Automated Speech Recognition of Swiss German dialect and Machine Translation into other languages

Master’s thesis submitted to the Constructor Institute in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Software Engineering

presented by
Iana Mikhailova

supervised by
Manuel Oriol

June 2023
I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Iana Mikhailova
Schaffhausen, 11 June 2023
Abstract

This startup thesis is dedicated to developing and comparing Speech Recognition Machine Learning models for Swiss German dialects. It is divided into two parts: business plan for the startup and technical development of Automatic Speech Recognition (ASR) system. Business plan was mainly created by my university colleague, Viktoryia Kananchuk. This part focuses on technical implementation. The research also includes a part with Neural Machine Translation (NMT) from Swiss German into Standard German. During the study, different Deep Learning architectures and text preprocessing methodologies were explored. In addition, great efforts have been spent into audio dataset collection. The main issue in Speech Recognition for dialects is that there is no official grammar or pre-trained models. This research compares the performance of fine-tuned ASR models with the own developed Speech Recognition architecture on the collected data and discovers, that novel ways of audio data preprocessing and probabilistic language models significantly improve the quality of text prediction. Our final ASR model demonstrates the 14 % Character Error Rate (CER) and 35 % Word Error Rate (WER). The Sequence to Sequence model with Attention mechanism for NMT task shows the 58.9 BLEU score, trained on the textual dataset of 38 000 sentence pairs.
I would like to thank my startup colleague and best friend Victoryia Kananchuk for being a great support during the development of this startup, and my supervisor Manuel Oriol for believing in this thesis, giving us inspiration, and pointing our researches in the right direction.
# Contents

ABSTRACT ............................................................................. 3

ACKNOWLEDGEMENTS ....................................................... 4

ABBREVIATIONS ................................................................. 8

INTRODUCTION .................................................................... 9

1 SUBJECT AREA OVERVIEW ............................................. 11
  1.1 Methodologies for ASR and NMT ................................. 11
    1.1.1 Convolutional Neural Networks .............................. 12
    1.1.2 Recurrent Neural Networks ................................. 12
    1.1.3 LSTM and GRU ................................................. 14
    1.1.4 Sequence to sequence and Attention .................... 16
    1.1.5 Transformers ................................................... 19
    1.1.6 Language Model ................................................. 19
  1.2 Related works ............................................................... 20
    1.2.1 Wav2Vec2 ........................................................ 20
    1.2.2 Baidu DeepSpeech and DeepSpeech 2 ................. 20
    1.2.3 SpecAugment .................................................... 21
    1.2.4 Whisper .......................................................... 21
    1.2.5 KenLM library ................................................... 21
    1.2.6 Transformer in «Attention is all you need» ............ 21
  1.3 Our approach ............................................................... 21
  1.4 Potential applications .................................................... 23

2 AUDIO DATA COLLECTION .............................................. 24
  2.1 Own collected data ..................................................... 24
  2.2 ArchiMob corpus UZH ............................................... 24
  2.3 Bern Parliament Debates ............................................. 25
  2.4 Digital representation of audio recording .................... 25
  2.5 Conclusions ............................................................... 26
3 TEXTPROCESSING .................................................. 28
  3.1 Basic types of text preprocessing ............................ 28
  3.2 Word embeddings ............................................. 29
    3.2.1 One-hot encoding and bag of words ................. 30
    3.2.2 N-grams .................................................. 30
    3.2.3 TF-IDF ................................................... 30
    3.2.4 Word embeddings ........................................ 31
  3.3 Byte pair encoding .......................................... 31
  3.4 Metrics for text similarity .................................. 32
    3.4.1 Levenshtein distance .................................. 33
    3.4.2 Character Error Rate ................................... 33
    3.4.3 Word Error Rate ........................................ 33
    3.4.4 BLEU score .............................................. 34
  3.5 Conclusions .................................................. 35

4 SPEECH RECOGNITION EXPERIMENTAL ARCHITECTURE .... 36
  4.1 Residual CNN for spectrograms ............................ 36
  4.2 Regularization and Activation ................................ 37
  4.3 Bidirectional GRU ............................................ 37
  4.4 CTC loss function ............................................ 38
  4.5 Beam Search Technology ..................................... 40
  4.6 Language Model ............................................... 40
  4.7 Conclusions .................................................. 40

5 NEURAL MACHINE TRANSLATION ARCHITECTURE .......... 41
  5.1 Word Embeddings ............................................. 41
  5.2 GRU ........................................................... 41
  5.3 Attention Decoder ............................................ 42
  5.4 Loss function, Regularization and Activation ............ 42
  5.5 Misspelling correction ...................................... 42
  5.6 Conclusions .................................................. 43

6 SET UP AND RESULTS OF THE EXPERIMENTS .................. 44
6.1 Set up for ASR .................................................. 44
  6.1.1 Source code ............................................. 45
  6.1.2 Data format ............................................. 45
  6.1.3 Optimizer and Learning rate scheduler .......... 45
6.2 ASR Experiments ............................................ 46
  6.2.1 Results of the own built model on the whole dataset . 46
  6.2.2 Results of the own built model on the whole dataset with KenLM ........................................ 49
6.3 Set up for Machine Translation ............................. 50
  6.3.1 Data format ............................................. 51
  6.3.2 Optimizer ................................................. 51
6.4 Machine Translation Experiments .......................... 51
  6.4.1 Misspelling correction results ....................... 51
  6.4.2 NMT model results .................................... 52

CONCLUSION ......................................................... 54

BIBLIOGRAPHIC REFERENCES .................................. 56

APPENDIX .......................................................... 60
Abbreviations

ASR — Automatic Speech Recognition
NMT — Neural Machine Translation
CER — Character Error Rate
WER — Word Error Rate
ML — Machine Learning
DL — Deep Learning
RNN — Recurrent Neural Networks
CNN — Convolutional Neural Networks
GRU — Gate Recurrent Units
LSTM — Long Short Term Memory
NLP — Natural Language Processing
Seq2seq — sequence to sequence Deep Learning model
BLEU — Bilingual Evaluation Understudy Metrics
Introduction

Swiss German is a set of a variety of dialects spoken by 5 M people in the German-speaking part of Switzerland, which is very different from Standard German in terms of both pronunciation and spelling, it represents almost a separate language. Swiss citizens are very proud of this distinction, however, such difference generates problems for both Swiss citizens texting in their dialect and foreigners understanding Swiss citizens. It also does not allow existing Speech Recognition Machine Learning models, trained on Standard German dataset, to be reused and finetuned on a Swiss German dataset very accurately. In addition, there is not much transcribed audio data present in the Swiss German. For these reasons, online video subtitles, transcribing audio messages in social networks, and auto correction T9 for Swiss German are not available. There is much evidence that there is a demand for the solution to the stated problem and many research teams at Swiss universities are currently struggling with it. Thus, this thesis is dedicated to developing competitive Automatic Speech Recognition (ASR) and Neural Machine Translation (NMT) models that are to convert Swiss Speech into text and to translate it into Standard German.

Lack of Swiss German audio data inspired us to collect recordings with transcriptions from native speakers and such online platforms, as YouTube. We also explored the existing datasets that are most of the time not properly labeled and we needed to label them manually. With the data we developed an ASR model with an impressive accuracy, conducted a validation with the help of Swiss German native speakers and integrated it with TelegramBot to demonstrate its potential application in the nearest future. The model consists of different Deep Learning (DL) layers, comprises various state of the art hacks for optimization and sound preprocessing, and operates with our own probabilistic language model.

Due to some morphological and semantic differences between different languages, it is necessary to choose methods for text preprocessing wisely. For example, the German language is distinguished by its long compound words, which
are formed in most cases by simple concatenation. Nevertheless, for all languages, a transformation is eventually made from words or phrases into the space of meanings (word embeddings), from which the translation takes place. This technology is one of the most advanced in the field of Neural Machine Translation and is also applied in this thesis.

On the one side, we achieved desirable progress on the topic, on the other side, there is still much room for improvement and we should not stop there.

The workflow is divided into several parts, which can be traced in the chapters of this thesis:

1. Subject area overview with related state-of-the-art solutions;
2. Dataset description and audio preprocessing methods;
3. Ways of text preprocessing for both ASR and NMT tasks;
4. Description of discovered and built ML models for ASR task;
5. Description of discovered and built ML models for NMT task;
6. Results of the experiments;
7. Conclusion.

Subject area overview describes Deep Learning Networks, used for the final model creation and lists several advanced solutions in related studies. Second chapter explains how the data was collected and how the audio recordings can be transformed into tensors. Next chapter covers the topic of possible text preprocessing for both the ASR and NMT problems. In the two following chapters our final ML architectures for the ASR and NMT tasks are illustrated in detail. In the seventh chapter, we present results obtained by those models. At last, we conclude this thesis with the brief results and future work discussions.

The code is written in Python programming language with such dependencies as: Torchaudio, Pytorch, Sklearn, Pandas, Numpy, Nltk. See link to the source and brief files description in the Appendix.
1 Subject area overview

Speech Recognition, also known as Automatic Speech Recognition (ASR), is a capability of a computer program to convert human speech into a written text. While there are a lot of solutions for verbal recognition, the most advanced ones apply Machine Learning and Artificial Intelligence. This thesis focuses on the most recent prominent Deep Learning solutions to the problem and compares different methodologies and optimizations, either adapted in a well-known NLP tool such as ChatGPT. Since Swiss German is not the official language in Switzerland, we present some ML applications for Neural Machine Translation (NMT) into Standard German, thereby our Speech Recognition model can transcribe Swiss speech directly in Standard German.

1.1 Methodologies for ASR and NMT

There are several widely-used methodologies and Deep Learning architectures for ASR and NMT tasks:

1. Convolutional Neural Networks;
2. Simple Recurrent Neural Networks;
3. LSTM, GRU;
4. Sequence-to-sequence Models with Attention mechanisms;
5. Transformer-based Machine Learning models;

Since Spectrograms (pictures of sound) are often used as the audio representation, Convolutional Neural Networks need to be in place as they can process the pictures. Next two points represent the Deep Neural Networks applied for the sequential data like text, audio, video, where the context and sense depend on the sequence of words, sounds, pictures respectively. Seq2Seq Attention and Transformer-based models, such as Whisper and Wav2Vec2, are the most complex models based on the word embeddings which give a huge leap forward in ASR and NMT. At last, Probabilistic Language Models allow to predict context better as they
provide probabilities for the words occurred in a text (the more frequently the word appears in the text, the more important it is).

Here some background information regarding the listed architectures is presented.

1.1.1 Convolutional Neural Networks

Convolutional Neural Networks (ConvNet) is a class of Neural Networks that specializes in processing data that has a grid topology, such as images. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid, where each cell contains visual data: brightness and color.

A CNN usually has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

1. The convolutional layer is the main building block of CNN. This layer performs a dot product between two matrices, where one matrix is a set of learning parameters, otherwise called the core, and the other matrix is a limited part of the perceiving field. The core slides over the height and width of the image, creating a two-dimensional representation of the image, known as an Activation Map.

2. The pooling layer replaces the output of the network at specific locations, retrieving summary statistics of the nearest outputs. This helps to reduce the spatial size of the representation, which reduces the amount of computation and weights required.

3. A fully connected (linear) layer helps to map the dimensions between the input and the output.

1.1.2 Recurrent Neural Networks

Recurrent Neural Network (RNN) basically processes pieces of data information sequentially and preserves the context of the previous pieces in the sequence. Data is converted to digital representations — for instance, text will be divided into words which will be then converted into vectors or tensors, also called *states*. Here we explain, why the use of the simple RNNs is not always sufficient for good results, and introduce GRU or LSTM — more complex variations of RNNs.
One link for updating the state looks as follows [14] (fig. 1.1). We get the previous state $z_t$ as input, then we want to add information about the new input word («word» — this is an element of some sequence here, not necessarily text), we simply concatenate (add $x_t$ from below) it with the previous state. The resulting state has a dimension different from the initial one. Then you can get a new output state by applying the $W$ transformation to it. However, to complicate the system, nonlinear dependencies are still needed: we apply the activation functions and obtain the output state.

![Figure 1.1 – RNN layer [37]](image)

In this case, the weight matrix $W$ will be the same for all layers of the network, since sequences of all possible lengths must be processed in the recurrent neural networks.

![Figure 1.2 – RNNs sequentially [37]](image)

Formal description of the state update process:
$h_0 = 0$ — init state

$h_1 = \sigma([W_{hid}(h_0, x_0)] + b)$$

$h_2 = \sigma([W_{hid}(h_1, x_1)] + b) = \sigma([W_{hid}(\sigma([W_{hid}(h_0, x_0)] + b), x_1)] + b)$$

$h_{i+1} = \sigma([W_{hid}(h_i, x_i)] + b)$$

P(x_{i+1}) = softmax([W_{out}, h_i] + b_{out})$

As we can see, during the backpropagation method, the error will move along the RNN from the last step to the earliest one. With a sufficiently small initial gradient (say, less than 0.25) to the third or fourth module, the gradient will almost disappear (since, according to the rule of the derivative of a complex function, the gradients will be multiplied), and then the hidden states of the very first steps will not be updated. This «vanishing gradient» problem is resolved by adding some filters in GRU and LSTM. Since human speech often represents a long sequential data and we do not want to lose the context of earlier words and to transcribe more accurately, vanilla RNN should be replaced by GRU or LSTM.

1.1.3 LSTM and GRU

LSTM (Schmidhuber, 1997)[13] — This is the most commonly used type of recurrent neural networks. From the name Long short term memory we see that this type of networks provides for long-term memorization of specific information. This type of network is designed to eliminate unnecessary information in some sequences. For example, spoken language is a sequential data structure that can be processed using RNN, but which often contains interjections that do not carry
any useful information (for example, during an interview, a person says: «Mmm», «Well», «Em» etc.). Or in a regular text, either value judgments or adverbs that do not carry much meaning can also be found.

That is, the idea of LSTM is to include a «forget» or «remember» classifier for each token and remove or add this token to long-term memory accordingly. Note that this is a binary classifier!

![LSTM cell representation](image)

Figure 1.4 – LSTM cell representation [29]

![Definitions](image)

Figure 1.5 – Definitions [29]

The figure 1.6 demonstrates the main idea: «X» is the removal or addition of information with some classifier weights, «+» is the addition of the transformed current information.

![Long-term memory](image)

Figure 1.6 – Long-term memory [29]
Gated Recurrent Unit (GRU) (fig. 1.8) is a simplified modification of LSTM. It combines «forgetting gate» and «input gate» into a single «gate of renewal». It also merges the cell state and hidden state and makes other changes. The resulting model is less popular than LSTM, but outstands LSTM in terms of speed.

\[
\begin{align*}
    z_t &= \sigma (W_z \cdot [h_{t-1}, x_t]) \\
    r_t &= \sigma (W_r \cdot [h_{t-1}, x_t]) \\
    \tilde{h}_t &= \tanh (W \cdot [r_t \ast h_{t-1}, x_t]) \\
    h_t &= (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t
\end{align*}
\]

Figure 1.8 – GRU [29]

1.1.4 Sequence to sequence and Attention

The Sequence to Sequence (Seq2seq [19]) model can be thought of as two blocks, an encoder and a decoder, connected by a vector, which is called «context vector».

1. Encoder: The encoder processes each token in the input sequence. It tries to write all the information about the input sequence into a vector of fixed length, i.e. «context vector». After passing all the tokens, the encoder passes this vector to the decoder.

2. Context vector: The vector is constructed in such a way that it is expected to encapsulate the entire value of the input sequence and help the decoder make accurate predictions. We will see later that these are the final internal states of our encoder block.
3. Decoder: The decoder reads the context vector and tries to predict the target sequence token from the token.

![Diagram of Seq2seq model]

Figure 1.9 – Simple view of Seq2seq [28]

Part of the encoder is the LSTM cell. It is fed in and tries to encapsulate all its information and store it in its final internal states $h_t$ (hidden state) and $c_t$ (cell state). The internal states are then passed to the decoder side, which it will use to try and create the target sequence. This is the «context vector» we talked about earlier.

All outputs at each time step of the encoding part are excluded.

The decoder block is also an LSTM cell. The main thing to note here is that the initial states $(h_0, c_0)$ of the decoder are set to the final states $(h_t, c_t)$ of the encoder. They act as a context vector and help the decoder create the desired target sequence. The decoder now operates in such a way that its output at any time step $t$ must be the word $t^{th}$ in the target sequence (fr. «ravi de vous rencontrer» (fr.)).
What if the input sentence is long, can one vector from the encoder contain all the relevant information provided to the decoder? Is it possible to focus on a few relevant words in a sentence when predicting a target word, rather than a single vector containing information about the entire sentence? Attention mechanisms help solve the problem.

The main idea of the attention mechanism is not to try to learn a single vector representation for each sentence, but instead pay attention to specific input vectors of the input sequence based on attention weights. At each stage of decoding, the decoder is told how much "attention" to give to each input word, using a set of attention weights. These attention weights provide the decoder with contextual information for translation.

The Attention layer consists of an Alignment Layer, Attention weights, and a Context vector [26]:

1. Layer Alignment. The estimate is based on the latent state of the previous decoder, \( s_{i-1} \), just before the prediction of the target word and the latent state, \( h_j \) of the input sentence. The decoder decides which part of the original sentence it should pay attention to, instead of the encoder encoding all the information in the original sentence into a fixed length vector.

2. Attention weights. We apply a softmax activation function to Alignment to get Attention weights. The Softmax activation function will receive probabilities that sum to 1. This will help represent the attention weight for
each input sequence. The higher the attention weight of the input sequence, the higher will be its influence on the prediction of the target word.

3. Context vector. Used to compute the decoder’s final output. The context vector \( c_i \) — is the weighted sum of attention weights and encoder latent states \((h_1, h_2, \ldots, h_{tx})\).

Computing different types of attention score formally:

\[
\begin{align*}
    h_i^T h_s & \quad (1.1) \\
    h_i^T W_a h_s & \quad (1.2) \\
    v_a^T \tanh W_a [h_i^T h_s] & \quad (1.3)
\end{align*}
\]

1.1.5 Transformers

Transformer is a Seq2seq model, as it also consists of such blocks as Encoder and Decoder.

But the new interesting thing is a Multi-head Attention. This is a special new layer that allows each input vector to interact with other vectors (words, e.g.) through the Attention mechanism, instead of passing a hidden state like in an RNN or neighboring words like in a CNN.

Multi-head Attention is given Query vectors as input, and several Key and Value pairs (in practice, Key and Value are always the same vector). Each of them is transformed by a trainable linear transformation, and then the dot product \( Q \) with all \( K \) is calculated one by one, the result of these dot products is run through softmax, and with the resulting weights, all vectors \( V \) are summed into a single vector. This formulation of Attention is very close to the previous one. The only difference is that several of such attentions are trained in parallel, i.e. several linear transformations and parallel dot products/weighted sums. At last, the result of all these parallel attentions is concatenated, once again run through a trained linear transformation and goes to the output.

1.1.6 Language Model

Language models represent a set of the probabilities a word (or a string of N characters, called N-gramms) appear in the text. In the language model each N-gram
has a corresponding number which stands for the probability, and it is computed by \( \log_{10} \). This representation is particularly useful for ASR and NMT.

### 1.2 Related works

Here are some of the recent results for the solutions to the ASR and NMT problems.

#### 1.2.1 Wav2Vec2

Recent related studies include exploration of the Transformer-based pre-trained cross-lingual Wav2Vec2-XLS-R model for Swiss Speech translation and classification (Stucki, Randjelovic; 2022) [3], where 16.8% WER was achieved transcribing Swiss Speech directly into Standard German. They used 2100 hours of unlabeled Swiss Speech and pre-trained the model on it.

#### 1.2.2 Baidu DeepSpeech and DeepSpeech 2

The result for the 58.6% WER for the same task, as the previous one, was achieved at the university of Essen (Aashish Agarwal; 2020) who applied Baidu DeepSpeech model (Baidu; 2015 [15]) and KenLM Language Model for Standard German. The model is trained on the dataset of approximately 120 hours of transcribed Swiss German Speech. The Baidu DeepSpeech and DeepSpeech2 are based on the Recurrent Neural Networks and the Convolutional Neural Networks for audio spectrograms. DeepSpeech 2 is more advanced and implements preprocessing of audio data into spectrograms, applies Data Augmentation with Gaussian noise, then feeds it into a Deep Network (DNN), consisting of 2-3 CNNs and 3-6 RNNs.

The experiments regarding Swiss German Speech to text Recognition conducted at the University of Zurich (Tannon Kew; 2020)[4] show WER rate of 48%. The ML model was trained on the ArchiMob 70 hours corpora, where the audio is not very well transcribed.

Another research into Deep Neural Network acoustic models for ASR (Abdel-rahman Mohamed; 2014)[12] proves that recent innovations in RNN allow ASR models to perform better, than earlier Gaussian Markov Models.
1.2.3 SpecAugment

SpecAugment methodology of data augmentation through processing CNN (Google Brain; 2019) [1] also demonstrated promising results with the 6.8% WER score on the model based on the high quality 300 hours LibriSpeech dataset in English. The methodology is based on the deformation of the time series and masking blocks of time and audio power spectrum dimensions.

1.2.4 Whisper

Whisper (Radford; 2022)[25] is an Automatic Speech Recognition (ASR) system trained on 680,000 hours of multilingual supervised data collected from the Internet. It comprises Spectrogram processing, CNNs, RNNs and Transformers with Cross Attention. Using such a big and diverse dataset results in an increased robustness to accents, technical language, and noise. In addition, it not only allows transcription in multiple languages (but no Swiss German), but also implements translation from those languages into English. Whisper achieves WER of 1.4% on the English language.

1.2.5 KenLM library

The KenLM (K. Heafield; 2011)[39] library is a fast queries language model, which can be pretrained on any textual dataset for any language.

1.2.6 Transformer in «Attention is all you need»

Machine Translation recent solutions are based on the Seq2Seq modeling (Bahdanau, Cho, Bengio; 2016)[23] and Transformers. In the paper «Attention is all you need» (Vaswani; 2017)[24] researchers shared the impressive results of an NMT Transformer-based model that achieved BLEU score of 41.8 on the English-to-French corpus of 50 million words.

1.3 Our approach

Our task is different from the explored ones in the related works for ASR, since it aims to transcribe Swiss Speech into Swiss written text that is different from Standard German. Another issue is, there is no common pronunciation and spelling in Swiss German, as it varies from region to region in the country (fig. 1.11).
Although there are open source tools for ASR task accomplishment, we must emphasize, that they are based on the official languages with the great amount of data: 300 000 hours for transcribed datasets in English, 100 000 hours of data for transcribed datasets in German, etc. For Swiss German, there are only about 2000 hours of unlabeled data and 150 hours of labeled data in the Internet. Consequently, we need to focus on the Swiss German data collection, transcription, Swiss German Language Model development and accurate text preprocessing. Furthermore, we choose the most appropriate ways of ML model construction, measurement of its quality and justify our choice in further chapters.

To sum up, our Speech Recognition tool development can be divided into following steps:

1. Audio and transcriptions data collection and preprocessing into a digital representation;
2. Machine Learning architecture development;
3. Text preprocessing for transcriptions;
4. Development of a Swiss Language Model;
5. Definition and application of appropriate text similarity metrics.
The solutions for Translation task are also investigated and the Attention Seq2Seq is applied in this thesis. For this task all we need is a dataset of Swiss German to Standard German assigned sentence pairs.

1.4 Potential applications

Our solutions to the Speech Recognition problem from Swiss German into textual representation as well as the Machine Translation problem from Swiss German to Standard German can be integrated with many services. First of all, ASR tool can be applied for online subtitles generation on YouTube, Netflix, online Zoom and GoogleMeet meetings, etc. Moreover, it can be easily adapted by the Swiss German speakers for the audio messages transcription into text. Another application is seen for people using Voice assistants, such as Siri. The NMT tool seems valuable for the both foreigners who want to integrate into Swiss culture and to learn the dialect, and the Swiss German native speakers who can apply it in a way of generating responses in a formal (German) language while being used to text in their dialect.
2 Audio data collection

Our dataset has been collected over a period of half a year and consists of approximately 80 hours of high quality audio recordings of Swiss German from different regions in Switzerland. All of the composed parts as well as a possible way of converting them into a digital representation are described below.

2.1 Own collected data

This part has been collected from YouTube and from native Swiss speakers we approached for our project.

The YouTube data includes in total about 30 minutes of Swiss German speech from different regions (Bern, Zurich, Aarau), each audio recording represents an utterance of 3-10 seconds on average.

Moreover, we approached 8 native speakers in person (6 male and 2 female from Zurich, Thurgau, Schaffhausen, and Bern) and collected 2 hours of Swiss German speech recordings in .wav extension with transcriptions. As basic sentences, tatoeba.org German sentences were used. Our interviewees were asked these sentences to be transcribed into Swiss German and then recorded. There might be an issue that Swiss usually write in Swiss German the way they feel, as long as they do not have any common spelling for Swiss German unlike High German.

Here is one example of collected utterances:

High German from tatoeba.org: «Du wärst überrascht, was man in einer Woche lernen kann.»
Swiss German: «Du bisch überrasch gsi, wasmer in eini wuche lerne chan.»

2.2 ArchiMob corpus UZH

This dataset corpus consists of approximately 72 hours of unlabeled Swiss German WW2 interviews from different regions in .wav extension, data was inquired from a University of Zurich. Since it is unlabeled, much effort was put on transcribing
the utterances manually, and in total we have about 4 hours of transcribed Swiss German from this corpus.

Here is one example of collected utterances:

Swiss German: «aber was es wirklich gsii isch han i mit ölfì nid erfasst.»

In this example, double «i» occurs many times in the corpus as well as different spelling peculiarities. This causes problems of rarely recognized words by ML model and lowers the quality of our predictions since the model considers the rare words as outliers. Possible solution would be to train a best model without this corpus.

2.3 Bern Parliament Debates

This corpus consists of around 70 hours of Bern Parliament Debates (MIT license) in .flac extension and consists substantionally of Bern dialect utterances. Data was labeled in Standard German and was translated into Bern dialect spelling. This part of our dataset must have the biggest impact on the model and was used separately from other parts in a model to get more accurate predictions. One disadvantage of the dataset is, it is still not labeled very precisely: there are cases where the audio utterance is shorter than its transcription and vice versa, however, the exact percentage of those cases is unknown and we will consider such utterances as outliers.

Here is one example of collected utterances:

German: «wir sollten das auch mit dem staatshaushalt nicht machen.»
Swiss German: «mirsöttedasoumitemstaatshuushalt nid mache.»

2.4 Digital representation of audio recording

In total, the dataset consists of nearly 80 hours of labeled data. One unit of data is a transcribed (in German or in Swiss German) audio recording with the average duration of 5-10 seconds. The extension of one recording is either .flac or .wav, although other extensions can also be processed in the source code (see Appendix).

For ASR both audio and text should be transformed into some digital representation.
Audio can be converted into a Spectrogram (Abdel-Rahman Mohamed, 2014) [12], that represents a «picture of sound» using filter banks that work better than Mel-Frequency Cepstral Coefficients (MFCCs). Mel-frequency cepstrum (MFC) is a representation of the short-term audio power spectrum based on a linear cosine transform of a logarithmic power spectrum over a non-linear mel frequency scale. Filter banks can be computed via several steps: pre-emphasis where the amplitudes are being amplified, division into frames, Fourier Transformation, calculation of a power spectrum and the filter banks themselves (Haytham Fayek, 2016) [9]. The Mel-scale aims to draw the non-linear perception of sound by the human ear, as it better distinguishes between lower frequencies than the higher frequencies. The output can be seen on the picture:

![Figure 2.1 – Spectrogram [9]](image)

Data augmentation is also important: utterances data can be extended by applying such techniques as changing pitch, amplitude, etc. However, in this thesis we considered a recent Specaugment technique (Google brain, 2019) [1], which is based on the deformation of the time series and cutting out (masking) blocks of time and Mel frequency dimensions.

Such kind of feature extraction and augmentation from an utterance are implemented in torchaudio library in Python, which we used for this thesis.

### 2.5 Conclusions

The collected dataset is relatively small and labelled in different spellings because of the difference between dialects, which causes issues for model testing as it can overfit for a very specific vocabulary. The text data is also not enough to create a comprehensive language probabilistic model as it predicts words of specific area
(biased for Bern dialect and mostly formal language). This issues may be partially resolved by collecting new data and audio data augmentation with the Specaugment methodology.
3 Text processing

Since the study includes parts with audio-to-text alignment and text-to-text translations, we consider some text preprocessing methodologies, compare them and choose the most appropriate and advanced ones for the ASR and NMT tasks. In addition, the quality of the developed tool is measured with the help of text similarity metrics, which are also explained in this chapter.

Sample sentences from our dataset in Swiss German and High German respectively:

Swiss German 1: «da stah i itze, i arme tor!»
High German 1: «da steh ich nun, ich armer tor!»
Swiss German 2: «aus zythorizont für d realisierig hei di planer damals ds Jahr 2001 vor gsch.»
High German 2: «als zeithorizont für die realisierung sahen die planer damals das jahr 2001 vor.»

For both Speech Recognition and Translation problems text should be converted into a digital representation. However, for ASR we need to predict each character or combinations of 2-3 characters separately, hence, all figures in the utterances (such as a year «2001» in the example above) should be transcribed as a text («Zweitausendeins»). This is accomplished with a function in the source code, where all of the figures are first found in each sentence with the help of re, Python regular expressions library, and replaced with their German spellings with the help of Python num2words library.

3.1 Basic types of text preprocessing

To the well known and widely-used text preprocessing part we will attribute lemmatization and stemming.

1. Lemmatization is a word normalization. For instance, past perfect tense of a verb «gegessen»(eaten) to the infinitive — «essen»(eat). Similarly with nouns: derive to the nominative case singular form (if possible).
2. Stemming — cut prefixes, suffixes or endings to derive root meaning. For instance, word «Kätzchen» (soft form of the word cat) becomes «Katz» (cat). Both techniques have significant disadvantages, that cannot be evaded with existing linguistic methods:

1. With lemmatization there are situations when a normalized word form has several different meanings. For example, German noun "Das Leben” (life) and verb "leben” (live) will turn into “leben” after lemmatization, although they have different grammatical applications.

2. In stemming same root can have different meanings in different words. For example, German noun "die Universität” (university) and noun "das Universum” (universe) will turn into “univers”, thus, in the context of a text the sense may be lost.

Linguists also composed a so-called WordNet (Miller, 1998) — graph with nodes, where synonymic words are stored. The problem of such Net is obvious: it is not comprehensive because of the constant language update and appearance of new words.

Nevertheless, word normalization is a good practice, for this purpose, Python NLTK library based on the WordNet is applied in this thesis.

3.2 Word embeddings

To perform NMT, it is necessary to use a vector representation of words or phrases, that is, the translation of a part of a sentence into the so-called space of meanings (fig. 3.1), so that words with close context meanings lay close on the coordinate plane.

This method is called word-embeddings (T. Mikolov; 2013) [21], it is widely-used and often outperforms less recent ways of text digital representations.
3.2.1 One-hot encoding and bag of words

There are many ineffective options for text representation in the form of separate semantic units (words, for example). The simplest example is one-hot encoding, where a word is presented as a dictionary element — a dictionary dimension vector \( v_{\text{onehot}} = (0000100) \) with 1 at the position of the word number in that dictionary (here \( v_{\text{onehot}} \) lays in dictionary of 7 words).

Another text representation is «bag of words», which assumes that the meaning of the text does not depend on word order. It is clear that such representation is incorrect (it only takes place for simple tasks like determination of the main topic in the text).

3.2.2 N-grams

The next solution is to represent the text in the form of n-grams (a subsequence of n words), which allows to detect connections between words, while removing not very informative n-grams (for example, those that include interjections or prepositions). But it is obvious that this approach increases the dictionary dimension by several orders of magnitude, depending on n.

3.2.3 TF-IDF

Further, a tool such as TF-IDF (term frequency — inverse document frequency) was developed. This statistical metric estimates how often a token (word) occurs in a given document and how often it occurs in others. The tool is convenient as it allows to separate different texts by topic and understand how much a single word
makes sense in principle, because there are such parts of a sentence as prepositions.

\[
tf(t, d) = f_{t,d}
\]

\[
idf(t, d) = \log \frac{N}{|\{d \in D : t \in d\}|}.
\]

Here \( f_{t,d} \) — token frequency \( t \) in the document \( d \), \( N \) — amount of the document in the dataset, \( |\{d \in D : t \in d\}| \) — amount of the documents where token \( t \) is found.

### 3.2.4 Word embeddings

Word embeddings [30] is the most advanced tokenization tool. For understanding, we introduce a conditional probability (the relationship between words) Pointwise Mutual Information (PMI) [30].

\[
PMI = \log \frac{p(u, v)}{p(u)p(v)} = \log \frac{n_{uv}}{n_un_v},
\]

where \( n_{uv} \) is the number of times we encountered words together in the same window \( u \) and \( v \). This score allows to create a high-dimensional matrix for a bunch of words, which can be reduced to a low-dimensional context vector, which is called embedding, using Singular Value Decomposition (SVD) and matrix decomposition. Such vectors are useful at prediction by context.

The implementation of embedding layers has already been presented in the Pytorch module (torch.nn.embedding), which is used in this thesis.

### 3.3 Byte pair encoding

As mentioned above, German can be sometimes characterized by word formation by concatenation of simpler words. For example, the word «Apfelkuchen» is literally translated into English as «Apple pie». There are a lot of such examples in the German language, so taking a whole word as a token in German does not always make sense: it is better to break the word into subwords in order to get a more accurate translation.

Such a partition can be achieved using the BPE (Byte pair encoding) [20] technique, according to which the text is initially divided into letters (characters), then the frequency of each letter in the text is examined, then the letters are
concatenated into pairs and the frequencies of the pairs are examined, then they try to form triples letters and so on, until you get divisions into semantic parts of the sentence — parts of compound words (in the example «Apfelkuchen» is divided into «Apfel» — «Apple» and «Kuchen» — «Pie»), words and even phrases.

We used the BPE implementation from the youtokentome library from Vkontakte.

### 3.4 Metrics for text similarity

Measuring the performance of ASR and NMT is about the comparison of two texts: predicted by a model and real (target) one.

At first, we introduce two metrics used to evaluate Speech Recognition model output, namely Character Error Rate (CER) and Word Error Rate (WER) [8]. The intuition behind both of them is based on the accuracy score, which, however, does not provide enough granularity to assess performance effectively. Hence, we should focus on how the predicted text is different from the ground truth (transcribed). First idea is to check the amount of characters misspelled, but they could be misspelled in three possible ways (fig. 3.2):

- **Substitution error**: Misspelled characters/words
- **Deletion error**: Lost or missing characters/words
- **Insertion error**: Incorrect inclusion of character/words

![Figure 3.2 - Misspelling errors](image)

To answer the question how to measure the extent of errors occurred in a text, we need to remind of a Levenstein distance.
3.4.1 Levenshtein distance

The Levenshtein distance is a score illustrating the distance between two strings. It shows minimal amount of single character edits that is necessary to transform one string into another. For example, the Levenstein distance between words «Mitte» and «Smith» is 4.

3.4.2 Character Error Rate

Character Error Rate (CER) is based on a Levenshtein distance and is calculated using $S$, $D$, $I$ as Substitution, Deletion, Insertion errors and $N$ as a number of character in the target sentence in a following way:

$$CER = \frac{S + D + I}{N}.$$ 

It is applied to every pair of all predicted outputs and target sentences and then the mean value is calculated, which is considered as a $CER$ for the ASR model. Thus, $CER$ illustrates the ratio of characters which were incorrectly predicted in the ASR model.

However, $CER$ can exceed 1.0 if there are many insertions in the text: for instance, for the target sentence «ABC» and a predicted sentence «ABCDEFGH» $CER$ will be $\frac{5}{3} = 1.67$. Hence the normalization is needed: the $C$ value which represents the number of correct characters is included:

$$CER_{\text{normalized}} = \frac{S + D + I}{S + D + I + C},$$

Hereby, $CER_{\text{normalized}}$ will always be less or equal to 1.0. Although, good $CER$ is considered to be 0.01-0.02, it must be observed individually, as in our case when the data is not well labelled since Swiss German does not have a proper spelling.

3.4.3 Word Error Rate

Word Error Rate (WER) is calculated as a CER but on a word level:

$$WER = \frac{S_w + D_w + I_w}{N_w}.$$ 

WER and CER are quite correlated, however, WER even in good models is expected to be much higher, than CER. In the example:
**Target:** «Meine Name ist Paul»

**Predicted** «Meeine Nam is Paul»

CER rate would be 0.06 while WER would be 0.75.

### 3.4.4 BLEU score

As for machine translation, there are many metrics used to assess the quality of predicted translation from one language to another. The most reliable would be, of course, the assessment by real people, who compares how close the ML predicted text is to the text generated by a real human. Obviously, such statistics collected from real people is a long labour intensive process, since such people must be fluent in both languages from explored sample of languages. Time taken to compare several texts is also a weak point here. For these reasons, for quality estimation other metrics were invented. For instance, BLEU, ROUGE, NIST, etc. We introduce the first one since it is the simplest in understanding and is the least computationally expensive, in addition, BLEU is widely used for Neural Machine Translation in texts without major limitations.

BLEU (Bilingual evaluation understudy, IBM T. J. Watson Research Center, 2002) [33] is a metric, developed for two texts (sentences) similarity estimation. It is convenient, specially, for Neural Machine Translation, as it shows how different is the computer translation from a translation generated by a human.

\[
\text{BLEU} = \text{brevity penalty} \cdot \left( \prod_{i=1}^{n} \text{precision}_i \right)^{\frac{1}{n}} \cdot 100%,
\]

where

- *brevity penalty* — penalty for a big difference between target and predicted sentences (we want to avoid significant shortenings and extending of sentences),
- *precision*$_i$ — accuracy of matched i-gramm.

We demonstrate how it works on the following example with one target sentence and two generated ones:

**Target:** «French officials are responsible for airport security»

**System 1:** «French officials responsibility of airport security»
System 2: «airport security french officials are responsible»

Table 3.1 – Matrix for comparison 2 generated with target

<table>
<thead>
<tr>
<th>Metric</th>
<th>System 1</th>
<th>System 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision(1gram)</td>
<td>3/6</td>
<td>6/6</td>
</tr>
<tr>
<td>precision(2gram)</td>
<td>1/5</td>
<td>4/5</td>
</tr>
<tr>
<td>precision(3gram)</td>
<td>0/4</td>
<td>2/4</td>
</tr>
<tr>
<td>precision(4gram)</td>
<td>0/3</td>
<td>1/3</td>
</tr>
<tr>
<td>brevity penalty</td>
<td>6/7</td>
<td>6/7</td>
</tr>
<tr>
<td>BLEU</td>
<td>0%</td>
<td>52%</td>
</tr>
</tbody>
</table>

There is a major disadvantage of the BLEU: translation may contain many synonyms to the words of a real translation, in this case BLEU score will be low because synonyms with different morphological parts lead to frequent mismatches among n-grams of the target and the predicted sentences.

3.5 Conclusions

For the ASR problem characters should be predicted sequentially. We first tried to encode all alphabet characters simply as different numbers (1 to 26 for each letter respectively). However, to preserve the context of the text transcription the n-grams encoding might be useful. The model output can be assessed with the WER and CER scores. For the Translation problem we choose to apply the WordNet, BPE, and Word Embeddings for vector representations, since they give the most prominent results in the related works. The model quality can be assessed with the BLEU score.
4 Speech Recognition Experimental Architecture

The final architecture is inspired by Deep Speech 2 (Baidu Research – Silicon Valley AI Lab, 2015). Our DNN model performs better and consists (fig. 4.1) of several residual CNNs, bidirectional RNNs, and uses CTC loss function with a Beam Search technique and a Language Model. As the augmentation technology we use the SpecAugment, described in the previous chapter.

Figure 4.1 – Own experimental ASR architecture

4.1 Residual CNN for spectrograms

Convolutional Neural Networks (CNN) are the best at pictures processing and abstract features extraction. We could not avoid using them, moreover, we used special Residual CNN layers [11] (Microsoft, 2015), which allow to train significantly deeper models, to optimize them easier, and to achieve a higher accuracy. The idea behind is to implement so called «skip connections» or «residual connections», 

36
when the loss function appears to have a more convex surface (fig. 4.2) (University of Maryland, 2018) [6], than does without those connections.

Figure 4.2 – The loss surfaces of ResNet-56 with/without skip connections. [6]

In the source code we use Python implementation torch.nn.Conv2d for Convolutional neural networks and define own class for Residual CNN.

4.2 Regularization and Activation

As Regularizations, Dropout and BatchNormalization are incorporated. GELU (Gaussian Error Linear Unit) function is a widely-used function (particularly in Google for Transformer-based models), it is retrieved from Tanh, Relu as their combination and is used for the ASR model.

4.3 Bidirectional GRU

In this thesis, to achieve the most accurate speech recognition, we use GRU, since LSTM and GRU are comparable according to various estimates [27], however, GRU outperforms LSTM in terms of time resources.

RNNs allow us to predict only the future context, and if we want to predict the past context as well, Bidirectional RNN should be applied.

Bidirectional Recurrent Neural Network (biRNN) is two unidirectional recurrent networks, one of which processes the input sequence in forward order, and the other in reverse (fig. 4.3). Thus, for each element of the input sequence, two vectors of hidden states are calculated, on the basis of which the network output is
calculated. It solves the problem of unidirectional recurrent networks which output only the future context vector. For training biRNN, the same algorithms are used as for RNN.

**CTC loss function**

The model consists of several DL layers and outputs a probability matrix for characters: each character is believed to be spoken at a certain timestamp. Connectionist Temporal Classification (CTC) is a technique to get around not knowing the correspondence between input and output. It is proven to be particularly well suited for applications such as ASR.

Consider mapping audio input sequences $X = [x_1, x_2, ..., x_T]$ to corresponding text output sequences $Y = [y_1, y_2, ..., y_U]$, from which the accurate alignment should be received. There are several obstacles usually encountered in supervised algorithms:

- $X$ and $Y$ can vary in length, their ratio can also vary in length;
- There is no accurate correspondence between the elements of $X$ and $Y$.

And CTC algorithm overcomes these obstacles by providing an output of probability distribution over any possible $Y$. With this probability distribution we infer a possible character that is spoken at some timestamp in $X$.

For every ML problem Loss Function must be minimized by Gradient Descent, thus, for CTC loss function the conditional probability $p(Y|X)$ should be computed efficiently and should also be differentiable. The prediction during
inference in calculated in a following way: \( Y^* = \arg\max_Y p(Y|X) \). For the timestamps \( t < T \) and possible alignments \( A \) conditional probability looks as a following expression:

\[
p(Y|X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X).
\]

The algorithm is presented on the fig. 4.4:

```
1. Start with a digital representation of sound (spectrogram);
2. Feed the digital input into sequential ML structure as RNN;
3. Network outputs a distribution of \( p(a_t|X) \) for possible mappings \( h, e, l, o \);
4. Then the marginalization process takes place, resulting in a distribution of outputs.
```

Figure 4.4 – CTC algorithm [38]
In Python CTC loss is implemented in a Pytorch library: `torch.nn.CTCloss`.

4.5 Beam Search Technology

Previously we tried to predict the output in a Greedy way, getting one character per timestamp. Beam search is a way that extends the Greedy search and returns a list of the most likely output sequences.

Instead of greedily choosing the most likely next step when constructing a sequence, Beam search expands all possible subsequent steps and stores k most likely, where k is a user-specified parameter, and controls the number of Beams or parallel searches in the probability sequence.

The local Beam search algorithm keeps track of k states, not just one. It starts with k randomly generated states. At each step, all successors of all k states are generated. If anyone is the target, the algorithm stops. Otherwise, it selects k best successors from the complete list and repeats.

In Python Beam Search for CTC is implemented in a Pytorch library: `torch.nn.ctc-decoder()`.

4.6 Language Model

A Language Model, which represents an alignment of n-grams and their corresponding probabilities to appear in the text, should be incorporated into the scoring computation to improve the results. Addition of a lexicon (vocabulary) constraint restricts the next possible tokens for the hypotheses so that only words from the vocabulary can be generated. We implement an n-gram language model with the custom KenLM library (K. Heafield; 2011)[39], trained on our textual Dataset.

4.7 Conclusions

We explained the use of the Residual connections in CNNs for Spectrograms, Bidirectional RNNs and decided to apply CTCLoss with Beam Search + KenLM method to get the most impressive scores in the resulting model for ASR problem. Although it takes many resources to train the designed model, it is still not as heavy as the pre-trained Whisper, yet our model illustrates good results for the collected dataset.
5 Neural machine translation architecture

For Neural Machine Translation we need to consider state-of-the-art solutions such as Sequence to Sequence (Seq2Seq) with Attention mechanism and Transformers, which are also based on Seq2Seq and embeddings. Final model for NMT (fig. 5.1) represents a Seq2Seq and Attention-based model and is trained on our Swiss German — Standard German aligned dataset of 38 000 sentences (Bern dialect). We constructed the model with the GRU cell, described previously, and got impressive results.

Figure 5.1 – NMT architecture

5.1 Word Embeddings

NMT problem can be resolved with the help of meaning space vectors - word embeddings. These data unit representations then are fed into RNN cells.

We use torch.nn.Embedding class to transform words in a sentence into the word embeddings.

5.2 GRU

In this part, to achieve the fastest machine translation performance on training and evaluation, we use GRU, since GRU outperforms LSTM in terms of time resources.
5.3 Attention Decoder

Attention weights are computed in an AttnDecoder() class in the source code. The calculation procedure was explained in the second chapter.

5.4 Loss function, Regularization and Activation

For this part of the thesis, negative log likelihood loss (torch.nn.NLLLoss) is used. As Regularizations, Dropout and BatchNormalization are incorporated. RELU (Rectified Linear Unit) function is a simple and differentiable function which is often applied in various NLP tasks.

5.5 Misspelling correction

Since the Swiss German is a dialect of the Standard German, and the collected text data demonstrates some similarities between them, Swiss words may be sometimes considered as «misspelled» German words. For example, Swiss sentence «I chomme us Schaffhuuse.» is translated into German as «Ich komme aus Schaffhausen.». According to the explanation of the Levenshtein distance mentioned above, the words in these sentences do not differ much from their corresponding translations: «Ich» can be transformed from «I» by 2 insertions, «komme» — «chomme» by 1 deletion and 1 substitution, «Schaffhausen» — «Schaffhuuse» by 1 insertion and 1 substitution. Another important aspect is, the Swiss German and the aligned Standard German sentences often have the same amount of the words which can be substituted one by one. There are sometimes peculiarities, such as omitted auxiliaries in Swiss German, however, if we assume, that they occur quite rarely, we can redefine the task from «Translation» to «Misspelling correction», that is, imagining Swiss German writing as the «misspelled» German.

This can be resolved with the help of Python module spellchecker.SpellChecker() that is based on the Levenshtein distance and matches the most probable «correct» (German) word to the «misspelled» (Swiss German) one. If we set a reasonable amount of letters that can be substituted in one word to get the correct word, we might get reasonable results and calculate the BLEU score to assess the quality and decide whether the ML translator is needed.
5.6 Conclusions

In this part of the thesis the NMT solution is constructed. The ML model can be seen as two blocks: encoder and decoder which include word embeddings and GRU recurrent networks. In the decoder Attention weights are computed, this gives significantly better results as compared to simple decoders which do not include Attention mechanism and, therefore, lose some context from the sentence they try to translate.
6 Set up and Results of the Experiments

We conducted several ASR experiments, including training and testing own developed architecture with Residual CNNs, Bidirectional RNNs, addition of the KenLM Language Model for the Beam Search at the inference stage, and testing separate models, such as Whisper Large, Whisper Small with a fine-tuning on the Swiss German audio dataset of 40 000 utterances. We do not include results of the fine-tuning Whisper models, since we did not obtain reasonable results: the models work only for the Standard German text tokenizer and the test loss does not decrease. Moreover, the Whisper is very heavy (6 GB) and requires much time to show any performance.

NMT experiments include application of a SpellChecker for Swiss German to translate it into a Standard German and construction of the Seq2seq GRU model with Bahdanau Attention. Training and testing stages are conducted on the 38 000 Swiss German — Standard German sentence pairs.

6.1 Set up for ASR

Our dataset consists of 80 hours of audio data which takes 2 GB RAM. To download and process the audio to tensors and then to train a model, GPU accelerator is needed. We purchased additional 500 computational units in Google Colab Pro environment and used data parallelism with the help of Torch.nn.DataParallel(model) that doubled the speed of the computations. Using GPU accelerator and Data Parallelism, it takes about 10 hours to train one model on 40 000 Swiss German utterances. We also conducted experiments on 10 000 utterances and 30 000 utterances to see, whether we need all the data involved in trainig. We took train data to test data ratio of about 90:10.

To boost the computational speed even more, we uploaded all of the 40 000 utterances, converted to tensors, on the HuggingFace site (private account), as first training epoch took significantly more time because of the downloading and preprocessing.
The result model shows its best performance on the whole corpus, it contains 23 millions trainable parameters and takes about 90 MB of memory.

6.1.1 **Source code**

The source code consists of 5 separate parts (see Appendix):

1. Substitution of numbers in text;
2. Main jupyter notebooks with both data preparation and model training for transcription in the Swiss German and in the Standard German;
3. Building and integration of KenLM Language Model for the Swiss German;
4. Testing the sample audio data code;
5. Integration with TelegramBot, accomplished by Viktoryia Kananchuk.

6.1.2 **Data format**

The dataset is initially allocated in folders with both .wav or .flac audio files and an aligned clean transcriptions in a .csv table. The alphabet in a text only includes English letters without German specific «ü», «ä», «ö», and «ß», which were replaced with «ue», «ae», «oe», and «ss» respectively. To see the online training and evaluation results we registered an account on comet-ml where different graphs with WER, CER, Learning rate, and Train/Test loss are demonstrated.

For the Language Model KenLM training we used our text data in Swiss German (40 000 sentences). We extracted a vocabulary from this dataset as well, that is why model recognizes only specific and relatively small vocabulary. The weight of the file with this model SwissKenLM.bin is around 40 MB.

6.1.3 **Optimizer and Learning rate scheduler**

As an optimizer AdamW [34] (Loshchilov, Hutter; 2019) is used. Adam is one of the best optimizers that helps the model to converge faster, unlike other widely used optimizers such as SGD. Nevertheless, Adam has an issue with the weights decay implementation, which AdamW aims to fix. We also consider One Cycle Learning Rate Scheduler [35] (Smith, Topin; 2018) which fits well with small set of data (like we collected) since it implements a trick with a low learning rate at the beginning, that increases to the top and linearly decreases to the starting point at the end.
6.2 ASR Experiments

Experiments include building several models with different number of epochs, Dropout, batch size, number of CNN and RNN layers and size of the dataset. Train and test data ratio is around 90:10.

6.2.1 Results of the own built model on the whole dataset

The results with different Word Error Rates (WER) and Character Error Rates (CER) are illustrated below.

Table 6.2 – CER and WER for different models on 40 000 utterances before KenLM

<table>
<thead>
<tr>
<th>Model name</th>
<th>Epochs</th>
<th>Dropout</th>
<th>Batch</th>
<th>CNNs</th>
<th>RNNs</th>
<th>CER,%</th>
<th>WER,%</th>
</tr>
</thead>
<tbody>
<tr>
<td>deer</td>
<td>20</td>
<td>0.1</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td>33.0</td>
<td>76.2</td>
</tr>
<tr>
<td>chameleon</td>
<td>20</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>28.1</td>
<td>67.1</td>
</tr>
<tr>
<td>salmon</td>
<td>30</td>
<td>0.3</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td>26.8</td>
<td>65.4</td>
</tr>
<tr>
<td>riser</td>
<td>20</td>
<td>0.2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>27.1</td>
<td>65.4</td>
</tr>
<tr>
<td>platypus</td>
<td>20</td>
<td>0.2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>29.4</td>
<td>69.1</td>
</tr>
<tr>
<td>wombat</td>
<td>15</td>
<td>0.1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>26.8</td>
<td>65.0</td>
</tr>
<tr>
<td>first</td>
<td>10</td>
<td>0.2</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td>32.7</td>
<td>86.2</td>
</tr>
</tbody>
</table>

Wombat model has WER of 65% and CER of 26.8%, the evolution of these metrics as well as test loss through epochs is presented below (fig.6.1, 6.2, 6.3).
Figure 6.1 – wombat CER

Figure 6.2 – wombat WER

Figure 6.3 – wombat test loss
Output examples for the best model wombat with WER 65% and CER 26%:

Table 6.3 – Examples of the decoded predictions of the best model without KenLM

| decoded 1 | dr klimawande isch nach mire persoenleche uffassig eis vo de ganz grosse problem |
| target 1  | dr klimawandel isch nach mire persoenliche uffassig eis vo de ganz grosse problem |
| decoded 2 | mi sin vo transparenz u vor rechtsbestaendigkeit betrag beatraeg i di motion abzlehne |
| target 2  | im sinn vo transparenz und raechtsbestaendigkeit beatraege i ne d motion abzlehne |
| decoded 3 | es git stationaer ud ambult |
| target 3  | es git stationaer und ambulant |
| decoded 4 | er ghoert zum schwizerische foederalismussund isch e wichtige standortfak |
| target 4  | er ghoert zum schwizerische foederalismus und isch e wichtige standortfaktor |
| decoded 5 | f zwei wird abglehnt wi o e motion eistimmig abglehnt wid |
| target 5  | d ziffer zwei wird abglehnt wie ou e motion istimig abglehnt wid |

Test sample illustrates that our wombat model transcribes sounds very well in terms of characters, however, it does not necessarily know the proper words of Swiss German vocabulary and predicts the characters in a «greedy» way according to the probability distribution of single characters. It is anticipated since no vocabulary and language model was included. For this reason, we trained our own probabilistic Language Model in 4-grams for Swiss German based on the KenLM Language Model, applied BeamSearch decoding with it on top of the wombat model and got the following results.
6.2.2 Results of the own built model on the whole dataset with KenLM

Table 6.4 – CER and WER for the best model after applying KenLM Swiss

<table>
<thead>
<tr>
<th>Model</th>
<th>CER, %</th>
<th>WER, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>wombat</td>
<td>14.0</td>
<td>35.0</td>
</tr>
</tbody>
</table>

Table 6.5 – Examples of the decoded predictions of the best model with and without KenLM Swiss

<table>
<thead>
<tr>
<th>target 1</th>
<th>dr kanton het im bildigsbereich massnahme troffedr mit persone mit schlechte duetschkenntnis d grundkompetenze choei erreiche</th>
</tr>
</thead>
<tbody>
<tr>
<td>wombat 1</td>
<td>d kanto he im bildigsbereich massnahme detrof dambi persone mit schlechteduetsktnisu d grundkompetaeenzer erreichee</td>
</tr>
<tr>
<td>wombat + KenLM 1</td>
<td>dr kanton het im bildigsbereich massnahme troffedr mit persone mit schlechte duetschkenntnis d grundkompetenze erreiche</td>
</tr>
<tr>
<td>target 2</td>
<td>es geit um ds potential fuerstromproduktion us wasserchraft fuer rund sechstausend hushauve</td>
</tr>
<tr>
<td>wombat 2</td>
<td>es geit um ds potential fuerstromproduktion uswasserchraft fuerrund sechstausend hushaltige</td>
</tr>
<tr>
<td>wombat + KenLM 2</td>
<td>es geit um ds potential fuerstromproduktion us wasserchraft fuer rund sechstausend hushauve</td>
</tr>
<tr>
<td>target 3</td>
<td>di si fuer ues sehr wertvoll</td>
</tr>
<tr>
<td>wombat 3</td>
<td>isi fuer ues sehr waert</td>
</tr>
<tr>
<td>wombat + KenLM 3</td>
<td>fuer ues sehr</td>
</tr>
</tbody>
</table>

We come to the conclusion that wombat + KenLM performs better on longer utterances with more context: it recognizes words more correctly than a simple wombat does. However, when it comes to short recordings with less context,
wombat + KenLM deletes some information and predicts only the correct words which have a higher probability to appear in a text. Nevertheless, it is our best model with 30% WER and 14% CER.

We also trained the model with the same parameters as wombat for the ASR of the Swiss German, but transcribed into the Standard German (accordingly, we trained only on the data where the Standard German labels are presented). We applied kenLM language model for the Beam Search Decoder as well, however, this time we used available kenLM .bin file trained on the German Wikipedia.

The results are as follows after (application of the LM): WER = 68%, CER = 35%. Some of the decoded examples can be seen in the table below.

Table 6.6 – Examples of the decoded predictions of the German model with and without KenLM German

<table>
<thead>
<tr>
<th>target 1</th>
<th>es ist ein thema das praesent ist</th>
</tr>
</thead>
<tbody>
<tr>
<td>decoded wombat 1</td>
<td>des ist e themun diplisentist</td>
</tr>
<tr>
<td>decoded wombat + KenLM 1</td>
<td>ist themen eisen</td>
</tr>
<tr>
<td>target 2</td>
<td>zehntausend neue paedagogen jedes jahr schweizweit naturlich</td>
</tr>
<tr>
<td>decoded wombat 2</td>
<td>dieausin neu peagogen jedes jahr schweizwinatuelich</td>
</tr>
<tr>
<td>decoded wombat + KenLM 2</td>
<td>neu go gen des jahr schweizweiter</td>
</tr>
</tbody>
</table>

We conclude that the model trained on the Swiss German transcriptions is much more accurate.

### 6.3 Set up for Machine Translation

This part of the thesis does not require much RAM. It takes about 15 minutes to train the NMT model on 30 epochs in Google Colab environment with GPU T4.

The source code is introduced in a single Jupyter notebook where both parts with misspelling checker and NMT model are introduced.
6.3.1 Data format

The data represents a .txt file with the 38 000 normalized sentence pairs: Swiss German and the corresponding Standard German sentence. Under normalization we mean the cleaning from punctuation (accomplished with the help of regular expressions). Unlike audio dataset, here the umlauts can be processed.

The implementation includes data shortening for the faster training process: only the sentences shorter than 20 words are chosen. Train : test data ratio is approximately 80 : 20.

6.3.2 Optimizer

Application of such optimizer as the Adam gives very promising results for the final model. It is employed here with the help of torch.optim.Adam.

6.4 Machine Translation Experiments

At first, the hypothesis with misspelling correction is checked. Then we proceed with the Neural Machine Translation model.

6.4.1 Misspelling correction results

We previously expressed our assumptions regarding similarities between written Swiss German and Standard German in Chapter 3. We tested random sets of sentences in the collected text dataset (Bern Parliament Part) by applying Python SpellChecker for German language (language=’de’) and set different numbers of the Levenshtein distance (number of letters in the Swiss word that can be changed to get the German word). Here the results of the German SpellChecker are introduced.

Table 6.7 – BLEU scores for SpellChecker on sentences sets of different size

<table>
<thead>
<tr>
<th>Set number</th>
<th>Set size</th>
<th>distance</th>
<th>BLEU score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>4</td>
<td>8.8</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>5</td>
<td>8.5</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>7</td>
<td>5.3</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>7</td>
<td>8.8</td>
</tr>
</tbody>
</table>
The BLEU scores and «translation» examples prove that a simple Levenshtein distance-based algorithm cannot transform Swiss German text into Standard German accurately. Consequently, the NMT model is needed.

### 6.4.2 NMT model results

Let us introduce the results of the NMT model, trained on different amounts of epochs on different dataset sizes (sizes are reduced by choosing only shorter sentences).

Table 6.8 – BLEU scores for NMT models from Swiss to Standard German

<table>
<thead>
<tr>
<th>Model</th>
<th>Max # words in sentence</th>
<th>epochs</th>
<th>BLEU score</th>
</tr>
</thead>
<tbody>
<tr>
<td>first</td>
<td>10</td>
<td>10</td>
<td>53.4</td>
</tr>
<tr>
<td>second</td>
<td>10</td>
<td>20</td>
<td>74.7</td>
</tr>
<tr>
<td>third</td>
<td>10</td>
<td>30</td>
<td>71.1</td>
</tr>
<tr>
<td>fourth</td>
<td>15</td>
<td>10</td>
<td>58.9</td>
</tr>
<tr>
<td>fifth</td>
<td>15</td>
<td>12</td>
<td>72.5</td>
</tr>
<tr>
<td>sixth</td>
<td>15</td>
<td>15</td>
<td>78.8</td>
</tr>
<tr>
<td>seventh</td>
<td>20</td>
<td>10</td>
<td>54.9</td>
</tr>
<tr>
<td>eighth</td>
<td>20</td>
<td>12</td>
<td>72.5</td>
</tr>
</tbody>
</table>

BLEU score greater than 60 can be assigned to the linguistic translations that are close to the human speech. 50 < BLEU < 60 is considered as an excellent result as well. Score of 78 might represent an overfitting (it is possible because of the dataset size). Thus, our best model is chosen to be the **fourth** one with BLEU = 58.9.
Some translated predictions:

<table>
<thead>
<tr>
<th>Swiss sentence 1</th>
<th>es isch jedoch kes strom us fossile produkteion e d chunde glieferet worde</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual translation 1</td>
<td>es wurde jedoch kein strom aus fossilen produktionen an die kunden geliefert</td>
</tr>
<tr>
<td>Predicted translation 1</td>
<td>es ist jedoch kein strom aus fossilen produktionen an die kunden geliefert</td>
</tr>
<tr>
<td>Swiss sentence 2</td>
<td>ihazg fueu mir soette wuerkelech sorg traeg zu dere berufsbildig</td>
</tr>
<tr>
<td>Actual translation 2</td>
<td>ich habe das gefuehl wir sollten wirklich sorge tragen zu dieser berufsbildung</td>
</tr>
<tr>
<td>Predicted translation 2</td>
<td>ich habe das gefuehl wir sollten wirklich sorge tragen zu dieser berufsbildung berufsbildung</td>
</tr>
<tr>
<td>Swiss sentence 3</td>
<td>unschoenisch dsituation o drum gsy will</td>
</tr>
<tr>
<td>Actual translation 3</td>
<td>unschoen war die situation auch deshalb weil</td>
</tr>
<tr>
<td>Predicted translation 3</td>
<td>unschoen ist deshalb war weil sie</td>
</tr>
</tbody>
</table>

We conclude that the NMT model performs much better than the spellchecker, as it shows good predicted translations in comparison to the actual ones. It still can be improved with the bigger dataset, but then a lower BLEU score will be expected. Another option for the model improvement is using Transformer-based model. However, at the moment we do not need to build a much more complex and heavy model because of the relatively small dataset without very long sentences.
Conclusion

In this thesis we built Machine Learning models to perform the Automatic Speech Recognition of the Swiss German speech, transcribing it into the Swiss German written dialect and to perform the Neural Machine Translation from the Swiss German to the Standard German language. We trained the ASR model on 40 000 utterances (80 hours) and achieved the 35% WER and 14% CER on the test set, using an integration with own built Language Model based on kenLM library. There are many existing tools for the ASR problem with impressive performance, however, because of the lack of Swiss German transcribed data, none of the tools can show a good performance in this language.

As for Neural Machine Translation, we achieved the BLEU score of 58.9 with the Seq2Seq and Attention mechanism model. The model was trained on the 38 000 textual sentence pairs of the Bern Parliament dataset. The performance is impressive on this data but to check this properly, a more diverse dataset of several Swiss German dialects is required.

During this research we discovered that we need more data from different regions, more validation on native Speakers; extended resources for model training since Google Colab Pro and available Clusters are still too slow. Future work includes:

- Resources optimization of the training process;
- Collecting a bigger (at least 1000 hours of Swiss Speech of natives of different age and gender) properly labelled dataset;
- Collecting a bigger dataset of sentence pairs Swiss German — Standard German for NMT task;
- Building new language model with extended vocabulary;
- Training more accurate model with the DL layers we used for this thesis;
- Building Swiss German text tokenizer for heavy ASR models as Whisper;
- Fine-tuning of the heavy architectures like Whisper for ASR, and Transformer for NMT;
– Validation of the built tools on more Swiss German native speakers from different regions;
– Integration with services that provide Speech Recognition and Translation in other languages, yet not in Swiss German.

Execution of the listed steps will noticeably refine the current applications.

At last, we believe, that our Speech Recognition and Machine Translation application is scalable and can expand into other countries where there are many dialects that have some informal written language rules amongst them, but different from the language on which the dialect is based, such as Flemish (Dutch dialect, 6 M people), Fries (Dutch dialect, 2 M people), Valenciano (Catalan, 2 M people).
Bibliographic references


2. Speech and Language Processing Chapter 5, Daniel Jurafsky, James H. Martin (2023)

3. Exploring Wav2Vec2 Pre-Training on Swiss German Dialects using Speech Translation and Classification Samuel Stucki, Patrik Randjelovic (2022)

4. Automatic Speech Recognition (ASR) - DeepSpeech Swiss German Aashish Agarwal; University of Duisburg-Essen(2020); Electronic source link: https://github.com/AASHISHAG/deepspeech-swiss-german-results

5. Language representation and modelling for Swiss German ASR Tannon Kew; University of Zurich; (2020)

6. Visualizing the Loss Landscape of Neural Nets Hao Li1, Zheng Xu1, Gavin Taylor2, Christoph Studer3, Tom Goldstein1; 1University of Maryland (2018)

7. Identity Mappings in Deep Residual Networks Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun; Microsoft Research (2016)

8. Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER); Kenneth Leung (2021)

9. Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What’s In-Between Haytham Fayek (2016)


12. Deep Neural Network acoustic models for ASR Abdel-Rahman Mohamed; University of Toronto (2014)


20. Neural Machine Translation of Rare Words with Subword Units 3.2 Rico Sennrich, Barry Haddow, Alexandra Birch; School of Informatics, University of Edinburgh; (2016)


23. **NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE** Dzmitry Bahdanau, KyungHyun Cho Yoshua Bengio (2016)


25. **Robust Speech Recognition via Large-Scale Weak Supervision** Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, Ilya Sutskever; (2022)


28. **Structure of Seq2seq** [Electronic resource] // URL: https://medium.com/analytics-vidhya/encoder-decoder-seq2seq-models-clearly-explained-c34186f8f49b

29. **Structure of LSTM** [Electronic resource] // github URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/


32. **The Unreasonable Effectiveness of Recurrent Neural Networks** Andrey Karpathy [Electronic resource] // github URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/


34. **DECOUPLED WEIGHT DECAY REGULARIZATION** Ilya Loshchilov, Frank Hutter; University of Freiburg (2019)


37. *RNN architecture* Moscow Institute of Physics and Technologies; Machine Learning lectures, R. Neychev (2017)


Appendix

Source code:

Jupyter notebooks description:
1. spell-numbers-in-german.ipynb — Substitution of numbers in text;
   1 function;
2. audio-model-training-master-thesis.ipynb — Main jupyter notebook with
   both data preparation and model training;
   13 functions; 7 classes;
3. kenlm-swiss-german.ipynb — Building and integration of KenLM Language
   Model;
   1 function;
4. beam-search-decoder-ASR-master-thesis.ipynb — Testing the sample audio
   data code;
   14 functions; 7 classes;
5. Integration with TelegramBot, accomplished by Viktoryia Kananchuk.
6. seq2seq-translation-swiss.ipynb — Seq2seq Attn NMT model training and
   inference + spellchecker;
   17 functions; 5 classes;