a common choice for deep learning with quick convergence and great robustness. This is because the Adam optimiser uses an adaptive learning rate, which means that the learning rates are adapted for individual parameters based on estimates. The specified learning rate is only used as an upper limit.

Apart from the used optimiser, the learning rate plays a significant role in the performance of a model. The learning rate is even described as "the single most important hyperparameter and one should always make sure that has been tuned" [44]. We decided to not play around with the learning rate too much for most experiments nevertheless, so that no unexpected side effects are introduced.

A common practice is the decay of the learning rate over the training period which can help in both optimisation and generalisation. One explanation for why learning rate decays are so effective is that an initial high learning rate allows the network to escape local minima and decreasing the learning rate over the training period helps to settle down and avoid oscillation [45]. It was therefore chosen to use linear learning rate decay by starting with a high initial learning rate in the first epoch and reducing the learning linearly after every epoch until the final low learning rate is reached in the last epoch.

[24] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

The pre-trained models require smaller learning rates, because they were already trained and just need to be fine-tuned. The other parts of our models however might benefit from a higher learning rate to achieve faster convergence. It was therefore decided to use two optimisers instead of one. The optimiser for the pre-trained model uses a significantly lower learning rate while the other optimiser for the rest of the model uses a higher learning rate.

For the pre-trained model, the initial learning rate was taken over from predecessor bachelor's thesis "Exploring Wav2Vec2 Pre-Training on Swiss German Dialects using Speech Translation and Classification" [9] from 2022. In that thesis the authors used a single optimiser with an initial learning rate of 3×10^{-5} and a linear decay to zero over 25 epochs. Instead of doing a linear decay to zero which seemed to be too extreme, a learning rate of 3×10^{-8} was chosen as the final learning rate for the linear decay for the experiments in this thesis.

For the rest of the model, the learning was chosen to be the highest stable learning rate possible. Because when using a learning rate decay, the learning rate only goes down during training, choosing the highest possible learning rate to start from seemed to be a reasonable choice. This evaluation was done experimentally by first starting with a high learning rate of 0.1 and then decreasing the learning rate until the loss becomes stable and is steadily decreasing. This resulted in the learning rate of 1×10^{-3} as the initial learning rate. To have a similar decay to the pre-trained model, the final learning rate for the decay was chosen to be 1×10^{-6}.

To summarise, for most of our experiments, two Adam optimisers configured to the default parameters were used. With the default parameters being a β_1 of 0.9, a β_2 of 0.999, an ε of 1×10^{-8} and a λ of 0. The first optimiser was dedicated to the pre-trained model and set to an initial learning rate of 3×10^{-5} with a linear decay to 3×10^{-8} over the epochs. The second optimiser was used for the projector and classifier and set to an initial learning rate of 1×10^{-3} with a linear decay to 1×10^{-6} over the epochs. The specific learning rates used for each experiment can be found in the Appendix 8.2 below.

3.8.2 Batch Size and Gradient Accumulation

In deep learning, samples are fed through the neural network in batches. The batch sizes can have a big impact on the performance of a neural network. While larger batches significantly reduce the training duration at the cost of more computing resources, they also lead to poor generalisation [46].

The maximum achievable batch size depends on the size of the model and the available computing resources. By using the training infrastructure described in the Chapter 3.9 below, it was not always possible to reach the desired batch size, because often not enough memory was available. However, by using gradient accumulation, it is possible to train with bigger batch sizes than would fit into memory. Instead of stepping the optimiser on every batch, the gradients of multiple batches are accumulated, and then the optimiser steps once for all the accumulated batches.

For most experiments in this thesis, a low batch size of either 32 or 48 was chosen. Some experiments also use larger batch sizes such as 64 or even 256. The specific batch sizes used for each experiment can be found in the Appendix 8.2 below.

3.9 Training Infrastructure

This chapter describes the various infrastructures used to perform the experiments in this thesis. Like previous bachelor's theses at the ZHAW, the official APU infrastructure of the ZHAW was used for this thesis. This was a virtual machine on the OpenStack cluster from the Institute of Applied Information Technology, which was equipped with one NVIDIA Tesla T4 16 GB graphics card. The performance of the graphics card, especially the available memory, proved to be insufficient for training larger models in a reasonable time. Because of that, the APU infrastructure was mainly used for smaller models.

For larger models, a higher performing infrastructure was needed to reduce the training time. While NVIDIA A100 graphics cards are available to the Centre for Artificial Intelligence at ZHAW, they were not allowed to be used for this thesis because of the high demand for those resources and their strict access restrictions.

In search for an alternative, different cloud providers such as Amazon AWS[25], Google Cloud[26], Microsoft Azure[27], Paperspace[28], and Vast.ai[29] were evaluated. Predominantly because of the low price and easy access, Vast.ai was selected.

Vast.ai is an open GPU market and, according to his own advertisement, the market leader in low-cost cloud GPU rental. In contrast to the other cloud providers, Vast.ai offers consumer grade GPUs such as the NVIDIA GeForce series which are well performing and much lower in price than data centre grade graphics cards such as the NVIDIA A100. The reason cloud providers do not offer these graphics cards is because the NVIDIA GeForce Driver EULA[30] explicitly prohibits the deployment of the driver software to data centres. Since Vast.ai is an open market for individuals where everyone can rent out their personal graphics cards at home and those renting the graphics cards must bring their own drivers and software, this EULA restriction does not apply to graphics cards rented from Vast.ai.

For bigger experiments, an instance with one or two NVIDIA GeForce RTX 3090 24GB graphics cards was rented on Vast.ai. For the deployment of the instances a Docker image was created which contains all the required software components. To setup the instances, the training and testing code was cloned from the source code repository and the training data was transferred over SSH from the APU server to

[25] https://docs.aws.amazon.com/dlami/latest/devguide/gpu.html

[26] https://cloud.google.com/gpu

[27] https://learn.microsoft.com/en-us/azure/virtual-machines/sizes-gpu

[28] https://www.paperspace.com/gpu-cloud

[29] https://vast.ai

[30] https://www.nvidia.com/en-us/drivers/geforce-license

the Vast.ai instance. Because an instance with two graphics card is cheaper than two instances with one graphics card, an instance with two graphics cards was preferred to run two experiments in parallel. Conveniently, this also reduced the required setup time.

To use as much of the available computing time, experiments were queued on the servers. This allows an experiment to start as soon as another one finishes and was accomplished using the stuff command of the Linux utility screen[31].

[31] https://www.gnu.org/software/screen/manual/screen.html#Paste

4 Experiments

In this chapter the conducted experiments are presented. Each experiment has its own dedicated chapter in which the applied methods as well as the obtained results are summarised.

Throughout this chapter, runs are referenced by their unique number denoted by an "R". A detailed overview of all the experiments including the used parameters and corpora can be found in the Appendix 8.2 below.

All the used corpora were split into a train, validation and test dataset. When corpora are referenced as the train or validation dataset of a run, it is always referenced to the corresponding train or validation dataset and not to the entire corpus. A dataflow diagram giving an overview of all the used corpora and where their data is coming from is contained in the Appendix 8.3 below. Additionally, detailed numbers on the data contained within the used corpora and their splits can be found in the Appendix 8.4 below.

4.1 Exploring Models and Corpora

The first experiment explored the capabilities of the available resources such as pre-trained models and corpora for Swiss German dialect identification. The goal of the experiment was to get an impression of the performance which can be achieved by using the available pre-trained models and corpora.

4.1.1 Experimental Setup

For the experiment multiple deep neural networks were trained and validated using the model architecture explained in Chapter 3.4 above. The Table 5 below contains the performed runs in this experiment.

Run	Model		Train	Dataset	
ID	Pre-Trained	Label	Epochs	Train	Valid
R01	wav2vec2-xls-r-300m	canton group	3	SDS-200	SDS-200
R02	wav2vec2-xls-r-300m	canton group	3	STT4SG-350	STT4SG-350
R03	wav2vec2-xls-r-300m	subregion	3	SDS-200	SDS-200
R04	wav2vec2-xls-r-300m	subregion	3	STT4SG-350	STT4SG-350
R05	whisper-small	canton group	3	SDS-200	SDS-200
R06	whisper-small	canton group	3	STT4SG-350	STT4SG-350
R07	whisper-small	subregion	3	SDS-200	SDS-200
R08	whisper-small	subregion	3	STT4SG-350	STT4SG-350
R09	whisper-base	subregion	3	SDS-200	SDS-200
R10	wav2vec2-xls-r-300m	subregion	20	SDS-200	SDS-200
R11	wav2vec2-xls-r-300m	subregion	20	STT4SG-350	STT4SG-350

Table 5: Models and Corpora Runs

All the above runs used the same optimiser and learning rates as defined in Chapter 3.8.1 above. The batch size of 48 was chosen for all the runs. More details on the exact parameters used can be found in the Appendix 8.2 below.

For the runs R05-08 in the Table 5 above, the whisper-small model was chosen because with 244 million parameters it features a comparable number of parameters compared to the wav2vec2-xls-r-300m model which has 300 million parameters. This allowed for fairer comparison between runs R01-04 and runs R05-08 when comparing wav2vec to Whisper.

To evaluate which performance can be achieved using smaller models, the classification to subregions on the SDS-200 corpus was also performed using the whisper-base model which features only 74 million parameters compared to the 244 million parameters of the whisper-small model. The corresponding run is run R09 in the Table 5 above.

The models were trained and validated on both the SDS-200 and STT4SG-350 corpus. Both corpora were pre-processed according to the data pre-processing steps described in Chapter 3.2 above. The SDS-200 was split using the train-test-split method described in Chapter 3.3 above. For the STT4SG-350, no additional splits had to be created since a train and validation dataset already exists. The Figure 13 below visualises the dataflow of the corpora.

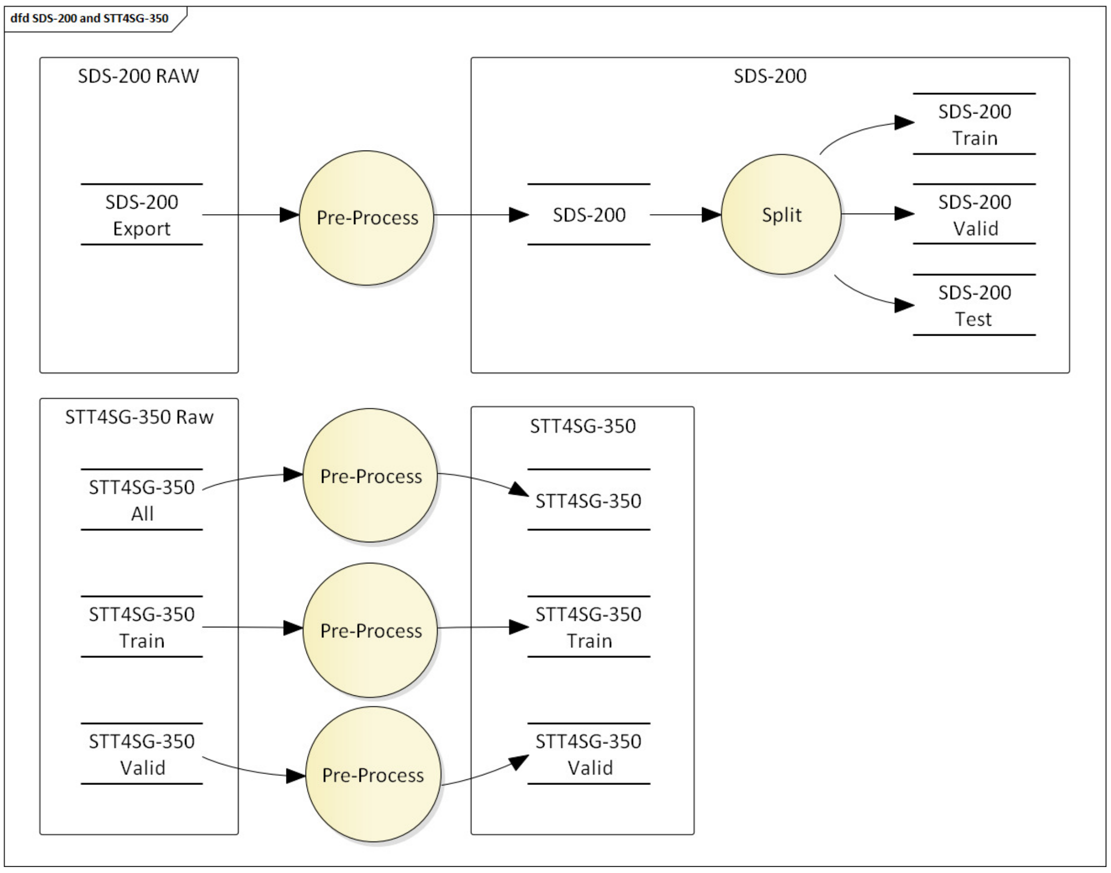


Figure 13: Dataflow of SDS-200 and STT4SG-350 Corpora

To have a low training duration, the training was done for only three epochs for most of the runs. Except for runs R10 and R11 which were trained for 20 epochs on the SDS-200 and STT4SG-350 corpus using the wav2vec2-xls-r-300m model. This was done to see if a higher performance can be achieved by training more than three epochs. The corresponding runs are R10 and R11 in the Table 5 above.

4.1.2 Results

Instead of presenting all the results consolidated into a single table, the result chapter is split into multiple subchapters which go into specific aspects of the obtained results. A complete list of all runs and their results can be found in the Appendix 8.2 below.

4.1.2.1 Comparison to Predecessor Theses

In a first comparison, the results on canton group classification on the SDS-200 corpus were compared to the results of the predecessor theses. The Table 6 below summarises the results of the corresponding runs from this experiment as well as the results of the project thesis “Automatic Detection of Swiss German Dialects using Wav2Vec” [8] from 2021 (below referenced to as PA-2021) and the bachelor’s thesis “Exploring Wav2Vec2 Pre-Training on Swiss German Dialects using Speech Translation and Classification” [9] from 2022 (below referenced to as BA-2022).

Run	Model	Performance	
ID	**Pre-Trained**	**Micro F1**	**Macro F1**
PA-2021 [8]	wav2vec2-xls-r-300m	52.89%	45.95%
BA-2022 [9]	CH-STT-FromPretrain-300M-Full-1[32]	52.45%	44.38%
R01	wav2vec2-xls-r-300m	**70.36%**	**49.84%**
R05	whisper-small	67.53%	49.75%

Table 6: Previous Theses Comparison

The classification to canton groups trained on the SDS-200 corpus achieved significantly higher F1 scores compared to the direct predecessor theses. An explanation on why the performance in our experiment was higher compared to the predecessor theses may be the use of a different optimisers and learning rates as explained in Chapter 3.8.1 above. This experiment was successfully able to reproduce the results of the project thesis “Automatic Detection of Swiss German Dialects using Wav2Vec” [8], which was in doubt by the authors.

[32] This is a wav2vec2-xls-r-300m model pre-trained on the SDS-200 corpus. More details can be found in the referenced bachelor’s thesis “Exploring Wav2Vec2 Pre-Training on Swiss German Dialects using Speech Translation and Classification” [9].

4.1.2.2 Comparison of Canton Group to Subregion Classification

Apart from the comparison to the previous theses, the classification to canton groups is not in scope of this thesis. However, the performed runs allowed to compare the performance in canton group classification to the one of subregion classification. The results of this are summarised in the Table 7 below.

Run	Model		Dataset		Performance
ID	Pre-Trained	Label	Train	Valid	Macro F1
R01	wav2vec2-xls-r-300m	canton group	SDS-200	SDS-200	**49.8%**
R03	wav2vec2-xls-r-300m	subregion	SDS-200	SDS-200	35.8%
R02	wav2vec2-xls-r-300m	canton group	STT4SG-350	STT4SG-350	**37.7%**
R04	wav2vec2-xls-r-300m	subregion	STT4SG-350	STT4SG-350	20.9%
R05	whisper-small	canton group	SDS-200	SDS-200	**49.7%**
R07	whisper-small	subregion	SDS-200	SDS-200	43.1%
R06	whisper-small	canton group	STT4SG-350	STT4SG-350	**37.9%**
R08	whisper-small	subregion	STT4SG-350	STT4SG-350	26.2%

Table 7: Canton Group to Subregion Classification Comparison

From the results, it can be observed that the classification to canton groups generally performs better than to subregions. This matches the general assumption that the higher the number of classes in a classification task, the harder the task.

4.1.2.3 Comparison of SDS-200 to STT4SG-350 Corpora

From the performed runs the performance of models trained on the SDS-200 corpus can be compared to the models trained on the STT4SG-350 corpus. Because the SDS-200 is not as balanced as the STT4SG-350 corpus, the macro F1 score is used for this comparison. The Table 8 below summarises the macro F1 scores of models trained and validated on the SDS-200 corpus against models trained and validated on the STT4SG-350 corpus.

Run	Model		Dataset		Performance
ID	Pre-Trained	Label	Train	Valid	Macro F1
R01	wav2vec2-xls-r-300m	canton group	SDS-200	SDS-200	**49.8%**
R02	wav2vec2-xls-r-300m	canton group	STT4SG-350	STT4SG-350	37.7%
R03	wav2vec2-xls-r-300m	subregion	SDS-200	SDS-200	**35.8%**
R04	wav2vec2-xls-r-300m	subregion	STT4SG-350	STT4SG-350	20.9%
R05	whisper-small	canton group	SDS-200	SDS-200	**49.7%**
R06	whisper-small	canton group	STT4SG-350	STT4SG-350	37.9%
R07	whisper-small	subregion	SDS-200	SDS-200	**43.1%**
R08	whisper-small	subregion	STT4SG-350	STT4SG-350	26.2%

Table 8: SDS-200 to STT4SG-350 Corpora Comparison

As it can be seen from the results in the Table 8 above, the models trained on the SDS-200 corpus consistently perform better than the ones trained on the STT4SG-350 corpus. The models trained on the SDS-200 achieve significantly higher macro F1 scores than the ones trained on the STT4SG-350 corpus. Even though the

STT4SG-350 corpus contains more samples than the SDS-200 corpus. This is contradicting with the initial assumption that training on the STT4SG-350 corpus can achieve a higher performance than training on the SDS-200 corpus since it contains more samples and is balanced over all classes.

One possible explanation for this performance difference is that the samples in the validation dataset of the SDS-200 corpus are easier to predict than the ones of the STT4SG-350 corpus and therefor a higher performance can be achieved on that validation dataset. To investigate this further, the wav2vec-based runs R01-04 trained on the SDS-200 corpus were validated using the validation dataset of the STT4SG-350 corpus and vice-versa. The Table 9 below summarises the macro F1 scores from these cross validations.

Run	Model	Dataset		Performance
ID	Label	Train	Valid	Macro F1
R01	canton group	SDS-200	SDS-200	**49.8%**
R02	canton group	STT4SG-350	SDS-200	26.0%
R01	canton group	SDS-200	STT4SG-350	**38.0%**
R02	canton group	STT4SG-350	STT4SG-350	37.7%
R03	subregion	SDS-200	SDS-200	**35.8%**
R04	subregion	STT4SG-350	SDS-200	13.5%
R03	subregion	SDS-200	STT4SG-350	**29.0%**
R04	subregion	STT4SG-350	STT4SG-350	20.9%

Table 9: SDS-200 and STT4SG-350 Comparison using Cross Validation

From the above results, it can be observed that the macro F1 scores of the models trained on the STT4SG-350 corpus are lower compared to the models trained on the SDS-200 corpus when validating on both the SDS-200 and STT4SG-350 corpus. This confirms that the models trained on the STT4SG-350 corpus are performing worse than the ones trained on the SDS-200 corpus.

Another explanation for the performance difference between the SDS-200 and the STT4SG-350 corpus might be the difference in variety of speakers in the corpora. The SDS-200 corpus contains less samples, which are distributed over a larger number of speakers. The STT4SG-350 corpus on the other hand contains more samples, but they are distributed over a small number of speakers. This difference in distribution is visualised in the Figure 5 in Chapter 3.1.3 above.

We conjecture that the models trained on the STT4SG-350 corpus learn to recognise speakers instead of learning to identify dialects because there is not a big enough variety of speakers in the STT4SG-350 corpus to generalise the model. Further experiments in the Chapter 4.3.1 below investigate this.

4.1.2.4 Comparison of wav2vec to Whisper Models

When comparing the performance of wav2vec2-xls-r-300m to whisper-small it cannot be concluded whether one performs better than the other. The results of the runs comparing wav2vec2-xls-r-300m to whisper-small are summarised in the Table 10 below.

Run	Model		Dataset		Performance
ID	Pre-Trained	Label	Train	Valid	Macro F1
R01	wav2vec2-xls-r-300m	canton group	SDS-200	SDS-200	**49.8%**
R05	whisper-small	canton group	SDS-200	SDS-200	49.7%
R02	wav2vec2-xls-r-300m	canton group	STT4SG-350	STT4SG-350	37.7%
R06	whisper-small	canton group	STT4SG-350	STT4SG-350	**37.9%**
R03	wav2vec2-xls-r-300m	subregion	SDS-200	SDS-200	35.8%
R07	whisper-small	subregion	SDS-200	SDS-200	**43.1%**
R04	wav2vec2-xls-r-300m	subregion	STT4SG-350	STT4SG-350	20.9%
R08	whisper-small	subregion	STT4SG-350	STT4SG-350	**26.2%**

Table 10: wav2vec to Whisper Models Comparison

As it can be seen from the Table 10 above, in this specific experiment, the whisper-small performed better in most cases. Interestingly, on canton group classification the macro F1 scores were nearly identical for both models while on subregion classification the performance was considerably better for the whisper-small model.

In terms of training duration, training whisper-small models was noticeably slower than training wav2vec2-xls-r-300m models. The training durations of the runs are summarised in the Table 11 below.

Run		Model		Dataset	Training
ID	Machine	Pre-Trained	Label	Train	Duration
R01	VAST	wav2vec2-xls-r-300m	canton group	SDS-200	**0 days 4:16**
R05	VAST	whisper-small	canton group	SDS-200	0 days 06:59
R02	APU	wav2vec2-xls-r-300m	canton group	STT4SG-350	**1 days 08:47**
R06	APU	whisper-small	canton group	STT4SG-350	2 days 12:32
R03	VAST	wav2vec2-xls-r-300m	subregion	SDS-200	**0 days 03:39**
R07	VAST	whisper-small	subregion	SDS-200	0 days 05:58
R04	APU	wav2vec2-xls-r-300m	subregion	STT4SG-350	**1 days 08:38**
R08	APU	whisper-small	subregion	STT4SG-350	2 days 13:55

Table 11: wav2vec to Whisper Models Training Duration Comparison

As it can be seen from the training durations in the Table 11 above, wav2vec2-xls-r-300m models were consistently training faster. For the models trained on the APU instance, which is equipped with a NVIDIA Tesla T4 16GB graphics card, the training took an additional day for the whisper-small model.

The performance difference might be due to the additional mel spectrogram conversion required for Whisper. To reduce the training duration with Whisper models, an attempt was made to do the mel spectrogram conversion as a pre-processing step and not during training. But this would have required more than 300 gigabytes of disk space which was not available during this thesis.

4.1.2.5 Comparison to Smaller Whisper Model

To confirm that a comparable performance can be achieved using the smaller model, the classification to subregion was also performed on the SDS-200 corpus using the whisper-base model which has only 74 million parameters. The Table 12 below summarises the training duration of the runs which were all performed on an Vast.ai instance equipped with an NVIDIA RTX 3090 24GB graphics card and ran for 3 epochs.

Run	Model		Performance		Training
ID	**Pre-Trained**	**Label**	**Micro F1**	**Macro F1**	**Duration**
R03	wav2vec2-xls-r-300m	subregion	63.1%	35.8%	0 days 03:39
R07	whisper-small	subregion	**69.3%**	**43.1%**	0 days 05:58
R09	whisper-base	subregion	55.0%	37.9%	**0 days 01:42**

Table 12: Smaller Whisper Model Comparison

As it can be seen from the Table 12 above, the F1 scores are lower when using the whisper-base model. However, the training duration was reduced from nearly 6 hours to 1h and 42 minutes while still achieving a comparable performance.

It was found that using the whisper-base model is especially useful to reduce the training duration when testing out model configurations. Because of that most of the later experiments in this thesis use the whisper-base version of Whisper.

4.1.2.6 Comparison of Training Epoch Counts

To further validate the results, the training epoch count of the wav2vec2-xls-r-300m model was extended to 20 epochs. This resulted in higher micro F1 and macro F1 scores which are displayed in the Table 13 below.

Run	Model	Dataset		Train	Performance	
ID	**Pre-Trained**	**Train**	**Valid**	**Epochs**	**Micro F1**	**Macro F1**
R03	wav2vec2-xls-r-300m	SDS-200	SDS-200	5	63.1%	35.8%
R10	wav2vec2-xls-r-300m	SDS-200	SDS-200	20	**66.6%**	**42.0%**
R04	wav2vec2-xls-r-300m	STT4SG-350	STT4SG-350	5	23.9%	20.9%
R11	wav2vec2-xls-r-300m	STT4SG-350	STT4SG-350	20	**25.3%**	**23.2%**

Table 13: Training Epoch Counts Comparison

As it can be seen from the Table 13 above, the macro F1 scores of the runs training for 20 epochs were slightly higher compared to the runs with only 3 epochs. However, the highest validation micro F1 score was always reached at around 3 to 5 epochs. This indicates that training beyond 5 epochs is probably not beneficial. The higher learning rate and the use of multiple optimisers might explain why more epochs were not required compared to previous thesis which were doing 25 epochs [8], [9].

4.2 Exploring Mixed Corpora

In the previous experiment the performance which can be achieved on the existing SDS-200 and STT4SG-350 corpora was explored. This experiment explored the performance which can be achieved when both corpora are combined to bigger corpora. We anticipated that mixed corpora could benefit from both the high variety of speakers in the SDS-200 corpus and the high number of high-quality samples in the STT4SG-350 corpus, therefore providing a better train dataset which can achieve higher macro F1 scores.

4.2.1 Creation of the MIX-ALL Corpus

To create the mixed corpora, first the MIX-ALL corpus was created. It contains all the samples from both the SDS-200 and the STT4SG-350 corpus. All the samples were taken from the STT4SG-350 corpus and not only the balanced subset. The samples which have been removed as part of the pre-processing are not present in the MIX-ALL corpus. The MIX-ALL corpus served as a base for the creation of further mixed corpora. The Figure 14 below visualises this in a dataflow diagram.

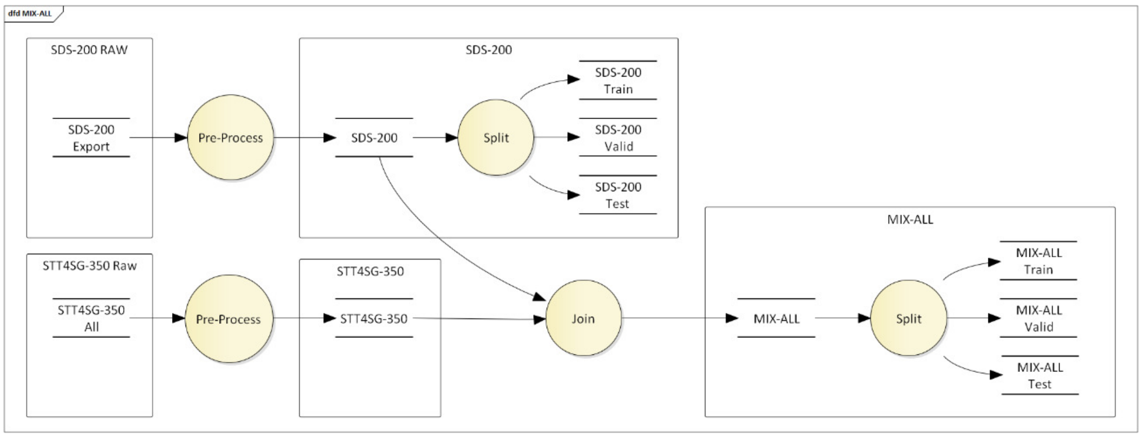


Figure 14: Dataflow of MIX-ALL Corpus

For the mixing, it was necessary to identify clips uniquely over both corpora. For this a new identifier was needed because the clip identifier which is already present in both corpora was not globally unique and therefore could not be used. The simplest approach to create a deterministic globally unique identifier is to use a hash function. To create the unique identifier the dataset name, the clip identifier as well as the path to the clip were concatenated and then hashed using the MD5 hash function. Afterwards, a sanity check was performed to ensure that no hash collisions occurred.

The MIX-ALL corpus contains in total 339,629 samples from 1,632 speakers. The subregion with the least number of samples in the MIX-ALL corpus is the subregion Grisons, which only has 29,717 samples compared to Zurich, which has 85,849 samples. The Table 14 on the next page summarises the total samples as well as the total speakers per subregion in the MIX-ALL corpus.

Subregion	Total Samples	Total Speakers
Grisons	29,717	90
Basel	33,973	93
Central Switzerland	39,644	168
Eastern Switzerland	40,572	196
Valais	44,628	66
Bern	50,474	365
Zurich	85,849	398

Table 14: Total Samples and Speakers per Subregion in MIX-ALL Corpus

4.2.2 Creation of Balanced Mixed Corpora

The distribution of samples to the subregions is not balanced in the MIX-ALL corpus. But for multi-class classification tasks an unbalanced corpus can lead to substantial performance decreases [47]. If the classes are not equally present in the train dataset, the model will not be able to perform the same for all classes. Because of that the MIX-ALL corpus was reduced into multiple balanced corpora that can be used for training.

The balanced corpora were defined by the number of samples per subregion. This resulted in perfectly balanced corpora. To create those corpora, a special selection process for the samples was used for each subregion. The goal of the selection process was to include the whole variety of speakers.

During the selection process samples for a subregion were chosen from each speaker in turns to fill up the corpus. In case no more samples from a speaker were left, the speaker was skipped. This procedure was repeated until the number of samples within the subregion reached the target size. This approach allowed all samples from speakers where only a few samples were available to be included. However, to maintain a balanced corpus, the corpus was filled with samples from speakers having many samples.

As a starting point, it was decided to create a MIX-20000 corpus with exactly 20,000 samples per subregion and a MIX-10000 with exactly 10,000 samples per subregion. In theory it would have been possible to create a perfectly balanced mixed corpus with up to 29,717 samples per subregion, but this possibility was not further explored in this thesis, because the MIX-20000 corpus already contains all the samples from most of the speakers and further increasing the corpus size would just have included more samples from speakers which are already over-present in the corpus.

The Figure 15 on the next page visualises the dataflow of the MIX-20000 and MIX-10000 corpora. The MIX-ALL corpus was reduced using the sample selection as described above to create a MIX-20000 and MIX-10000 corpora. Afterwards, those corpora were then split using the train-test-split method as described in Chapter 3.3 above.

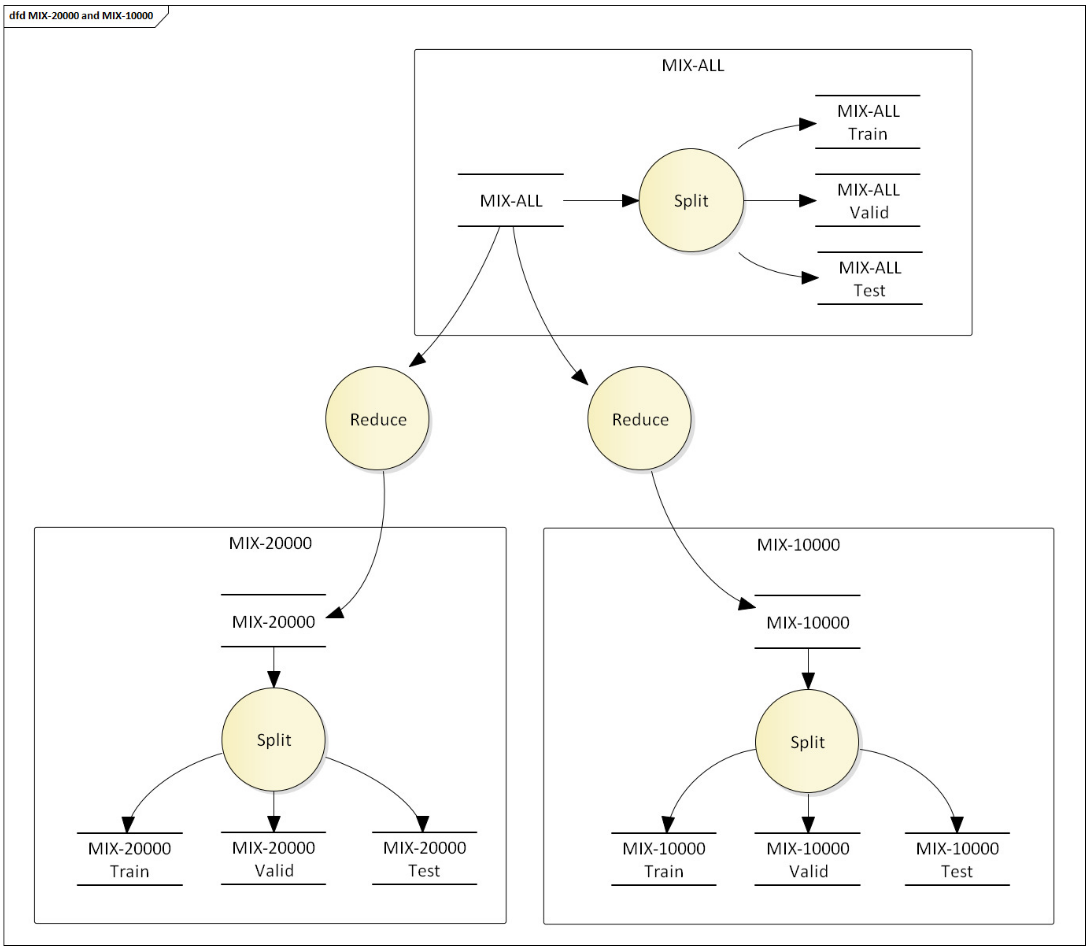


Figure 15: Dataflow of MIX-20000 and MIX-10000 Corpora

4.2.3 Experimental Setup

To explore the performance which can be achieved by using the mixed corpora, runs were performed on the corpora MIX-ALL, MIX-20000 and MIX-10000. The Table 15 below lists the runs of this experiment.

Run	**Model**	**Dataset**	
ID	**Pre-Trained**	**Train**	**Valid**
R12	whisper-base	MIX-ALL	MIX-ALL
R13	whisper-base	MIX-20000	MIX-20000
R14	whisper-base	MIX-10000	MIX-10000
R15	wav2vec2-xls-r-300m	MIX-20000	MIX-20000
R16	whisper-small	MIX-20000	MIX-20000

Table 15: Mixed Corpora Runs

The training was done for 20 epochs with a batch size of 48. All the above runs used the same optimisers and learning rates as defined in Chapter 3.8 above. More details on the exact parameters used can be found in the Appendix 8.2 below.

For runs R12-14 the whisper-base model was chosen since it provides reasonable performance with a low training duration. But to see which performance can be achieved with larger models, the training on the MIX-20000 corpus was also performed with the wav2vec2-xls-r-300m model in run R15 and with the whisper-small model in run R16.

4.2.4 Results

The results of the runs mixed corpora runs are summarised in the Table 16 below.

Run	**Model**	**Dataset**		**Performance**	
ID	**Pre-Trained**	**Train**	**Valid**	**Micro F1**	**Macro F1**
R12	whisper-base	MIX-ALL	MIX-ALL	**61.7%**	59.2%
R13	whisper-base	MIX-20000	MIX-20000	53.0%	51.1%
R14	whisper-base	MIX-10000	MIX-10000	61.4%	**60.8%**
R15	wav2vec2-xls-r-300m	MIX-20000	MIX-20000	54.6%	51.7%
R16	whisper-small	MIX-20000	MIX-20000	49.1%	47.6%

Table 16: Mixed Corpora Comparison

From the results in the Table 16 above, it can be observed that the model trained on the MIX-ALL corpus reached the highest micro F1 score. But because the MIX-ALL is not balanced, the macro F1 score needs to be prioritised. When it comes to macro F1 score the models trained on the MIX-ALL and MIX-20000 both achieved lower scores than the model trained on the MIX-10000 corpus even though the other corpora contain more samples. However, these results must be viewed with scepticism, as the scores are measured on different validation datasets. The samples in the validation dataset of the MIX-10000 might be easier to predict than the ones of the MIX-ALL and MIX-20000 corpus since they have been randomly selected during the test-train-split creation. Unfortunately, it is not possible to do a cross validation as it was done before with the SDS-200 and the STT4SG-350 corpus in Chapter 4.1.2.3 above, because the train and validation datasets of these corpora might contain overlapping speakers. We conjecture that the MIX-10000 corpus achieves higher macro F1 scores because it delivers the best samples per speaker ratio of all the tested corpora.

The larger wav2vec2-xls-r-300m and whisper-small models trained on the MIX-20000 reached lower macro F1 scores than the smaller whisper-base trained on the smaller MIX-10000 corpus. Interestingly, the larger whisper-small model trained on the MIX-20000 even performed worse than when using the smaller whisper-base model. In the end, this even was the worst performing model of this experiment.

4.3 Exploring Speaker Recognition

In the first experiment the existing SDS-200 and STT4SG-350 corpora and in the previous experiment the mixed corpora were explored. In both experiments training on larger corpora with more samples and especially more samples per speaker seemed to perform worse than using smaller corpora with fewer samples per speaker. We conjecture that this is because the models tend to recognise speakers instead of identifying dialects especially when presented with fewer speakers and more samples per speaker during training.

In this experiment our theory of speaker recognition was explored in more detail by constructing mixed corpora of increasing size and observing at which corpora size models trained on the corpora perform worse on dialect identification and better at speaker recognition. This would give an estimate on which corpora size and especially the number of samples per speaker which is optimal for dialect identification.

4.3.1 Reduction of the MIX-20000 Corpus

To test which number of samples per subregion gives the best trade-off between variety of speakers and total amount of samples, the train dataset of the MIX-20000 corpus was reduced to multiple smaller train datasets using the same method as it was used before in Chapter 4.2.2 above. These reduced corpora can then use the same validation dataset as the MIX-20000 corpus, which makes them easier to compare against each other. The Figure 16 below visualises the dataflow of the reduced corpora and from which data source they have been created.

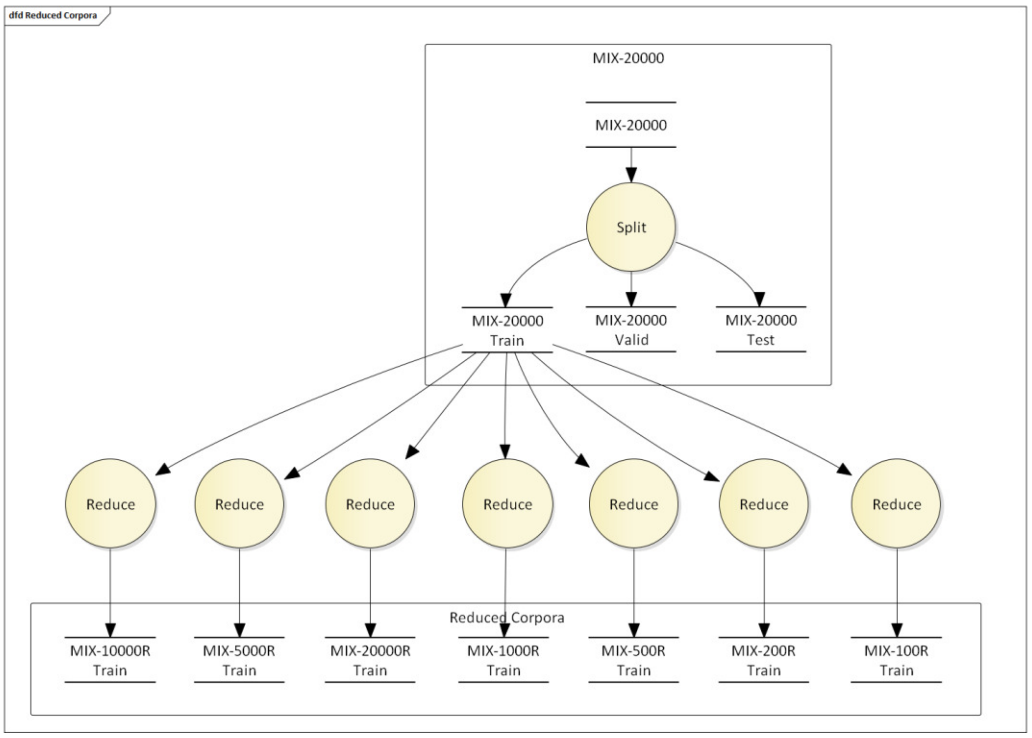


Figure 16: Dataflow of Reduced Corpora

The Table 17 below summarises the total samples and speakers as well as the mean samples per speaker of the MIX-20000 and the reduced corpora. The sizes of the reduces corpora were chosen so that the number of samples per subregion roughly doubled.

Corpus	Total Samples	Total Speakers	Mean Samples per Speaker
MIX-100R	700	649	1.08
MIX-200R	1,400	1,011	1.38
MIX-500R	3,500	1,376	2.54
MIX-1000R	7,000	1,376	5.09
MIX-2000R	14,000	1,376	10.17
MIX-5000R	35,000	1,376	25.44
MIX-10000R	70,000	1,376	50.87
MIX-20000	140,000	1,376	101.74

Table 17: Samples per Speaker of Reduced Corpora

As can be seen in the Table 17 above, all the available speakers were already included with 500 samples per subregion. After that only the number of total samples and the mean samples per speaker was increasing. The MIX-20000 contains an arithmetic mean of 101.74 samples per speaker.

The Table 17 above, however, does not illustrate the distribution of the samples to the speakers. To better visualise this and the trade-off between the variety of speakers to the total amount of samples, the Figure 17 on the next page visualises the number of samples per speaker within each subregion and each reduced corpus. It shows that the bigger the corpora, the smaller the number of speakers which contribute the majority of samples to the corpora. The detailed numbers can be found in the Appendix 8.5 below.

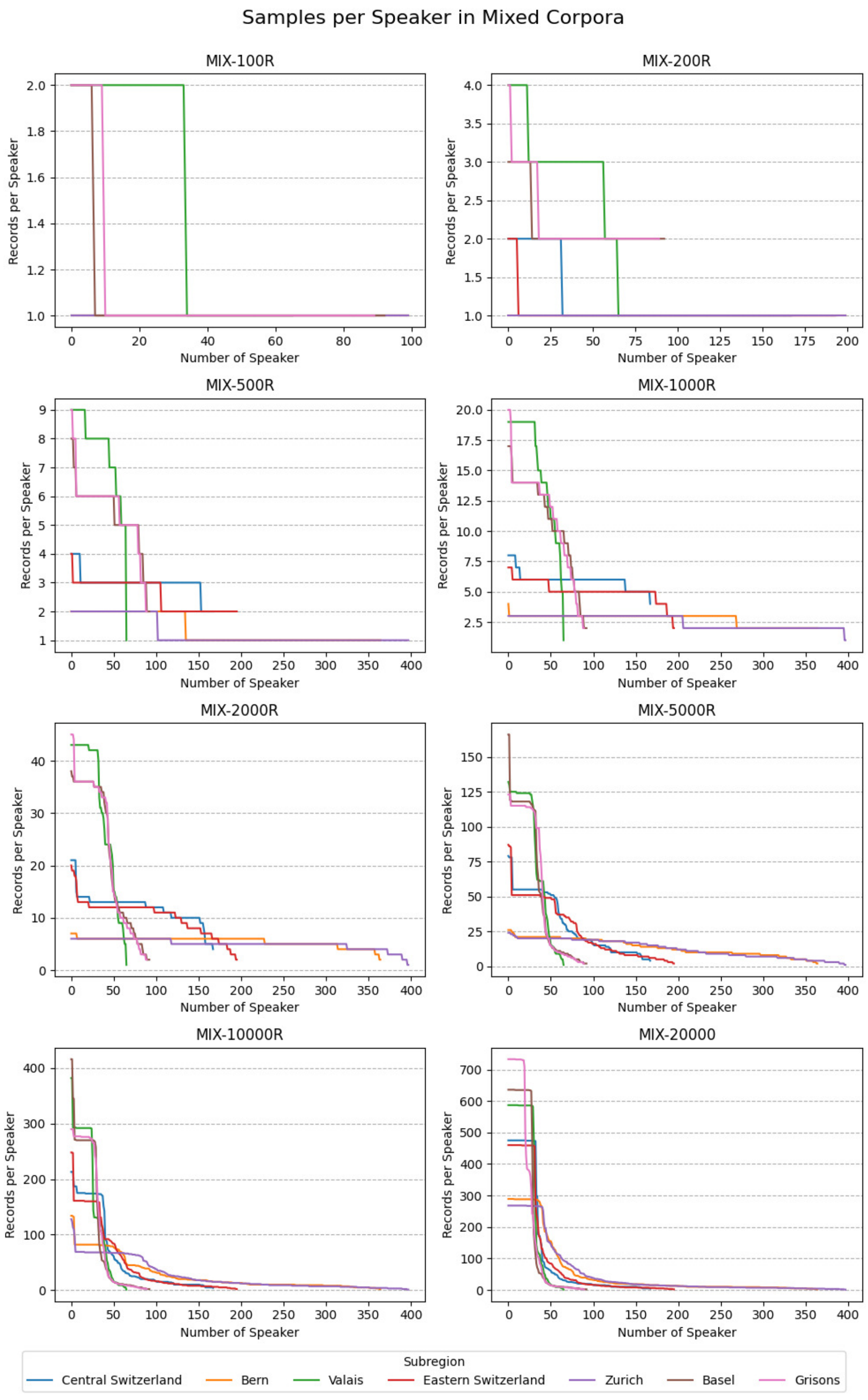


Figure 17: Samples per Speaker in Mixed Corpora

For example, in the MIX-20000 corpus most samples are from only around 50 speakers per subregion. For the subregion Grisons more than 730 samples were taken from 18 speakers. This means that for the subregion Grisons 65% of all samples are only from 18 speakers from the total of 89 speakers.

To evaluate the performance on speaker recognition, for every mixed corpus an additional test dataset was generated containing samples that do not occur in the train dataset but are from the same speakers as in the train dataset. With this test dataset it was possible to evaluate the capability of the models to recognise the speakers on which the models were trained on.

To create this test dataset, for every speaker in the reduced train dataset at maximum of five samples from the same speaker were chosen from the MIX-ALL corpus which were not included in the reduced train dataset.

4.3.2 Experimental Setup

To compare the performance between the reduced corpora a whisper-base model was trained on each reduced train dataset and then validated on the validation dataset of the MIX-20000 corpus. The Table 18 below lists these runs.

Run	**Model**	**Dataset**	
ID	**Pre-Trained**	**Train**	**Valid**
R17	whisper-base	MIX-100R	MIX-20000
R18	whisper-base	MIX-200R	MIX-20000
R19	whisper-base	MIX-500R	MIX-20000
R20	whisper-base	MIX-1000R	MIX-20000
R21	whisper-base	MIX-2000R	MIX-20000
R22	whisper-base	MIX-5000R	MIX-20000
R23	whisper-base	MIX-10000R	MIX-20000
R24	whisper-base	MIX-20000	MIX-20000

Table 18: Speaker Recognition Runs

The models were trained for 5 epochs and use a batch size of 32. The same optimisers and learning rates were used as explained in Chapter 3.8.1 above.

4.3.3 Results

The Figure 18 below visualises the micro F1 scores of the models on the validation dataset of the MIX-20000 corpus as well as on the corresponding test dataset with the same speakers as in the train dataset of each corpus.

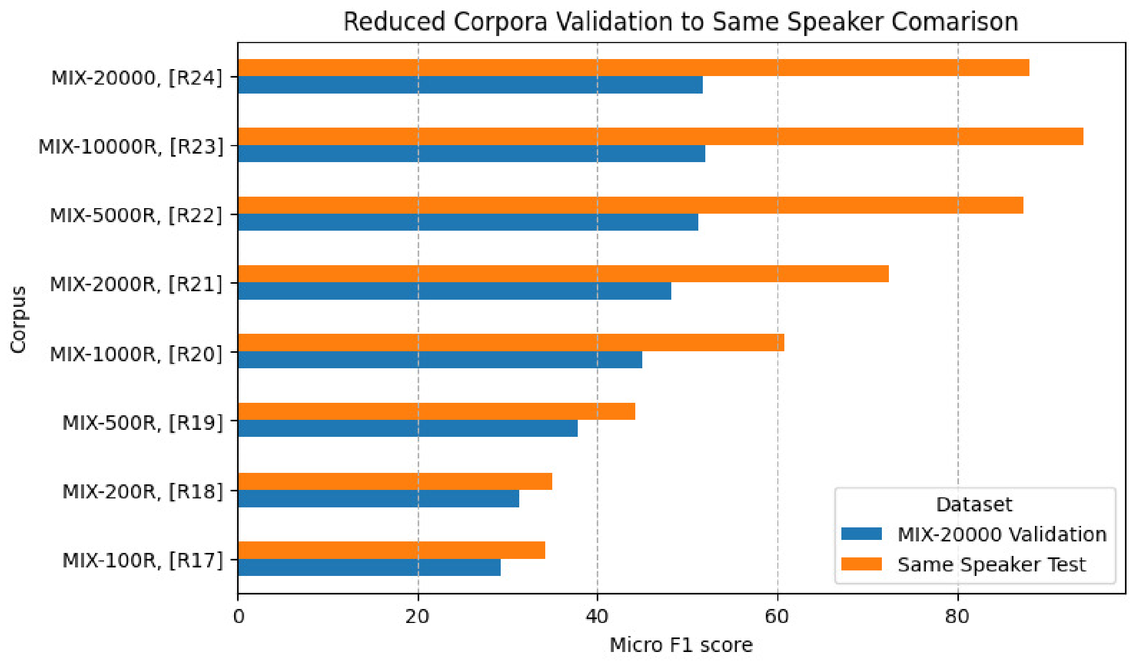


Figure 18: Reduced Corpora Validation to Same Speaker Comparison

The results show that the performance of the model was increasing with the corpus size but after 10,000 samples per class the performance was declining. These results confirm that the MIX-10000R corpus achieved the best performance over all dialect regions. We conjecture that this is because the MIX-10000R corpus has the best samples per speaker ratio of all the corpora.

Overall, the performance on the samples from the same speakers was very high and increasing with the corpus size. Because the performance was higher than the one on the validation dataset of the MIX-20000 corpus, it seems like the models had an advantage when classifying the same speakers. This is a strong indication that the model is learning to recognise speakers instead of identifying dialects.

4.4 Exploring Speech Augmentation

The previous experiment showed that when training with more samples per speaker, the models tend to recognise speakers instead of identifying dialects. In this experiment, various speech augmentation methods were explored on the mixed corpora. We anticipated that speech augmentation can help to overcome the problem of learning to recognise speakers instead of identifying dialects by artificially extending the variety of speakers and increasing model generalisation.

4.4.1 Augmentation Methods

For speech augmentation various well-known augmentation methods are available to choose from. The augmentations used in this experiment are all directly applied to the raw waveforms before feeding them through the network during training. During validation and testing no augmentation is applied.

By computing the augmentation right before feeding the speech signal through the network and because the used augmentation methods all make use of some sort of randomisation, multiple different versions of the same speech signal are processed by the network throughout the training. This essentially multiplies the available training samples every epoch.

4.4.1.1 SpecAugment

The SpeechBrain framework provides a time-domain approximation of the SpecAugment algorithm [17], [42]. It is called a time-domain approximation because originally the SpecAugment algorithm is applied to log mel spectrograms and not to signals in time domain. The used algorithm is therefore an approximation of the original SpecAugment algorithm which can be directly applied to raw waveforms. Whenever SpecAugment is mentioned as a used augmentation method in this thesis, it is always referenced to the time-domain implementation of the SpeechBrain framework [42].

The implementation of Speechbrain features the following augmentations:

1. **Perturb Speed:** Speed of speech is altered by resampling the speech signal to a slightly higher or lower sampling rate. The speech signal is randomly resampled to 80%, 90%, 100%, 110% or 120% of the original sampling rate [48].
2. **Drop Frequencies:** Frequencies are dropped from the speech signal by applying band-drop filters. Between 0 and 3 frequency bands are randomly removed from the speech signal.
3. **Drop Chunks:** Chunks of the audio signal are set to zero. Between 0 and 5 chunks of a length between 62.5ms and 125ms are randomly set to zero in the speech signal.

The time-domain approximation of the SpecAugment algorithm was chosen because it is a simple augmentation algorithm and was easy to adopt as it is part of the framework.

4.4.1.2 Speech Augmentation

The audiomentations[33] library provides a vast variety of audio augmentation methods which can be directly applied to raw waveforms. The library is easy to integrate and flexible. It allows the creation of custom audio augmentation pipelines and includes functionalities for randomising the applied augmentations.

The custom speech augmentation pipeline applies the following augmentations.

1. **Add Gaussian Noise[34]:** Gaussian noise is added to the speech signal with a random amplitude between 0.001 and 0.02.
2. **Stretch Time[35]:** The speed of the speech signal is changed without altering the pitch or its length. Non-silent parts of the speech signal stretched beyond the length are cut off. The stretch factor is randomly chosen between 0.8 and 1.3.
3. **Shift Pitch[36]:** The pitch of the speech signal is increased or decreased randomly between -4 and +4 semitones.
4. **Shift Signal[37]:** The speech signal is randomly shifted forwards or backwards between -50% and +50% of the signal's length. The part of the signal which is shifted off is rolled over to the other side of the signal.

Every step in the augmentation pipeline has a chance of 80% to be applied on a sample. This means that for 81.92% of the samples either 3 or 4 augmentations are applied, for 15.36% of the samples 2 augmentations are applied, for 2.56% of the samples only one augmentation is applied and for 0.16% of the samples no augmentation is applied.

The augmentations for the custom pipeline were chosen so that the dialect of the speaker is not altered. It is assumed that slower or faster speaking as well as speaking in higher or lower pitch does not change the dialect of a speaker. The gaussian noise as well as the shifting of the sample were chosen to make the model more robust.

[33] https://github.com/iver56/audiomentations

[34] https://iver56.github.io/audiomentations/waveform_transforms/add_gaussian_noise

[35] https://iver56.github.io/audiomentations/waveform_transforms/time_stretch

[36] https://iver56.github.io/audiomentations/waveform_transforms/pitch_shift

[37] https://iver56.github.io/audiomentations/waveform_transforms/shift

4.4.1.3 Speaker Mix

When working with variable length input such as speech signals it is often necessary to pad the signals for batching, because samples in the same batch need to fix into the same tensor. A common choice is to pad with zeros or white noise. Another option used for this speech augmentation is to pad the audio signal with other audio signals from the same dialect. Our assumption is that this approach helps to generalise the model and reduce the risk of speaker recognition because a sample can no longer be assigned to a single speaker.

The mixing of speakers was implemented as a generic module but was applied like other speech augmentations only during training before feeding the speech signals through the model.

The padding was done by filling up the speech signal using randomly selected chunks from other samples in the train dataset until a target length is reached. For Whisper models, a target length of 30 seconds was chosen because that is the speech signal length required by Whisper models [5]. For wav2vec models, the target length was set to 20 seconds because otherwise it would have required too much GPU memory. The samples for filling up the speech signal were chosen at random from the used train dataset but are from the same dialect subregion.

4.4.2 Extended Batches

Instead of applying the augmentation to the speech signal and then using the augmented signal during training, the batch can be extended with the augmented signal to create a larger batch. With this form of augmentation, the original speech signal is feed through the network together with the augmented signal. Effectively, this will double the amount of training data used in a single epoch.

4.4.3 Dropout Layers

With the addition of augmentation to the model, dropout layers were added since they can help to reduce the risk of overfitting [49]. The dropout layers were added after every module of the model. The Figure 19 on the next page visualises the model architecture as described in Chapter 3.4 above including the added dropout layers and their corresponding dropout rates.

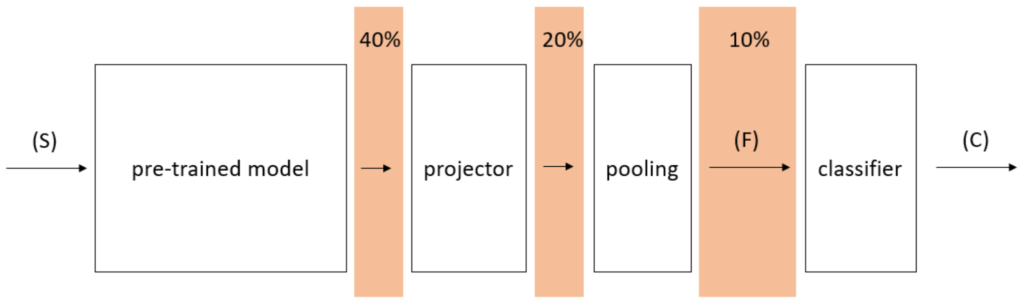


Figure 19: Drop Out Layers

The dropout rates were chosen to be decreasing from 40% to 10% with every layer. A dropout rate of 40% is not too aggressive and provides a reasonable starting point. In later layers of the neural network, the dropout rates were chosen to be lower since losing a neuron in those layers has a more significant effect on the final output of the network.

4.4.4 Experimental Setup

To evaluate the performance of the chosen augmentation methods, a whisper-base model was trained on the MIX-10000R train dataset and then validated on the MIX-20000 validation dataset for each augmentation method. The Table 19 below lists the performed runs in this experiment.

Run	**Model**			**Training**
ID	**Pre-Trained**	**Augmentation**	**Dropout**	**Batch Size**
R25	whisper-base	SpecAugment	YES	32
R26	whisper-base	Speech Augment	YES	32
R27	whisper-base	Speaker Mix (1s Chunks)	YES	32
R28	whisper-base	SpecAugment	YES	64
R29	whisper-base	Speech Augment	YES	64
R30	whisper-base	Speaker Mix (1s Chunks)	YES	64
R31	whisper-base	Speaker Mix (Full Length)	YES	64
R32	whisper-base	Combined Augmentations	YES	64
R33	whisper-base	None	YES	32
R34	whisper-base	SpecAugment	NO	32

Table 19: Speech Augmentation Runs

As in the previous experiment, the models were trained for 5 epoch and the same optimisers and learning rates were used as explained in Chapter 3.8.1 above.

To evaluate whether combining the original samples together with their augmented version in an extended batch helps to improve the generalisation of the models, the runs were performed once with a batch size of 32 in which the model is trained only with the augmented speech signal in runs R25-27 and a batch size of 64 in which both the original speech signal as well as the augmented signal are combined in an extended batch in runs R28-31.

In runs R27 and R30 the speaker mix augmentation was applied by filling up the speech signal with random 1 second chunks as explained in the Chapter 4.4.1.3 above. In the run R31, instead of only using 1 second chunks, the full length was used to fill up the speech signal.

Run R32 combines all the augmentation methods. First, the SpecAugment augmentation method was applied to the original speech signal and then used in-place of the original speech signal in the batch. Then the speech augmentation, as described in Chapter 4.4.1.2 above, was applied to the original speech signal followed by the speaker mix augmentation to fill up the augmented speech signal and then this speech signal was added to the batch. More details on why this approach might be helpful will be given in the results chapter below.

The last two runs R33 and R34 were performed to verify the effect of the added dropout layers. Run R33 used no augmentation but has dropout layers and run R34 used the SpecAugment augmentation but does not have any dropout layers.

4.4.5 Results

The Table 20 below summarises the results of the runs in this experiment. Additionally, the run R23 from the previous experiment is also included as a baseline since it uses the same parameters, but without augmentation and dropout layers.

Run	Model		Training	Performance	
ID	Augmentation	Dropout	Batch Size	Micro F1	Macro F1
R23	None	NO	32	52.1%	50.5%
R25	SpecAugment	YES	32	**57.1%**	**54.6%**
R26	Speech Augment	YES	32	52.4%	48.9%
R27	Speaker Mix (1s Chunks)	YES	32	35.9%	30.6%
R28	SpecAugment	YES	64	55.4%	52.6%
R29	Speech Augment	YES	64	55.4%	53.4%
R30	Speaker Mix (1s Chunks)	YES	64	52.7%	50.9%
R31	Speaker Mix (Full Length)	YES	64	53.7%	50.5%
R32	Combined Augmentations	YES	64	**60.1%**	**57.5%**

Table 20: Speech Augmentation Comparison

For the runs R25-R27 without extended batches, the SpecAugment augmentation method achieved the highest F1 scores. The other augmentation methods both performed worse than the baseline which did not use any augmentation or dropout layers.

Contrary to the assumption that mixing speakers together in a sample would result in improved generalisation, the speaker mix augmentation method did not perform very well in run R27, achieving the lowest F1 scores of all the augmentation methods.

Interestingly, the usage of extended batches decreased the performance on the SpecAugment augmentation method. An explanation for this might be that the SpecAugment augmentation method does not hide characteristics of a speaker enough, so the variety of speakers does not increase but the amount of samples doubles. This probably results in the model learning again to recognise speakers instead of identifying dialects.

When using extended batches in runs R29-31, the results show an improvement in F1 scores for both the speech and speaker mix augmentation methods. By using extended batches both models were able to perform better than the baseline.

The usage of extended batches significantly increased the F1 scores on the speaker mix augmentation method in the runs R30-31. The assumption is that this is because when using the mix augmentation method, the model is always trained on fixed-length input and then performs not well during validation on variable-length input. And by using extended batches, the model also trains on the original variable-length samples.

The mix augmentation was tested with randomly selected 1 second chunks as well as with the full speech signal. With 1 second chunks the model achieved a higher macro F1 score whereas with the full speech signal the model achieved a higher micro F1 score.

Since the SpecAugment augmentation method does seem to give better results when used instead of the original samples and the other speech augmentations performed better when using them in addition to the original samples. Combining both augmentations seemed to be a reasonable strategy since the effects might be combined. Additionally, the speaker mix can also be applied on top of the other augmentation methods. The results of this strategy can be seen in run R32 which performed best compared to all the other augmentation methods. This shows that the effects of the SpecAugment augmentation method can be combined with other speech augmentation methods for better performance.

The runs R33 and R34 show the effect of the added dropout layers. The results of those runs including the results of the baseline run R23 and the run R25 are displayed in the Table 21 below.

Run	**Model**		**Training**	**Performance**	
ID	**Augmentation**	**Dropout**	**Batch Size**	**Micro F1**	**Macro F1**
R23	None	NO	32	52.1%	50.5%
R33	None	YES	32	50.5%	47.3%
R34	SpecAugment	NO	32	**61.6%**	**59.5%**
R25	SpecAugment	YES	32	57.1%	54.6%

Table 21: Effect of Dropout Layers

Contrary to the initial assumption, the added dropout layers harm the generalisation of the models. As it can be seen from the results in the Table 21 above, the run R33 with no augmentation but added dropout layers performed worse than the baseline run R23 without augmentation and dropout layers.

The same effect can be seen when comparing the runs R34 and R25 in the Table 21 above. The run R25 with augmentation and dropout layers performed worse than the run R34 with augmentation but without dropout layers.

To summarise, from the runs performed in this experiment, the run R34 with the SpecAugment algorithm and without dropout layers seemed to perform best. However, the run R32 which used SpecAugment in combination with other augmentation methods as an extended batch seemed to be a promising strategy which could even perform better when trained without dropout layers.

4.5 Exploring Contrastive Learning

The previous experiment explored speech augmentation as a method for preventing models to recognise speakers instead of identifying dialects. In this experiment supervised contrastive learning was explored as another option to prevent speaker recognition. The idea was to learn useful representations of speech that capture dialect specific features by using supervised contrastive learning. These representations were then used to train a classifier for the dialect regions. We anticipated that by using contrastive learning the models learn more to identify dialects instead of recognising speakers.

Other applications of contrastive learning already proved to be successful, especially in computer vision for image classification [35]. In the paper "Contrastive Learning of General-Purpose Audio Representations" [50], contrastive learning shows promising results for learning audio representations. As outlined in Chapter 2.3.3 above, wav2vec is also trained using a contrastive learning approach.

For representation learning, the same model architecture was used as described in Chapter 3.4 above. The training was split into two phases. In the first phase, the pre-trained model together with the projector layer were trained to project speech into a latent feature space. During training, the model received pairs of speech samples which were either specified to be similar or dissimilar. Both samples were then fed through the neural network in parallel to produce two feature vectors. This setup is also known as Siamese networks. The training objective was then to optimise the cosine similarity of feature vectors from similar pairs and reduce the cosine similarity of feature vectors from dissimilar pairs. To accomplish this, the cosine embedding loss function was used as described in Chapter 2.2.3 above. This learning method is also known as contrastive learning. In the second phase, the model was trained with an added classification head to classify the learned representations into the seven dialect subregions.

To summarise, in the first phase of the training, contrastive learning was used to train the pre-trained model and the projector. In the second phase, only the classifier was trained, and the pre-trained model as well as the projector were frozen.

4.5.1 Selection of Sample Pairs

One benefit of contrastive learning is that the latent feature space can be controlled by the selection of sample pairs. The samples which are selected to be either similar or dissimilar have a major effect on the learned representations. It is therefore important that pairs are carefully selected for learning the desired representations. In this thesis, two strategies for pair selection were tested.

4.5.1.1 Speaker Separation Strategy

The focus of the first strategy was to prevent the model from learning to detect speakers. For every sample, one sample from the same speaker was chosen to form a dissimilar pair and another sample with the same dialect subregion but from a different speaker was chosen to form a similar pair. The idea behind choosing samples from the same speaker as dissimilar pairs was to prevent the network from learning representations which help to recognise speakers. The network should not be rewarded for recognising the same speaker.

The Figure 20 below visualises five samples in the latent feature space. Samples with the same colour are from the same speaker and samples in the same circle are from the same dialect subregion. The two shades of blue represent different speakers.

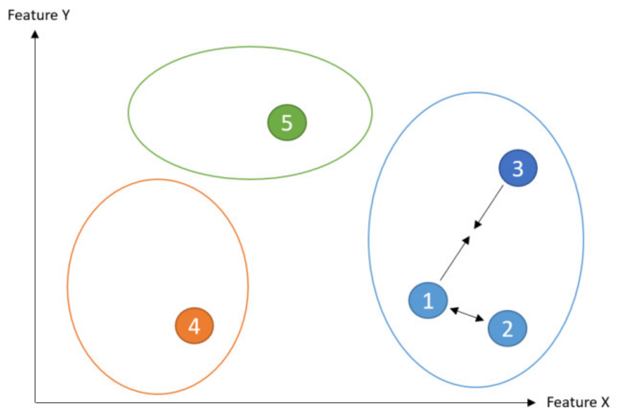


Figure 20: Speaker Separation Strategy

For the sample 1 in the Figure 20 above, two pairs were formed. Samples 1 and 3 form a similar pair since they are from different speakers but in the same dialect subregion, and sample 1 and 2 formed a dissimilar pair since they are from the same speaker. The idea was that the similar pair will be pulled closer in the latent feature space, which is represented by arrows pointing to each other, and the dissimilar pair will be pushed apart in the latent feature space, which is represented by arrows pointing away from each other.

4.5.1.2 Dialect Separation Strategy

In the first strategy, no dissimilar pairs with different dialects were used. To allow the model to learn dissimilarities between dialects, dissimilar pairs from different dialects were formed in the dialect separation strategy.

Additionally, the speaker separation might have caused a conflicting training objective to the clustering of dialects. A speaker is associated with a single dialect and therefore, those samples should be close together in the latent feature space. But by adding samples of the same speaker as dissimilar pairs, they will be pushed apart from each other in the latent feature space even though they are from the same dialect and should therefore be somehow close together in latent feature space. To simplify the training, those dissimilar pairs were left out with this strategy.

With the dialect separation strategy, for every sample, a similar pair was formed with another sample from the same dialect subregion but from a different speaker and for every other dialect subregion a sample was selected to form dissimilar pairs.

The Figure 21 below is an updated version of Figure 20 above. It visualises the same similar pair of samples 1 and 3 as before, but new dissimilar pairs for the other dialect subregions between sample 1 and 5 and 1 and 4 were formed.

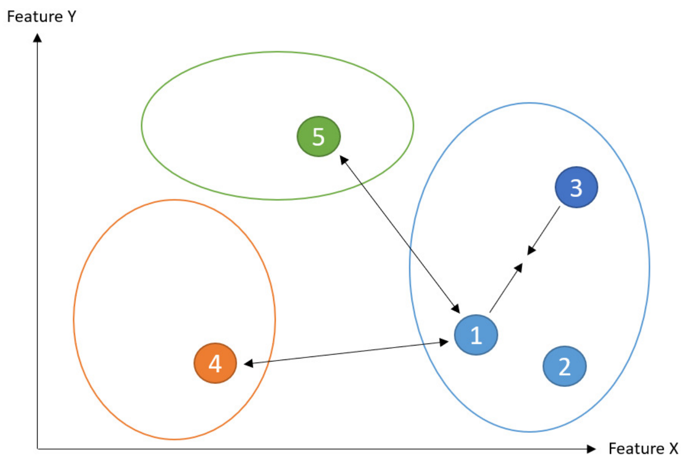


Figure 21: Dialect Separation Strategy

The idea was that samples 1 and 3 from the same dialect will be pulled together in the latent feature space, whereas samples 4 and 5 will be pushed away from sample 1. This pulling and pushing of samples is represented using arrows.

4.5.2 Experimental Setup

To create the sample pairs, the train dataset of the MIX-10000R corpus was used. The speaker separation strategy was evaluated 5 times per sample which resulted in 10 pairs per sample. For the dialect separation strategy, the evaluation was done 2 times which resulted in 14 pairs per sample. These newly created train datasets containing sample pairs were called MIX-10000RS5 and MIX-10000RD2, respectively. Both train datasets were quite large with 560,000 sample pairs in the MIX-10000RS5 train dataset and 784,000 sample pairs in the MIX-10000RD2 train dataset.

Four pre-training runs were performed in this experiment. The first run R35 was using the MIX-10000RS5 train dataset and the other two used the MIX-10000RD2 train dataset. These pre-train runs are listed in the Table 22 below.

Run	Model		Dataset
ID	**Pre-Trained**	**Notes**	**Train**
R35	whisper-base		MIX-10000RS5
R36	whisper-base		MIX-10000RD2
R37	whisper-base	Added Encoder Layer	MIX-10000RD2

Table 22: Contrastive Learning Pre-Train Runs

For all three models the whisper-base model was used. The SpecAugment augmentation method and the drop-out layers from the previous experiment were used. A single Adam optimiser with the same parameters as described in Chapter 3.8.1 above and a learning rate of 3×10^{-4} was used. The batch size was set to 256 which was achieved using an accumulation factor of 64.

To further increase the performance of the model, for run R37 an additional encoder head was added after the pooling. The used encoder head was a three-layered linear network with leaky rectified linear activation functions. The purpose of the encoder was to make the model more flexible after the pooling layer.

After the pre-training, three train runs were performed to train the classification head based on the learned representations. Those runs are listed in the Table 23 below.

Run	Model		Dataset	
ID	**Pre-Trained**	**Notes**	**Train**	**Valid**
R38	R36	Single Layer Classifier	MIX-10000R	MIX-20000
R39	R36	Extended Classifier	MIX-10000R	MIX-20000
R40	R37	Extended Classifier	MIX-10000R	MIX-20000
R41	R37	Not Frozen, Same as R25	MIX-10000R	MIX-20000

Table 23: Contrastive Learning Classification Runs

The first run R38 used a single layer classifier as used in the previous experiments and described as part of the model architecture in Chapter 3.4 above. The other experiments used an extended classifier with three additional linear layers with leaky rectified linear activation functions. The purpose of the extended classifier was to give the network more flexibility during training, since the representations learned during pre-training were frozen.

4.5.3 Results

Overall, the results from the representation learning were not as anticipated. And because the training took a significant amount of time, no further trainings were performed.

The Table 24 below summarises the results of the pre-training runs R35-37 including the achieved mean loss of the last training stage.

Run	Model		Dataset	Performance
ID	Pre-Trained	Notes	Train	Stage Loss
R35	whisper-base		MIX-10000RS5	0.50
R36	whisper-base		MIX-10000RD2	0.19
R37	whisper-base	Added Encoder Layer	MIX-10000RD2	0.18

Table 24: Contrastive Learning Pre-Train Results

The representation learning with the speaker separation strategy in run R35, did not seem to learn anything. The training loss did fluctuate a lot for run R35, but overall stayed right in the middle at around 0.5. This indicates that the cosine similarity of the representations was random. The second phase of training the classifier was therefore not ran for this model.

With the dialect separation strategy, the models were able to reduce the mean loss slightly below 0.2. This means that the mean cosine similarity of samples which are supposed to be similar approached 0.8 and 0.2 for dissimilar samples. With the addition of an encoder head in run R37, the loss was reduced even more.

During training, a high fluctuation of the loss was observed. However, the mean loss was slowly decreasing and did not appear to have reached convergence after 3 epochs. Due to long training times the training was aborted.

In the second phase, the pre-trained representations were used to train a classifier by adding a classification head. During training only, the classification head was trained, and the representations were frozen. The Table 25 on the next page summarises the results of those runs.

Run	Model		Performance	
ID	**Pre-Trained**	**Notes**	**Micro F1**	**Macro F1**
R38	R36	Single Layer Classifier	**18.5%**	**17.2%**
R39	R36	Extended Classifier	14.3%	10.0%
R40	R37	Extended Classifier	18.1%	14.7%

Table 25: Contrastive Learning Classification Results

Overall, the runs achieved very low F1 scores. With the best run R38 only reaching a micro F1 score of 18.5%. Using an extended classifier with more layers did not help to improve the performance of the model. The added encoder layer during representation learning in run R37 did not improve the performance.

In another run R41, the whole model was trained without frozen representations. With this setup the model was able to achieve a micro F1 score of 20.3%. This is only slightly better than with frozen representations. Interestingly, the run R41 without frozen representations uses the same layers, datasets and speech augmentation method as the run R25 from the previous experiment in Chapter 4.4.4 above. Despite their equal model architecture and dataset, the run R41 reached only a micro F1 score of 20.3%, whereas run R25 achieved a micro F1 score of 57.1%. The only difference between these two runs is that the run using representation learning started from the pre-trained checkpoint. This is an indication that during pre-training the model did not learn any useful representations. A possible explanation for this might be that the Siamese neural networks collapsed during representation learning despite the usage of contrastive learning.

4.6 Training the Best Model

In previous experiments various techniques to improve the performance for Swiss German dialect identification were explored. This experiment combined the learnings of the previous experiments with the goal to create a single model that achieves the highest performance for Swiss German dialect identification with the available resources.

The used model architecture proved to be capable of identifying Swiss German dialects, therefore the model architecture was kept. When it comes to wav2vec and Whisper there is not yet enough evidence to decide between one or another. Because of that, for this experiment both models were evaluated again.

The mixed corpora experiment showed that models trained on a mixed but reduced corpora achieved the best macro F1 scores and generalisation. For this experiment the model was therefore trained on mixed corpora containing samples from both the SDS-200 and STT4SG-350 corpus.

The speech augmentation experiment showed that using speech augmentation is a successful technique to improve performance on Swiss German dialect identification. From the evaluated augmentation methods, the SpecAugment augmentation method without extended batches as well as the combination of all augmentation methods performed best and were therefore evaluated in this experiment.

Since it was not possible to learn useful representations using contrastive learning in the previous experiment, no contrastive learning was used in this experiment.

4.6.1 Experimental Setup

A total of five runs were performed, in this experiment. The first three runs R42-44 used the SpecAugment augmentation method without extended batches. This means that the augmented version of the speech signal was used in place of the original speech signal in the batch. The last two runs R45 and R46 used all augmentation methods combined as explained in the Chapter 4.4.4 above. The Table 26 below lists all the runs from this experiment.

Run	**Model**		**Training**
ID	**Pre-Trained**	**Augmentation**	**Batch Size**
R42	wav2vec2-xls-r-300m	SpecAugment	32
R43	whisper-small	SpecAugment	32
R44	whisper-medium	SpecAugment	32
R45	wav2vec2-xls-r-300m	Combined Augmentations	64
R46	whisper-small	Combined Augmentations	64

Table 26: Best Model Runs

All the runs were trained for 5 epochs and used the same optimisers and learning rates as explained in Chapter 3.8.1 above. As in previous experiments, all the runs were trained on the train dataset of the MIX-10000R corpus and validated on the validation dataset of the MIX-20000 corpus.

To see which pre-trained model performed best, both the wav2vec2-xls-r-300m and whisper-small were trained and validated for each augmentation method. Additionally, the even larger whisper-medium with 769 million parameters was trained with the SpecAugment augmentation method.

After the above runs were completed, a hyperparameter tuning was performed for the best performing model. Both learning rates as well as the batch size were tuned over 5 epochs using Sweeps[38] from Weights & Biases. Bayesian hyperparameter optimisation was used to find the best hyperparameters [51]. No learning rate decay was used during hyperparameter tuning. The batch size was searched in the range from 2 to 64. This was done by setting the batch size to 2 and tuning the gradient accumulation factor in the range from 1 to 32. The learning rate of the pre-trained model was searched in the range from 3×10^{-5} to 3×10^{-8} and the learning rate of the rest of the model was searched in the range from 1×10^{-3} to 1×10^{-6}. Which corresponds to the initial and final learning rates used for the learning rate decay as described in Chapter 3.8.1 above. After the tuning of the hyperparameters, the

[38] https://wandb.ai/site/sweeps

model was then trained over 20 epochs using the tuned hyperparameters to achieve the final scores.

Contrary to the results of the speech augmentation experiment, all runs in this experiment used the added dropout layers. This was because, the effect of the dropout layers was uncovered after the above runs were already performed and no time was left to repeat the runs in this experiment. However, the hyperparameter tuning and the training of the final model over 20 epochs were performed without added dropout layers.

4.6.2 Results

From the results in the Table 27 below it can be seen that the wav2vec2-xls-r-300m model performed best for both augmentation methods. The run R25 and run R32 were added as a baseline to the Table 27 below for easier comparison.

Run	Model		Training	Performance	
ID	**Pre-Trained**	**Augmentation**	**Batch Size**	**Micro F1**	**Macro F1**
R25	whisper-base	SpecAugment	32	57.1%	54.6%
R42	wav2vec2-xls-r-300m	SpecAugment	32	**59.3%**	**57.2%**
R43	whisper-small	SpecAugment	32	58.2%	54.7%
R44	whisper-medium	SpecAugment	32	50.4%	46.8%
R32	whisper-base	Combined Augmentations	64	60.1%	57.5%
R45	wav2vec2-xls-r-300m	Combined Augmentations	64	**58.0%**	**54.0%**
R46	whisper-small	Combined Augmentations	64	56.4%	52.8%

Table 27: Best Model Results

Interestingly, using the SpecAugment augmentation method, the whisper-medium model performed significantly worse than the whisper-small model. This indicates that the whisper-medium model is most likely too big for the used training data, so that the model overfits to the trainings data or tends to recognise speakers.

Another interesting observation is that the runs R45 and R46 using larger models with the combined augmentation methods performed worse than the baseline run R32 using the whisper-base model.

Because the wav2vec2-xls-r-300m model with the SpecAugment augmentation method consistently performed best and the combined augmentation method required a high training duration, it was decided to tune the hyperparameters of the run R42. As already stated in the experimental setup above, no dropout layers were added to the model used in the hyperparameter optimisation.

During hyperparameter tuning a total of 29 runs were performed with different hyperparameters. The Figure 22 below visualises all the performed runs, their hyperparameters and their results. Every line represents a run and every bar a hyperparameter. The colour gradient on the right indicates the achieved macro F1 score of the runs.

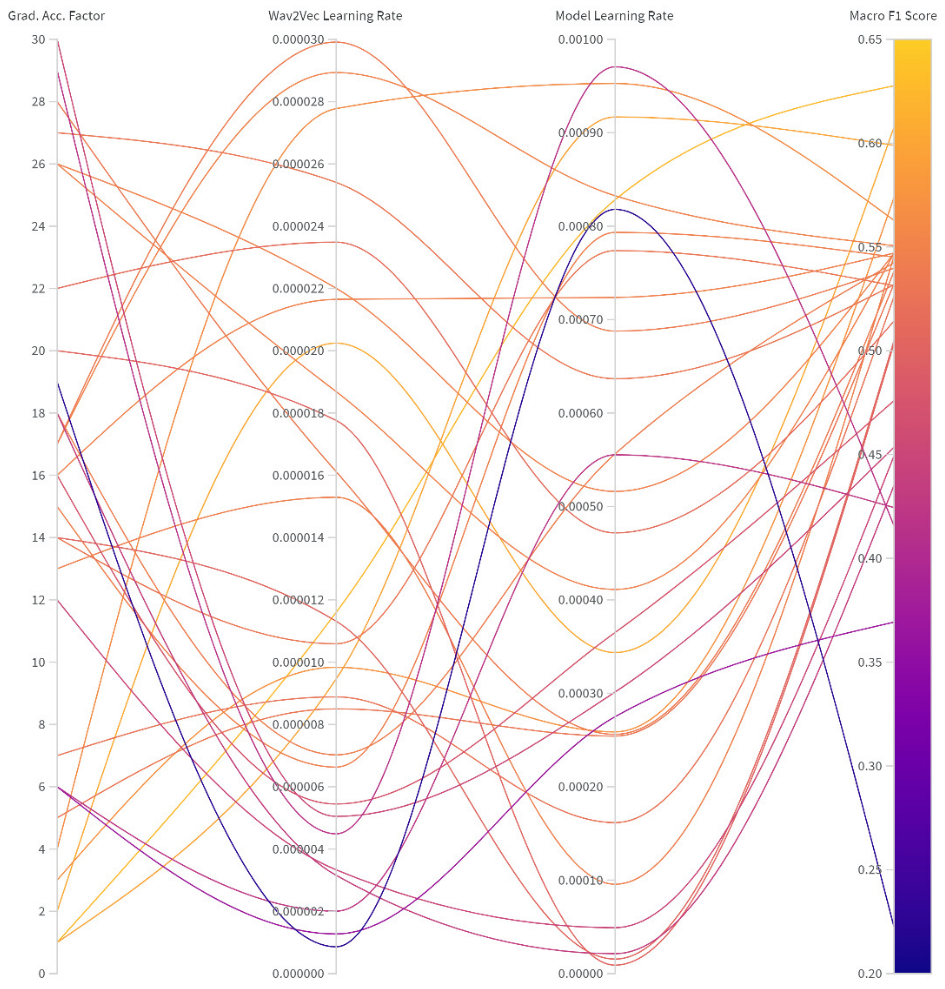


Figure 22: Hyperparameter Tuning Runs

The best performing hyperparameters were a gradient accumulation factor of 1, which corresponds to a batch size of 2, a wav2vec learning rate of 1.16×10^{-5} and a learning rate for the rest of the model of 8.28×10^{-4}. Those hyperparameters resulted in a micro F1 score of 64.72% and macro F1 score of 62.76%. Extending the training epochs to 20, did not further improve the F1 scores in run R47.

5 Results

The detailed results for each experiment were already presented in the dedicated results chapters. This chapter serves as a summary for the results from all the experiments.

Our experiments showed that models trained on the SDS-200 corpus reached higher F1 scores than models trained on the recently finalised STT4SG-350 corpus, even though the STT4SG-350 corpus contains more samples across all dialect regions. We conjecture that models trained on the STT4SG-350 corpus tend to recognise speakers instead of identifying dialects as there is not a big enough variety of speakers in the corpus.

Mixing the SDS-200 and STT4SG-350 corpora proved to be a promising strategy since a mixed corpus can take advantage of both the high variety of speakers in the SDS-200 corpus and the high number of samples in the STT4SG-350 corpus. While models trained on mixed corpora generally performed better than models trained only on the SDS-200 or STT4SG-350 corpus, smaller mixed corpora seemed to perform better than larger ones.

During further investigations of our speaker recognition theory, the experiments with mixed corpora of different sizes revealed that having too many samples of the same speaker and not enough variety can harm the generalisation of the model as it leads to speaker recognition. In our experiments a mixed corpora with 10,000 samples per subregion and a mean of around 50 samples per speaker seemed to perform best.

Our experiments with various augmentation methods showed that the generalisation of dialect identification systems can be improved by artificially increasing the number of samples and the variety of speakers. While some augmentation methods heavily benefited from using augmentation speech signals as additional samples, others were performing worse when using this approach.

While some experiments in the field of representation learning were conducted, it was not possible to learn useful representations for dialect identification with the employed setup.

When comparing wav2vec and Whisper models both have similar performance and can be used interchangeably. One benefit of the Whisper models is that smaller sizes are available which can be useful for prototyping because of the lower training durations but still reasonable performance.

In our final experiment, the wav2vec-large-xlsr-53 model in combination with a mixed corpora of 10,000 samples per subregion and speech augmentation was used for the best performing model which achieved a micro F1 score of 64.72%.

6 Discussion

During our pre-processing, some samples and speakers were removed from the SDS-200 corpus because the mapping to subregions was not straightforward. For future work, it is recommended to look for other ways to include those samples for experiments related to subregions. The paper “A quantitative approach to Swiss German - Dialectometric analyses and comparisons of linguistic levels” [52] describes a clustering method, which was utilised during the creation of the STT4SG-350 corpus and can assist in mapping the samples from the SDS-200 corpus to the subregions.

While the results showed that models trained on the MIX-10000 performed better than models trained on the MIX-20000 corpus, a balanced mixed corpora up to the size of 29,717 samples per dialect region would have been possible. However, this possibility was not further explored in this thesis, because the MIX-10000 corpus seemed to perform better than the MIX-20000 corpus, nonetheless.

The evaluated reduced corpora were restricted to the initially created MIX-20000 corpora and due to the chosen train-test-split strategy for the mixed corpora, no common validation dataset was available to compare the models trained on the MIX-ALL between ones trained on the MIX-20000 or MIX-10000. A better strategy would have been to do a train-test-split on the MIX-ALL corpus and then reducing the train dataset to different sizes.

Speech augmentation proved to be a successful tool for improving the performance in dialect identification. However, there are still a lot of augmentation methods which have not been explored yet. While combining augmentation methods achieved good results, more tests would need to be carried out to validate the effect of combining augmentations.

The effect of the added dropout layers was not evaluated in the beginning. Because of that dropout layers were present in later models without having a positive effect. Unfortunately, not enough time was left to repeat all the experiments without added dropout layers.

With the employed contrastive learning setup, it was unfortunately not possible to learn useful speech representations. However, contrastive learning might still be a promising strategy. There are a lot of possibilities in this area which have not been explored yet. For example, a promising strategy would be the use of contrastive learning on the same sentence spoken in different dialects. However, only 8,988 of 304,141 samples from both SDS-200 and STT4SG-350 corpora have the same sentence which corresponds to only 2.95%.

For most experiments, the tuneable hyperparameters such as the used optimiser, the learning rate and the batch size were the same. This was intentional to allow for easier comparison between models. However, this may have given an unfair advantage to some of the models as they could have benefited from the selected hyperparameters while other models would have performed better with other hyperparameters. A better option would have been to optimise the hyperparameters for every model individually, but doing a hyperparameter optimisation on every model would have been too resource intensive for this thesis.

By mixing the available corpora and applying speech augmentation, the best model from this thesis performed significantly better than models from previous work on Swiss German dialect identification, especially when it comes to macro F1 scores, thereby setting a new baseline for future systems.

7 Index

7.1 Bibliography

[1] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, ‘wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations’, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.11477

[2] S. Schneider, A. Baevski, R. Collobert, and M. Auli, ‘wav2vec: Unsupervised Pre-training for Speech Recognition’, Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.05862

[3] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, ‘Unsupervised Cross-lingual Representation Learning for Speech Recognition’, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.13979

[4] A. Babu *et al.*, ‘XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale’, Nov. 2021, [Online]. Available: http://arxiv.org/abs/2111.09296

[5] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, ‘Robust Speech Recognition via Large-Scale Weak Supervision’, 2022. Accessed: May 29, 2023. [Online]. Available: https://arxiv.org/abs/2212.04356

[6] M. Plüss *et al.*, ‘SDS-200: A Swiss German Speech to Standard German Text Corpus’, 2022, Accessed: Jun. 07, 2023. [Online]. Available: http://arxiv.org/abs/2205.09501

[7] M. Plüss *et al.*, ‘STT4SG-350: A Speech Corpus for All Swiss German Dialect Regions’, May 2023, [Online]. Available: http://arxiv.org/abs/2305.18855

[8] S. Stucki and P. Randjelovic, ‘Automatic Detection of Swiss German Dialects using Wav2Vec’, 2021. Accessed: May 29, 2023. [Online]. Available: https://www.zhaw.ch/storage/engineering/institute-zentren/cai/Project_Thesis_Automatic_Dialect_Detection.pdf

[9] S. Stucki and P. Randjelovic, ‘Exploring Wav2Vec2 Pre-Training on Swiss German Dialects using Speech Translation and Classification’, 2022. Accessed: May 29, 2023. [Online]. Available: https://www.zhaw.ch/storage/engineering/institute-zentren/cai/studentische_arbeiten/BA22_ciel_Stucki_Ranjelovic_Wave2Vec_for_Swiss_German.pdf

[10] ‘Cambridge Dictionary’. https://dictionary.cambridge.org/dictionary/english (accessed May 29, 2023).

[11] G. Mingliang and X. Yuguo, ‘Chinese dialect identification using clustered support vector machine’, in *2008 International Conference on Neural Networks and Signal Processing*, 2008, pp. 396–396. doi: 10.1109/ICNNSP.2008.4590380.

[12] L. Nour-Eddine and A. Abdelkader, ‘GMM-based maghreb dialect identification system’, *Journal of Information Processing Systems*, vol. 11, no. 1, pp. 22–38, 2015, doi: 10.3745/JIPS.02.0015.

[13] S. Shon, A. Ali, and J. Glass, ‘Convolutional Neural Networks and Language Embeddings for End-to-End Dialect Recognition’, Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.04567

[14] A. Ahmed, P. Tangri, A. Panda, D. Ramani, and S. Karmakar, ‘VFNet: A convolutional architecture for accent classification’, in *2019 IEEE 16th India Council International Conference (INDICON)*, Institute of Electrical and Electronics Engineers Inc., Dec. 2019, pp. 1–4. doi: 10.1109/INDICON47234.2019.9030363.

[15] M. Alrehaili, T. Alasmari, and A. Aoalshutayri, ‘Arabic Speech Dialect Classification using Deep Learning’, in *2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC)*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 1–5. doi: 10.1109/ICAISC56366.2023.10085647.

[16] C. Korkut, A. Haznedaroglu, and L. Arslan, ‘Comparison of Deep Learning Methods for Spoken Language Identification’, in *Speech and Computer*, Springer Science and Business Media Deutschland GmbH, 2020, pp. 223–231. doi: 10.1007/978-3-030-60276-5_23.

[17] D. S. Park *et al.*, ‘SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition’, in *Proc. Interspeech 2019*, Apr. 2019, pp. 2613–2617. doi: 10.21437/Interspeech.2019-2680.

[18] P. Fivian and D. Reiser, ‘Speech Classification using wav2vec 2.0’, 2021. Accessed: Jun. 07, 2023. [Online]. Available: https://www.zhaw.ch/storage/engineering/institute-zentren/cai/BA21_Speech_Classification_Reiser_Fivian.pdf

[19] T. Samardžić, Y. Scherrer, and E. Glaser, ‘ArchiMob - A Corpus of Spoken Swiss German’, in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, 2016, pp. 4061–4066. Accessed: Jun. 03, 2023. [Online]. Available: https://aclanthology.org/L16-1641

[20] M. Plüss, L. Neukom, C. Scheller, and M. Vogel, ‘Swiss Parliaments Corpus, an Automatically Aligned Swiss German Speech to Standard German Text Corpus’, Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.02810

[21] P. N. Garner, D. Imseng, and T. Meyer, ‘Automatic Speech Recognition and Translation of a Swiss German Dialect: Walliserdeutsch’, in *Proceedings of Interspeech*, 2014, pp. 2118–2122. doi: https://doi.org/10.21437/Interspeech.2014-480.

[22] P. Dogan-Schönberger, J. Mäder, and T. Hofmann, ‘SwissDial: Parallel Multidialectal Corpus of Spoken Swiss German’, Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.11401

[23] M. Zampieri *et al.*, ‘Findings of the VarDial Evaluation Campaign 2017’, in *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Association for Computational Linguistics, 2017, pp. 1–15. doi: 10.18653/v1/W17-1201.

[24] M. Zampieri *et al.*, ‘Language Identification and Morphosyntactic Tagging: The Second VarDial Evaluation Campaign’, in *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, 2018, pp. 1–17. Accessed: Jun. 07, 2023. [Online]. Available: https://aclanthology.org/W18-3901

[25] M. Zampieri *et al.*, ‘A Report on the Third VarDial Evaluation Campaign’, in *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, 2019, pp. 1–16. doi: 10.18653/v1/W19-1401.

[26] E. Dieth, ‘Schwyzertütschi Dialäktschrift’, *Sauerländer, Aarau*, 1986.

[27] S. Malmasi and M. Zampieri, ‘MAZA Submissions to VarDial 2017’, 2017. [Online]. Available: https://www.researchgate.net/publication/316602155

[28] T. Jauhiainen, H. Jauhiainen, and K. Lindén, ‘HeLI-based Experiments in Swiss German Dialect Identification’, in *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, 2018, pp. 254–262. Accessed: Jun. 07, 2023. [Online]. Available: http://hdl.handle.net/10138/308720

[29] N. Wu, E. DeMattos, K. H. So, P. Chen, and Ç. Çöltekin, ‘Language Discrimination and Transfer Learning for Similar Languages: Experiments with Feature Combinations and Adaptation’, in *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects*, 2019, pp. 54–63. doi: 10.18653/v1/W19-1406.

[30] P. Von Däniken, M. Hürlimann, and M. Cieliebak, ‘Overview of the GermEval 2020 Shared Task on Swiss German Language Identification’, in *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS)*, 2020. doi: 10.21256/zhaw-21549.

[31] M. Banaei, R. Lebret, and K. Aberer, ‘Spoken dialect identification in Twitter using a multi-filter architecture’, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.03564

[32] R. Sathya and A. Abraham, ‘Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification’, *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, pp. 34–38, 2013, doi: 10.14569/IJARAI.2013.020206.

[33] Y. Bengio, A. Courville, and P. Vincent, ‘Representation Learning: A Review and New Perspectives’, *IEEE Trans Pattern Anal Mach Intell*, vol. 35, no. 8, pp. 1798–1828, 2013, doi: 10.1109/TPAMI.2013.50.

[34] P. H. Le-Khac, G. Healy, and A. F. Smeaton, ‘Contrastive Representation Learning: A Framework and Review’, *IEEE Access*, vol. 8, pp. 193907–193934, 2020, doi: 10.1109/ACCESS.2020.3031549.

[35] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, ‘A Simple Framework for Contrastive Learning of Visual Representations’, Feb. 2020, [Online]. Available: http://arxiv.org/abs/2002.05709

[36] Y. LeCun, Y. Bengio, and G. Hinton, ‘Deep learning’, *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[37] D. Bahdanau, K. Cho, and Y. Bengio, ‘Neural Machine Translation by Jointly Learning to Align and Translate’, Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.0473

[38] A. Vaswani *et al.*, ‘Attention Is All You Need’, Jun. 2017, [Online]. Available: http://arxiv.org/abs/1706.03762

[39] M. Grandini, E. Bagli, and G. Visani, ‘Metrics for Multi-Class Classification: an Overview’, Aug. 2020, [Online]. Available: http://arxiv.org/abs/2008.05756

[40] ‘Swiss NLP’, *https://swissnlp.org/datasets/*, Apr. 2023.

[41] Federal Office of Topography (swisstopo), ‘Official directory of towns and cities’, Apr. 2023. https://www.swisstopo.admin.ch/en/geodata/official-geographic-directories/directory-towns-cities.html (accessed Mar. 03, 2023).

[42] M. Ravanelli *et al.*, ‘SpeechBrain: A General-Purpose Speech Toolkit’, Jun. 2021, [Online]. Available: http://arxiv.org/abs/2106.04624

[43] D. P. Kingma and J. Ba, ‘Adam: A Method for Stochastic Optimization’, Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980

[44] Y. Bengio, ‘Practical recommendations for gradient-based training of deep architectures’, Jun. 2012, [Online]. Available: http://arxiv.org/abs/1206.5533

[45] K. You, M. Long, J. Wang, and M. I. Jordan, ‘How Does Learning Rate Decay Help Modern Neural Networks?’, Aug. 2019, [Online]. Available: http://arxiv.org/abs/1908.01878

[46] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, ‘On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima’, Sep. 2016, [Online]. Available: http://arxiv.org/abs/1609.04836

[47] M. Buda, A. Maki, and M. A. Mazurowski, ‘A systematic study of the class imbalance problem in convolutional neural networks’, *Neural Networks*, vol. 106, pp. 249–259, 2018, doi: https://doi.org/10.1016/j.neunet.2018.07.011.

[48] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, ‘Audio augmentation for speech recognition’, in *Proc. Interspeech 2015*, 2015, pp. 3586–3589. doi: 10.21437/Interspeech.2015-711.

[49] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, ‘Improving neural networks by preventing co-adaptation of feature detectors’, Jul. 2012, [Online]. Available: http://arxiv.org/abs/1207.0580

[50] A. Saeed, D. Grangier, and N. Zeghidour, ‘Contrastive Learning of General-Purpose Audio Representations’, in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3875–3879. doi: 10.1109/ICASSP39728.2021.9413528.

[51] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, ‘Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb’, *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019, doi: https://doi.org/10.11989/JEST.1674-862X.80904120.

[52] Y. Scherrer and P. Stoeckle, ‘A quantitative approach to Swiss German - Dialectometric analyses and comparisons of linguistic levels’, *Dialectologia et Geolinguistica*, vol. 24, no. 1, pp. 92–125, 2016, doi: 10.1515/dialect-2016-0006.

7.2 Figures

7.3 Tables

7.4 Equations

7.5 Acronyms

APU	Accelerated Processing Unit
AWS	Amazon Web Services
CAI	Centre for Artificial Intelligence
ETH	Swiss Federal Institute of Technology
EULA	End User License Agreement
FFMPEG	Fast Forward Moving Picture Experts Group
FHNW	University of Applied Sciences and Arts Northwestern Switzerland
FLAC	Free Lossless Audio Code
GPU	Graphics Processing Unit
HDF5	Hierarchical Data Format 5
MD5	Message-Digest Algorithm 5
MGB-3	Multi-Genre Broadcast (third edition)
MP3	MPEG-2 Audio Layer III
NLP	Natural Language Processing
OSMnx	OpenStreetMap Extended
SDS-200	Schweizer Dialektsammlung (200 hours of speech)
SSH	Secure Shell
STT4SG-350	Speech-to-Text for Swiss German (343 hours of speech)
YAML	YAML Ain’t Markup Language, Yet Another Markup Language
ZHAW	Zurich University of Applied Sciences

8 Appendix

8.1 Code and Corpora

The code for the experiments is available on:
https://github.zhaw.ch/FS23-BA-freicla4-schneph5/swiss-german-dialect-identi-
fication

The code including all used data and experiment results is available on:
https://zhaw-my.sharepoint.com/:f:/g/personal/schneph5_stu-
dents_zhaw_ch/EpQPXXl8cRZNp8BUzi4OsKMBp0yLfSkPQrpVBVN7J4DJHQ

8.2 Experiment Details

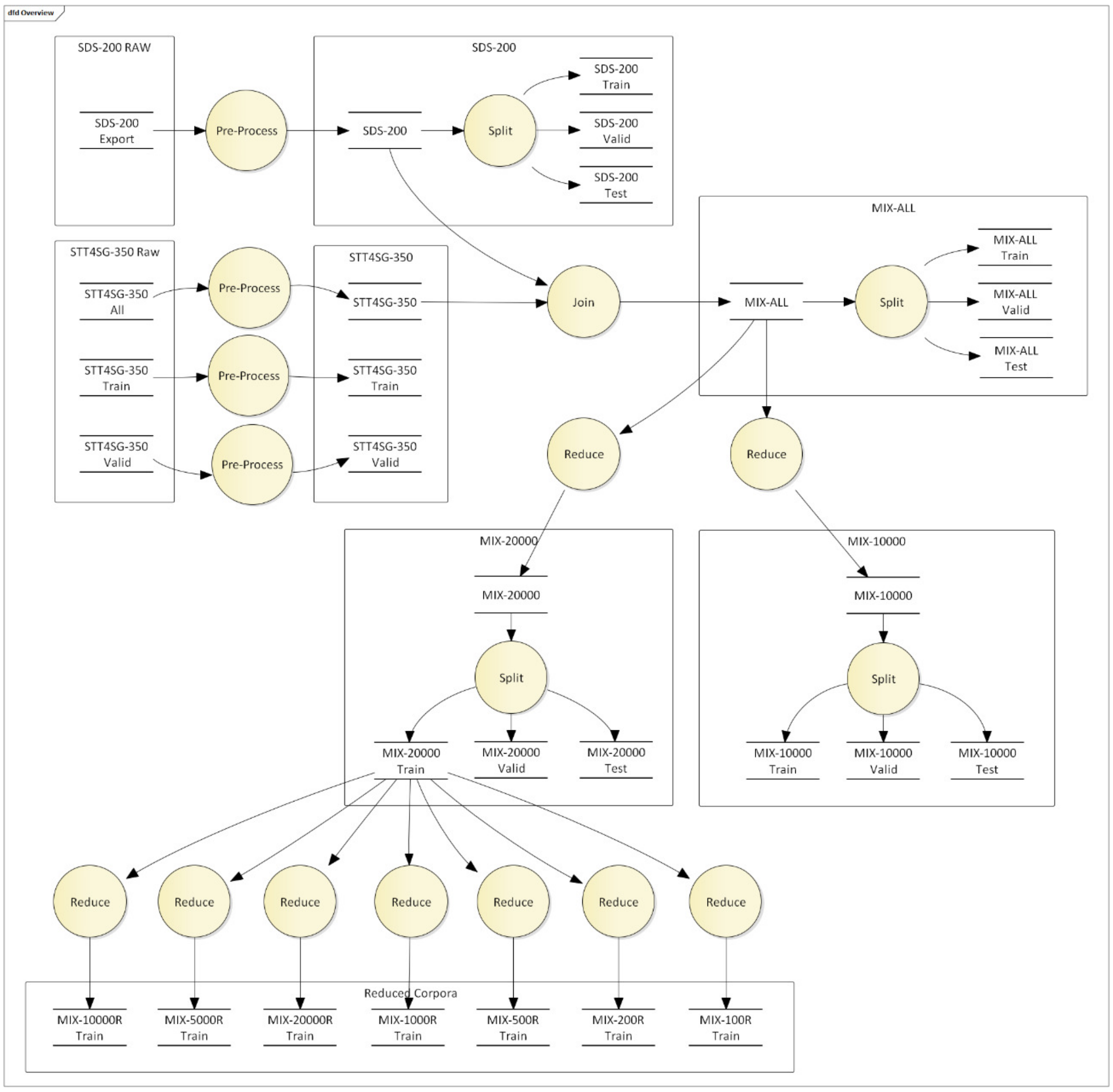
The table on the next page contains the details of all the performed experiments during this bachelor's thesis.

Run				Model			Dataset		Training Parameters					Performance		Training
ID	Name	Started at	Machine	Pre-Trained	Label	Notes	Train	Valid	Epochs	Batch Size		LR Pre-Trained	LR Model	Micro F1	Macro F1	Duration
Exploring Models and Corpora																
R01	wav2vec-run-0025	28.03.23 10:30	VAST (RTX 3090 24GB)	wav2vec2-xls-r-300m	canton group		SDS-200	SDS-200	3	48	4 x 12	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	66.6% - 70.4%	39.6% - 49.8%	0 days 04:16:26
R02	wav2vec-run-0029	04.04.23 07:44	APU (Tesla T4 16GB)	wav2vec2-xls-r-300m	canton group		STT4SG-350	STT4SG-350	3	48	2 x 24	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	34.3% - 39.6%	33.6% - 37.7%	1 days 08:47:17
R03	wav2vec-run-0024	28.03.23 06:06	VAST (RTX 3090 24GB)	wav2vec2-xls-r-300m	subregion		SDS-200	SDS-200	3	48	4 x 12	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	53.1% - 63.1%	32.4% - 35.8%	0 days 03:39:19
R04	wav2vec-run-0028	02.04.23 18:34	APU (Tesla T4 16GB)	wav2vec2-xls-r-300m	subregion		STT4SG-350	STT4SG-350	3	48	2 x 24	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	20.9% - 23.9%	19.1% - 20.9%	1 days 08:38:35
R05	whisper-run-0004	28.03.23 15:18	VAST (RTX 3090 24GB)	whisper-small	canton group		SDS-200	SDS-200	3	48	8 x 6	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	57.9% - 67.5%	48.5% - 49.7%	0 days 06:59:17
R06	whisper-run-0002	27.03.23 17:37	APU (Tesla T4 16GB)	whisper-small	canton group		STT4SG-350	STT4SG-350	3	48	4 x 12	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	38.1% - 38.6%	37.2% - 37.9%	2 days 12:32:32
R07	whisper-run-0007	03.04.23 21:56	VAST (RTX 3090 24GB)	whisper-small	subregion		SDS-200	SDS-200	3	48	8 x 6	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	59.2% - 69.3%	37.4% - 43.1%	0 days 05:58:15
R08	whisper-run-0005	30.03.23 12:42	APU (Tesla T4 16GB)	whisper-small	subregion		STT4SG-350	STT4SG-350	3	48	4 x 12	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	25.5% - 25.9%	24.6% - 26.2%	2 days 13:55:20
R09	whisper-run-0006	03.04.23 20:07	VAST (RTX 3090 24GB)	whisper-base	subregion		SDS-200	SDS-200	3	48	16 x 3	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	48.0% - 55.0%	33.6% - 37.9%	0 days 01:42:54
R10	wav2vec-run-0031	13.04.23 17:50	VAST (RTX 3090 24GB)	wav2vec2-xls-r-300m	subregion		SDS-200	SDS-200	20	48	4 x 12	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	48.2% - 66.6%	29.5% - 42.0%	1 days 00:11:25
R11	wav2vec-run-0032	13.04.23 17:52	VAST (RTX 3090 24GB)	wav2vec2-xls-r-300m	subregion		STT4SG-350	STT4SG-350	20	48	4 x 12	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	15.9% - 25.3%	14.9% - 23.2%	2 days 01:17:29
Exploring Mixed Corpora																
R12	whisper-run-0053	25.05.23 18:53	VAST (RTX 3090 24GB)	whisper-base	subregion		MIX-ALL	MIX-ALL	20	48	8 x 6	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	50.7% - 61.7%	47.4% - 59.2%	1 days 10:58:20
R13	whisper-run-0008	09.04.23 20:36	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-20000	MIX-20000	20	48	8 x 6	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	45.0% - 53.0%	41.8% - 51.1%	3 days 00:04:09
R14	whisper-run-0009	13.04.23 11:59	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-10000	MIX-10000	20	48	8 x 6	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	54.5% - 61.4%	52.3% - 60.8%	1 days 12:20:12
R15	wav2vec-run-0033	15.04.23 20:14	VAST (RTX 3090 24GB)	wav2vec2-xls-r-300m	subregion		MIX-20000	MIX-20000	20	48	4 x 12	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	44.2% - 54.6%	41.8% - 51.7%	1 days 10:59:40
R16	whisper-run-0010	15.04.23 08:45	VAST (RTX 3090 24GB)	whisper-small	subregion		MIX-20000	MIX-20000	20	48	4 x 12	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	39.3% - 49.1%	33.9% - 47.6%	2 days 02:39:47
Exploring Speaker Recognition																
R17	whisper-run-0013	17.04.23 13:31	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-100R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	14.4% - 29.3%	7.5% - 26.8%	0 days 00:49:29
R18	whisper-run-0014	17.04.23 14:25	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-200R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	11.8% - 31.3%	3.3% - 27.4%	0 days 00:52:51
R19	whisper-run-0015	17.04.23 15:19	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-500R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	19.9% - 37.9%	14.9% - 32.9%	0 days 01:13:00
R20	whisper-run-0016	17.04.23 16:33	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-1000R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	34.1% - 45.0%	30.6% - 42.8%	0 days 01:32:10
R21	whisper-run-0017	17.04.23 18:06	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-2000R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	34.0% - 48.2%	32.9% - 47.4%	0 days 02:22:31
R22	whisper-run-0018	17.04.23 20:30	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-5000R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	40.2% - 51.3%	40.8% - 48.5%	0 days 05:05:33
R23	whisper-run-0019	18.04.23 01:39	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-10000R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	40.4% - 52.1%	39.4% - 50.5%	0 days 09:00:03
R24	whisper-run-0020	18.04.23 10:42	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-20000	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	41.7% - 51.7%	40.8% - 47.6%	0 days 17:42:13
	whisper-run-0021	19.04.23 20:20	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-5000R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-04 - 1.00E-06	43.7% - 50.5%	42.8% - 48.7%	0 days 04:57:30
	whisper-run-0022	20.04.23 04:52	APU (Tesla T4 16GB)	whisper-base	subregion		MIX-5000R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	3.00E-05 - 2.60E-05	42.7% - 47.7%	41.9% - 46.0%	0 days 04:46:49
Exploring Speech Augmentation																
R25	whisper-run-0024	20.04.23 09:43	APU (Tesla T4 16GB)	whisper-base	subregion	SpecAugment	MIX-10000R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	49.5% - 57.1%	49.8% - 54.6%	0 days 09:34:14
R26	whisper-run-0049	23.05.23 15:25	VAST (RTX 3090 24GB)	whisper-base	subregion	Speech Augment	MIX-10000R	MIX-20000	5	32	16 x 2	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	47.7% - 52.4%	45.2% - 48.9%	0 days 10:28:59
R27	whisper-run-0042	22.05.23 01:47	VAST (RTX 3090 24GB)	whisper-base	subregion	Speaker Mix (1s Chunks)	MIX-10000R	MIX-20000	5	32	16 x 2	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	21.0% - 35.9%	14.2% - 30.6%	2 days 05:02:58
R28	whisper-run-0052	25.05.23 00:15	VAST (RTX 3090 24GB)	whisper-base	subregion	SpecAugment	MIX-10000R	MIX-20000	5	64	16 x 4	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	49.7% - 55.4%	47.7% - 52.6%	0 days 03:53:03
R29	whisper-run-0050	24.05.23 10:22	VAST (RTX 3090 24GB)	whisper-base	subregion	Speech Augment	MIX-10000R	MIX-20000	5	64	16 x 4	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	51.4% - 55.4%	48.8% - 53.4%	0 days 12:27:11
R30	whisper-run-0051	24.05.23 10:55	VAST (RTX 3090 24GB)	whisper-base	subregion	Speaker Mix (1s Chunks)	MIX-10000R	MIX-20000	5	64	16 x 4	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	45.8% - 52.7%	44.7% - 50.9%	2 days 07:18:10
R31	whisper-run-0054	27.05.23 11:52	VAST (RTX 3090 24GB)	whisper-base	subregion	Speaker Mix (Full Length)	MIX-10000R	MIX-20000	5	64	16 x 4	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	43.6% - 53.7%	43.9% - 50.5%	0 days 14:50:43
R32	whisper-run-0057	28.05.23 14:06	VAST (RTX 3090 24GB)	whisper-base	subregion	Combined Augmentations	MIX-10000R	MIX-20000	5	64	16 x 4	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	53.3% - 60.1%	52.9% - 57.5%	2 days 00:34:34
R33	whisper-run-0058	30.05.23 19:39	VAST (RTX 3090 24GB)	whisper-base	subregion	None Augment, Dropout	MIX-10000R	MIX-20000	5	32	16 x 2	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	40.2% - 50.5%	39.7% - 47.3%	0 days 01:51:33
R34	whisper-run-0061	01.06.23 20:43	VAST (RTX 3090 24GB)	whisper-base	subregion	SpecAugment, No Dropout	MIX-10000R	MIX-20000	5	32	16 x 2	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	49.6% - 61.6%	49.7% - 59.5%	0 days 01:58:16
	whisper-run-0025	20.04.23 19:58	APU (Tesla T4 16GB)	whisper-base	subregion	SpecAugment	MIX-10000R	MIX-20000	20	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	50.8% - 59.1%	48.2% - 55.6%	1 days 14:37:46
	whisper-run-0026	22.04.23 23:23	APU (Tesla T4 16GB)	whisper-base	subregion	SpecAugment	MIX-10000R	MIX-20000	20	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-04 - 1.00E-06	44.6% - 55.0%	42.1% - 51.0%	1 days 23:10:41

Run				Model			Dataset		Training Parameters					Performance		Training
ID	Name	Started at	Machine	Pre-Trained	Label	Notes	Train	Valid	Epochs	Batch size		LR Pre-Trained	LR Model	Micro F1	Macro F1	Duration
Exploring Contrastive Learning																
R35	whisper-run-0029	25.04.23 15:53	APU (Tesla T4 16GB)	whisper-base			MIX-10000RS5		3	256	4 x 64	3.00E-04 - 2.69E-04	3.00E-04 - 2.69E-04			4 days 04:44:25
R36	whisper-run-0030	05.05.23 22:18	APU (Tesla T4 16GB)	whisper-base			MIX-10000RD2		3	256	4 x 64	3.00E-04 - 2.69E-04	3.00E-04 - 2.69E-04			5 days 12:37:43
R37	whisper-run-0035	14.05.23 22:03	APU (Tesla T4 16GB)	whisper-base		Added Encoder Layer	MIX-10000RD2		3	256	4 x 64	3.00E-04 - 2.69E-04	3.00E-04 - 2.69E-04			5 days 11:55:40
R38	whisper-run-0031	11.05.23 11:10	APU (Tesla T4 16GB)	whisper-run-0030	subregion	Single Layer Classifier	MIX-10000R	MIX-20000	13	32	4 x 8	3.00E-05 - 1.11E-05	1.00E-03 - 3.69E-04	13.1% - 18.5%	12.4% - 17.2%	0 days 09:47:25
R39	whisper-run-0034	14.05.23 20:33	APU (Tesla T4 16GB)	whisper-run-0030	subregion	Extended Classifier	MIX-10000R	MIX-20000	1	32	4 x 8	3.00E-05 - 3.00E-05	1.00E-03 - 1.00E-03	14.3% - 14.3%	10.0% - 10.0%	0 days 00:46:58
R40	whisper-run-0036	20.05.23 11:14	APU (Tesla T4 16GB)	whisper-run-0035	subregion	Extended Classifier	MIX-10000R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 1.00E-06	16.6% - 18.1%	12.2% - 14.7%	0 days 03:41:46
R41	whisper-run-0032	11.05.23 21:26	APU (Tesla T4 16GB)	whisper-run-0035	subregion	Not Frozen, Same as R25	MIX-10000R	MIX-20000	5	32	4 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	11.2% - 20.3%	8.7% - 16.7%	0 days 09:15:06
Training the Best Model																
R42	wav2vec-run-0045	22.05.23 07:53	VAST (RTX 3090 24GB)	wav2vec2-xls-r-300m	subregion	SpecAugment	MIX-10000R	MIX-20000	5	32	2 x 16	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	44.3% - 59.3%	44.0% - 57.2%	0 days 05:20:42
R43	whisper-run-0046	22.05.23 14:42	VAST (RTX 3090 24GB)	whisper-small	subregion	SpecAugment	MIX-10000R	MIX-20000	5	32	8 x 4	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	53.8% - 58.2%	51.2% - 54.7%	0 days 06:36:39
R44	whisper-run-0056	28.05.23 10:19	VAST (RTX 3090 24GB)	whisper-medium	subregion	SpecAugment	MIX-10000R	MIX-20000	5	32	1 x 32	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	47.4% - 50.4%	44.1% - 46.8%	0 days 21:17:15
R45	wav2vec-run-0048	29.05.23 22:58	VAST (RTX 3090 24GB)	wav2vec2-xls-r-300m	subregion	Combined Augmentations	MIX-10000R	MIX-20000	5	64	2 x 32	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	49.4% - 58.0%	46.9% - 54.0%	2 days 14:28:23
R46	whisper-run-0060	30.05.23 22:05	VAST (RTX 3090 24GB)	whisper-small	subregion	Combined Augmentations	MIX-10000R	MIX-20000	5	64	8 x 8	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	52.1% - 56.4%	47.9% - 52.8%	2 days 09:13:28
R47	wav2vec-run-0049	06.06.23 02:43	VAST (RTX 3090 24GB)	wav2vec2-xls-r-300m	subregion	SpecAugment	MIX-10000R	MIX-20000	20	2	2 x 1	1.16E-05	8.28E-04	49.5% - 64.6%	44.5% - 62.1%	1 days 01:27:35
	wav2vec-run-0046	26.05.23 22:10	VAST (RTX 3090 24GB)	wav2vec2-xls-r-300m	subregion	SpecAugment	MIX-10000R	MIX-20000	20	32	2 x 16	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	45.7% - 60.6%	45.8% - 57.5%	0 days 21:43:54
	whisper-run-0048	22.05.23 22:19	VAST (RTX 3090 24GB)	whisper-base	subregion	SpecAugment	MIX-20000	MIX-20000	20	32	16 x 2	3.00E-05 - 3.00E-08	1.00E-03 - 1.00E-06	45.5% - 55.5%	43.5% - 54.0%	0 days 15:01:17

8.3 Corpora Overview

The figure below visualises the data flow between all the used corpora. The data initially comes from the SDS-200 and STT4SG-350 corpora and then multiple transformations such as pre-processing, splits, joins and reductions were performed to obtain each corpus and the associated train, valid and test datasets. In the figure below, "Pre-Process" refers to the pre-processing steps described in Chapter 3.2 above, "Split" refers to the train-test-split method described in Chapter 3.3 above and "Reduce" refers to the corpora reduction method described Chapter 4.2.2.



8.4 Corpora Details

The table on the next page contains the details of all the corpora and their train-test splits used during this bachelor's thesis.

Corpus Details					Total		Canton Group Label								Subregion Label													
Name	Type	Corpus create from	Split	Notes	Samples	Speakers	Speakers				Samples				Speakers							Samples						
							CA	EA	HA	WA	CA	EA	HA	WA	Basel	Bern	Grison	Central Switzerland	Eastern Switzerland	Valais	Zurich	Basel	Bern	Grison	Central Switzerland	Eastern Switzerland	Valais	Zurich
SDS-200																												
SDS-200	RAW			Original corpus	155,271	**4,207**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SDS-200	**Corpus**	**SDS-200 RAW**		**Pre-processed as described in Chapter 3.2**	**122,599**	**1,397**	**592**	**316**	**95**	**394**	**65,722**	**16,572**	**16,021**	**24,284**	**61**	**333**	**56**	**136**	**159**	**30**	**366**	**3,621**	**20,664**	**3,565**	**7,055**	**6,041**	**11,760**	**55,121**
	Train	SDS-200 Corpus	subregion	Splitted as described in Chapter 3.3 (80%)	89,409	912	369	172	63	308	47,223	7,316	14,528	20,336	55	255	43	107	129	26	297	3,213	17,143	2,923	5,509	4,393	11,596	44,626
	Valid	SDS-200 Corpus	subregion	Splitted as described in Chapter 3.3 (10%)	7,740	114	42	23	7	42	4,564	1,457	164	1,555	1	41	8	13	15	2	34	36	1,519	501	389	956	69	4,270
	Test	SDS-200 Corpus	subregion	Splitted as described in Chapter 3.3 (10%)	10,684	115	40	20	13	42	6,307	833	1,170	2,374	5	37	5	16	15	2	35	372	2,002	141	1,157	692	95	6,225
	Train	SDS-200 Corpus	canton group	Splitted as described in Chapter 3.3 (80%)	101,434	1,117	470	252	72	323	53,677	12,931	14,401	20,425	49	274	45	107	130	24	293	1,476	18,950	2,632	5,566	4,711	11,053	47,576
	Valid	SDS-200 Corpus	canton group	Splitted as described in Chapter 3.3 (10%)	10,889	140	53	33	11	43	6,752	1,956	628	1,553	8	35	3	11	14	3	30	488	1,065	721	324	344	510	3,278
	Test	SDS-200 Corpus	canton group	Splitted as described in Chapter 3.3 (10%)	10,276	140	69	31	12	28	5,293	1,685	992	2,306	4	24	8	18	15	3	43	1,657	649	212	1,165	986	197	4,267
STT4SG-350																												
STT4SG-350	RAW			Original balanced corpus	217,030	240	-	-	-	-	-	-	-	-	32	34	35	32	37	36	34	30,654	32,168	27,416	32,887	34,667	32,942	32,188
STT4SG-350	**Corpus**	**STT4SG-350 RAW**		**Pre-processed as described in Chapter 3.2**	**190,739**	**208**	**48**	**64**	**39**	**57**	**46,715**	**54,047**	**36,950**	**53,027**	**30**	**28**	**34**	**28**	**30**	**30**	**28**	**28,273**	**25,857**	**26,152**	**28,221**	**27,895**	**28,044**	**26,297**
	Train			Original balanced train dataset	167,646	187	43	58	35	51	41,261	47,417	32,546	46,422	27	25	31	25	27	27	25	24,966	22,559	22,824	24,925	24,593	24,716	23,063
	Valid			Original validation dataset	23,093	21	5	6	4	6	5,454	6,630	4,404	6,605	3	3	3	3	3	3	3	3,307	3,298	3,328	3,296	3,302	3,328	3,234
	All	STT4SG-350 RAW		Pre-processed as described in Chapter 3.2	217,030	235	54	71	47	63	53,337	60,683	43,951	59,059	32	32	34	32	37	36	32	30,352	29,810	26,152	32,589	34,531	32,868	30,728
Mixed Corpora																												
MIX-ALL	**Corpus**	**SDS-200 Corpus, STT4SG-350 All**		**Merged pre-processed corpus SDS-200 and STT4SG**	**339,629**	**1,632**	**646**	**387**	**142**	**457**	**119,059**	**77,255**	**59,972**	**83,343**	**93**	**365**	**90**	**168**	**196**	**66**	**398**	**33,973**	**50,474**	**29,717**	**39,644**	**40,572**	**44,628**	**85,849**
	Train	MIX-ALL Corpus	subregion	Splitted as described in Chapter 3.3 (80%)	262,293	1,100	412	229	103	356	91,076	56,895	48,528	65,794	81	278	71	134	158	51	327	29,395	37,522	23,144	32,372	33,751	35,741	70,368
	Valid	MIX-ALL Corpus	subregion	Splitted as described in Chapter 3.3 (10%)	32,592	138	54	27	12	45	10,383	7,629	5,809	8,771	4	41	7	15	20	6	45	1,298	7,473	3,701	5,091	3,928	3,539	7,562
	Test	MIX-ALL Corpus	subregion	Splitted as described in Chapter 3.3 (10%)	29,972	138	39	30	15	54	9,972	5,765	5,476	8,759	8	46	12	19	18	9	26	3,280	5,479	2,872	2,181	2,893	5,348	7,919
MIX-20000	**Corpus**	**MIX-ALL**			**140,000**	**1,376**	**505**	**286**	**130**	**455**	**33,198**	**40,000**	**27,109**	**39,693**	**93**	**365**	**90**	**168**	**196**	**66**	**398**	**20,000**	**20,000**	**20,000**	**20,000**	**20,000**	**20,000**	**20,000**
	Train	MIX-20000 Corpus	subregion		112,000	1,100	405	236	101	358	27,381	31,653	20,617	32,349	76	285	74	135	162	50	318	17,129	15,527	15,860	16,064	15,793	14,661	16,966
	Valid	MIX-20000 Corpus	subregion		13,961	138	53	26	16	43	3,009	3,449	4,532	2,971	10	33	10	20	16	9	40	1,439	1,532	1,814	1,786	1,635	3,932	1,823
	Test	MIX-20000 Corpus	subregion		14,039	138	47	24	13	54	2,808	4,898	1,960	4,373	7	47	6	13	18	7	40	1,432	2,941	2,326	2,150	2,572	1,407	1,211
	Same Speakers	MIX-ALL	subregion	Same speaker test dataset as described in Chapter 4.3.2	3,322	901	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MIX-10000	**Corpus**	**MIX-ALL**			**70,000**	**1,376**	**505**	**286**	**130**	**455**	**16,538**	**20,000**	**13,555**	**19,907**	**93**	**365**	**90**	**168**	**196**	**66**	**398**	**10,000**	**10,000**	**10,000**	**10,000**	**10,000**	**10,000**	**10,000**
	Train	MIX-10000 Corpus	subregion		56,113	1,100	406	226	100	368	13,401	15,687	11,342	15,683	77	294	71	136	155	52	315	7,946	7,830	7,710	8,420	7,977	8,509	7,721
	Valid	MIX-10000 Corpus	subregion		6,611	138	47	30	11	50	1,644	1,916	864	2,187	10	40	10	15	20	6	37	949	1,238	957	900	959	622	986
	Test	MIX-10000 Corpus	subregion		7,276	138	52	30	19	37	1,493	2,397	1,349	2,037	6	31	9	17	21	8	46	1,105	932	1,333	680	1,064	869	1,293

HA – Highest-Alemannic (cantons: GL, NW, OW, SZ, UR, VS)
WA – Western-High-Alemannic (cantons: BE, BS, FR, SO)
CA – Central-High-Alemannic (cantons: AG, LU, ZG, ZH)
EA – Eastern-High-Alemannic (cantons: AI, AR, GR, SG SH, TG)

Corpus Details					Corpus Total		Canton Group Label								Subregion Label													
Name	Type	Corpus create from	Split	Notes	Samples	Unique Speakers	Unique Speakers				Samples				Unique Speakers							Samples						
							CA	EA	HA	WA	CA	EA	HA	WA	Basel	Bern	Grison	Central Switzerland	Eastern Switzerland	Valais	Zurich	Basel	Bern	Grison	Central Switzerland	Eastern Switzerland	Valais	Zurich
Mixed Corpora Reduced																												
MIX-10000R	**Corpus**				**70,000**	**1,376**	**505**	**286**	**130**	**455**	**16,548**	**20,000**	**13,553**	**19,899**	**93**	**365**	**90**	**168**	**196**	**66**	**398**	**10,000**	**10,000**	**10,000**	**10,000**	**10,000**	**10,000**	**10,000**
	Train	MIX-20000 Train	subregion		56,000	1,100	405	236	101	358	13,152	16,000	10,949	15,899	76	285	74	135	162	50	318	8,000	8,000	8,000	8,000	8,000	8,000	8,000
	Valid	MIX-20000 Valid	subregion		7,000	138	53	26	16	43	1,662	2,000	1,338	2,000	10	33	10	20	16	9	40	1,000	1,000	1,000	1,000	1,000	1,000	1,000
	Test	MIX-20000 Test	subregion		7,000	138	47	24	13	54	1,734	2,000	1,266	2,000	7	47	6	13	18	7	40	1,000	1,000	1,000	1,000	1,000	1,000	1,000
	Same Speakers	MIX-ALL	subregion	Same speaker test dataset as described in Chapter 4.3.2	3,326	902	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
MIX-10000RS5	Train	MIX-10000R Train	subregion	Speaker separation strategy as described in Chapter 4.5.1.1	**560,000**	**1,100**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MIX-10000RD2	Train	MIX-10000R Train	subregion	Dialect separation strategy as described in Chapter 4.5.1.2	784,000	1'100	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MIX-5000R	**Corpus**				**35,000**	**1,376**	**505**	**286**	**130**	**455**	**8,236**	**10,000**	**6,804**	**9,960**	**93**	**365**	**90**	**168**	**196**	**66**	**398**	**5,000**	**5,000**	**5,000**	**5,000**	**5,000**	**5,000**	**5,000**
	Train	MIX-20000 Train	subregion		28,000	1,100	405	236	101	358	6,569	8,000	5,471	7,960	76	285	74	135	162	50	318	4,000	4,000	4,000	4,000	4,000	4,000	4,000
	Valid	MIX-20000 Valid	subregion		3,500	138	53	26	16	43	824	1,000	676	1,000	10	33	10	20	16	9	40	500	500	500	500	500	500	500
	Test	MIX-20000 Test	subregion		3,500	138	47	24	13	54	843	1,000	657	1,000	7	47	6	13	18	7	40	500	500	500	500	500	500	500
	Same Speakers	MIX-ALL	subregion	Same speaker test dataset as described in Chapter 4.3.2	3,521	914	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MIX-2000R	**Corpus**				**14,000**	**1,376**	**505**	**286**	**130**	**455**	**3,263**	**4,000**	**2,754**	**3,983**	**93**	**365**	**90**	**168**	**196**	**66**	**398**	**2,000**	**2,000**	**2,000**	**2,000**	**2,000**	**2,000**	**2,000**
	Train	MIX-20000 Train	subregion		11,200	1,100	405	236	101	358	2,616	3,200	2,201	3,183	76	285	74	135	162	50	318	1,600	1,600	1,600	1,600	1,600	1,600	1,600
	Valid	MIX-20000 Valid	subregion		1,400	138	53	26	16	43	333	400	267	400	10	33	10	20	16	9	40	200	200	200	200	200	200	200
	Test	MIX-20000 Test	subregion		1,400	138	47	24	13	54	314	400	286	400	7	47	6	13	18	7	40	200	200	200	200	200	200	200
	Same Speakers	MIX-ALL	subregion	Same speaker test dataset as described in Chapter 4.3.2	4,561	1,032	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MIX-1000R	**Corpus**				**7,000**	**1,376**	**505**	**286**	**130**	**455**	**1,625**	**2,000**	**1,383**	**1,992**	**93**	**365**	**90**	**168**	**196**	**66**	**398**	**1,000**	**1,000**	**1,000**	**1,000**	**1,000**	**1,000**	**1,000**
	Train	MIX-20000 Train	subregion		5,600	1,100	405	236	101	358	1,305	1,600	1,103	1,592	76	285	74	135	162	50	318	800	800	800	800	800	800	800
	Valid	MIX-20000 Valid	subregion		700	138	53	26	16	43	165	200	135	200	10	33	10	20	16	9	40	100	100	100	100	100	100	100
	Test	MIX-20000 Test	subregion		700	138	47	24	13	54	155	200	145	200	7	47	6	13	18	7	40	100	100	100	100	100	100	100
	Same Speakers	MIX-ALL	subregion	Same speaker test dataset as described in Chapter 4.3.2	5,046	1,083	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MIX-500R	**Corpus**				**3,500**	**1,376**	**505**	**286**	**130**	**455**	**814**	**1,000**	**690**	**996**	**93**	**365**	**90**	**168**	**196**	**66**	**398**	**500**	**500**	**500**	**500**	**500**	**500**	**500**
	Train	MIX-20000 Train	subregion		2,800	1,100	405	236	101	358	654	800	550	796	76	285	74	135	162	50	318	400	400	400	400	400	400	400
	Valid	MIX-20000 Valid	subregion		350	138	53	26	16	43	82	100	68	100	10	33	10	20	16	9	40	50	50	50	50	50	50	50
	Test	MIX-20000 Test	subregion		350	138	47	24	13	54	78	100	72	100	7	47	6	13	18	7	40	50	50	50	50	50	50	50
	Same Speakers	MIX-ALL	subregion	Same speaker test dataset as described in Chapter 4.3.2	5,259	1,099	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MIX-200R	**Corpus**				**1,400**	**1,011**	**306**	**284**	**130**	**291**	**326**	**400**	**276**	**398**	**93**	**200**	**90**	**168**	**194**	**66**	**200**	**200**	**200**	**200**	**200**	**200**	**200**	**200**
	Train	MIX-20000 Train	subregion		1,120	815	246	234	101	234	262	320	220	318	76	160	74	135	160	50	160	160	160	160	160	160	160	160
	Valid	MIX-20000 Valid	subregion		140	105	33	26	16	30	33	40	27	40	10	20	10	20	16	9	20	20	20	20	20	20	20	20
	Test	MIX-20000 Test	subregion		140	91	27	24	13	27	31	40	29	40	7	20	6	13	18	7	20	20	20	20	20	20	20	20
	Same Speakers	MIX-ALL	subregion	Same speaker test dataset as described in Chapter 4.3.2	4,001	815	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MIX-100R	**Corpus**				**700**	**649**	**167**	**190**	**100**	**192**	**167**	**200**	**134**	**199**	**93**	**100**	**90**	**100**	**100**	**66**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	Train	MIX-20000 Train	subregion		560	520	134	154	77	155	134	160	107	159	76	80	74	80	80	50	80	80	80	80	80	80	80	80
	Valid	MIX-20000 Valid	subregion		70	69	16	20	13	20	16	20	14	20	10	10	10	10	10	9	10	10	10	10	10	10	10	10
	Test	MIX-20000 Test	subregion		70	60	17	16	10	17	17	20	13	20	7	10	6	10	10	7	10	10	10	10	10	10	10	10
	Same Speakers	MIX-ALL	subregion	Same speaker test dataset as described in Chapter 4.3.2	2,574	520	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

HA – Highest-Alemannic (cantons: GL, NW, OW, SZ, UR, VS)
WA – Western-High-Alemannic (cantons: BE, BS, FR, SO)
CA – Central-High-Alemannic (cantons: AG, LU, ZG, ZH)
EA – Eastern-High-Alemannic (cantons: AI, AR, GR, SG SH, TG)

8.5 Mixed Corpora Samples per Speaker

The table below contains the number of samples per speaker within each subregion and dataset.

	Number of Recordings per Speakers																
subregion	**0-10**	**10-20**	**20-30**	**30-40**	**40-50**	**50-60**	**60-70**	**70-80**	**80-90**	**90-100**	**100-200**	**200-300**	**300-400**	**400-500**	**500-600**	**600-700**	**700-800**
MIX- 100R																	
Basel	93																
Bern	100																
Graubünden	90																
Innerschweiz	100																
Ostschweiz	100																
Wallis	66																
Zürich	100																
MIX-200R																	
Basel	93																
Bern	200																
Graubünden	90																
Innerschweiz	168																
Ostschweiz	194																
Wallis	66																
Zürich	200																
MIX-500R																	
Basel	93																
Bern	365																
Graubünden	90																
Innerschweiz	168																
Ostschweiz	196																
Wallis	66																
Zürich	398																
MIX-1000R																	
Basel	41	52															
Bern	365																
Graubünden	32	58															
Innerschweiz	168																
Ostschweiz	196																
Wallis	11	55															
Zürich	398																
MIX-2000R																	
Basel	31	15	6	41													
Bern	365																
Graubünden	32	11	4	39	4												
Innerschweiz	50	112	6														
Ostschweiz	73	123															
Wallis	11	6	13	4	32												
Zürich	398																
MIX-5000R																	
Basel	31	15	3	3	2	3	1	1	1		33						
Bern	156	147	62														
Graubünden	32	11	4	1	1	1	1	1	1	4	33						
Innerschweiz	47	39	16	7	5	49		5									
Ostschweiz	68	41	7	24	18	34			4								
Wallis	11	6	2	3	1	1	9		1	1	31						
Zürich	174	213	11														
MIX-10000R																	
Basel	31	15	3	3	2	3	1	1	1		3	26	2	2			
Bern	155	84	21	20	20	2	6	15	38		4						
Graubünden	32	11	4	1	1	1	1	1	1	1	5	31					
Innerschweiz	47	39	16	7	4	4	6	2	2	1	37	3					
Ostschweiz	68	41	7	14	3	3	5	3	7	6	36	3					
Wallis	11	6	2	3	1	1	1	1	1	1	12	24	2				
Zürich	174	82	30	15	7	7	78			1	4						
MIX-20000																	
Basel	31	15	3	3	2	3	1	1	1		3	1		1		28	
Bern	155	84	21	20	7	3	7	5	2	1	17	43					
Graubünden	32	11	4	1	1	1	1	1	1	1	5	3	6	2			20
Innerschweiz	47	39	16	7	4	4	6	2	2	1	5	2		33			
Ostschweiz	68	41	7	14	3	3	5	3	6	1	9	4		32			
Wallis	11	6	2	3	1	1	1	1	1	1	5	1	1	1	30		
Zürich	174	82	30	15	7	6	6	4	6	5	20	43					

8.6 Rented Machines from Vast.ai

Invoice	Item	Quantity	Rate	Amount
24278	Instance 6034927 GPU charge: hours * $/hr	0.13	$ 0.60	$ 0.08
24278	Instance 6034790 storage charge: hours * $/hr	0.52	$ 0.02	$ 0.01
24278	Instance 6034746 GPU charge: hours * $/hr	0.14	$ 0.25	$ 0.04
24278	Instance 6031961 GPU charge: hours * $/hr	0.26	$ 0.25	$ 0.07
24278	Instance 6031961 storage charge: hours * $/hr	2.57	$ 0.02	$ 0.05
24278	Instance 6028960 GPU charge: hours * $/hr	3.88	$ 0.25	$ 0.98
24278	Instance 6028960 storage charge: hours * $/hr	14.63	$ 0.02	$ 0.25
24278	Instance 6028960 download charge: GB * $/GB	2.65	$ 0.01	$ 0.03
24278	Instance 6034966 GPU charge: hours * $/hr	20.43	$ 0.60	$ 12.26
24278	Instance 6034966 storage charge: hours * $/hr	21.42	$ 0.09	$ 1.86
24278	Instance 6034966 download charge: GB * $/GB	11.9	$ 0.02	$ 0.24
24279	Instance 6102251 GPU charge: hours * $/hr	12.08	$ 0.26	$ 3.18
24279	Instance 6102251 storage charge: hours * $/hr	12.56	$ 0.02	$ 0.26
24279	Instance 6102251 download charge: GB * $/GB	1.63	$ 0.01	$ 0.02
24279	Instance 6150518 storage charge: hours * $/hr	0.41	$ 0.04	$ 0.02
24279	Instance 6150559 GPU charge: hours * $/hr	92.08	$ 0.36	$ 33.15
24279	Instance 6150559 storage charge: hours * $/hr	146.51	$ 0.01	$ 2.04
24279	Instance 6150559 download charge: GB * $/GB	6.6	$ 0.01	$ 0.07
24279	Instance 6150559 upload charge: GB * $/GB	0.63	$ 0.03	$ 0.02
24280	Instance 6293139 GPU charge: hours * $/hr	3.09	$ 0.30	$ 0.93
24280	Instance 6293139 storage charge: hours * $/hr	3.15	$ 0.04	$ 0.13
24280	Instance 6292692 GPU charge: hours * $/hr	1.61	$ 0.30	$ 0.48
24280	Instance 6292692 storage charge: hours * $/hr	1.61	$ 0.01	$ 0.02
24280	Instance 6293520 GPU charge: hours * $/hr	242.95	$ 0.32	$ 77.74
24280	Instance 6293520 storage charge: hours * $/hr	242.96	$ 0.04	$ 10.13
24280	Instance 6293520 download charge: GB * $/GB	22	$ 0.01	$ 0.22
24280	Instance 6293520 upload charge: GB * $/GB	2.01	$ 0.03	$ 0.06
24281	Instance 6293520 GPU charge: hours * $/hr	152.46	$ 0.32	$ 48.79
24281	Instance 6293520 storage charge: hours * $/hr	152.99	$ 0.04	$ 6.38
24281	Instance 6293520 download charge: GB * $/GB	12.5	$ 0.01	$ 0.13
24281	Instance 6293520 upload charge: GB * $/GB	27.89	$ 0.03	$ 0.84

Total: $ 200.48