Master's programme in Computer, Communication and Information Sciences

A Mobile App For Practicing Finnish Pronunciation Using Wav2vec 2.0

Nhan Chi Phan

Master's Thesis
2023

Author Nhan Chi Phan

Title A Mobile App For Practicing Finnish Pronunciation Using Wav2vec 2.0

Degree programme Computer, Communication and Information Sciences

Major Machine Learning, Data Science and Artificial Intelligence

Supervisor Prof. Mikko Kurimo

Advisors Dr Tamás Grósz, MSc Ekaterina Voskoboinik

Date 29 March 2023 **Number of pages** 65+3 **Language** English

Abstract

As Finland attracts more foreign talents, there are demands for self-learning tools to help second language (L2) speakers learn Finnish with proper feedback. However, there are few resources in L2 data in Finnish, especially focusing on the beginner level for adults. Moreover, since L2 adults are mainly busy studying or working in Finland, the application must allow users to practice anytime, anywhere.

This thesis aims to address the above issues by developing a mobile app for beginner Finnish L2 learners to practice their pronunciation. The app would evaluate the users' speech samples, give feedback on their pronunciation, and then provide them with instructions in the form of text, photos, audio, and videos to help them improve their pronunciation.

Due to the limited resources available, this work explores the wav2vec 2.0 model's capability for the application. We trained our models with the native Finnish speakers' corpus and used them to provide pronunciation feedback on L2 samples without any L2 training data. The results show that the models can detect mispronunciation on phoneme level about 60% of the time (Recall rate) compared to a native Finnish listener. By adding regularizations, selecting training datasets, and using a smaller model size, we achieved a comparable Recall rate of approximately 63% with a slightly lower Precision of around 29%. Compared to the state-of-the-art model in Finnish Automatic Speech Recognition, the trade-off resulted in a significantly faster response time.

Keywords Mispronunciation detection and diagnosis, mobile app, low-resource, wav2vec 2.0 , end-to-end

Preface

As the author of this thesis and a Finnish learner, I would like to thank all the contributors to open-sourced materials, particularly those for the Finnish language. Accordingly, this work is also licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International license.

I would like to express my gratitude to all the Finnish teachers and members of the Kielibuusti project. Your contribution and advice during developing the Finnish pronunciation mobile app are highly appreciated.

I also want to thank my supervisor, Professor Mikko Kurimo, and my advisors, MSc Ekaterina Voskonoinik and Dr Tamás Grósz, for the opportunity to work in their research group and for their invaluable guidance.

Finally, I want to thank my 2-year-old daughter, nicknamed Chà Bông. She was a great source of motivation for my work, and she also contributed a significant amount of her swimming time to this project. Additionally, I would like to to thank my family for their support during my busiest time and for helping me maintain my sanity.

Otaniemi, 29 March 2023

Nhan Chi Phan

Contents

Symbols and abbreviations

Symbols

sim	cosine similarity
$\mathbb{E}$	expectation
$\exp$	exponential function
$\mathcal{L}$	loss function
$\log$	logarithm
$\prod$	product
σ	sigmoid function
$\sum$	summation
$\top$	transpose of a matrix
W	weight matrix in a neural network

Abbreviations

ASR	automatic speech recognition
CER	character error rate
CAPT	computer-assisted pronunciation training
CERF	Common European Framework of Reference for Language
CTC	connectionist temporal classification
CD	correct diagnosis
DAR	diagnosis accuracy rate
DL	deep learning
DE	diagnostic errors
E2E	end-to-end
FN	false negative
FP	false positive
FFN	feed-forward network
GELU	Gaussian error linear units
GOP	Goodness of Pronunciation
IPA	International Phonetic Alphabet
KL	Kullback-Leiber
L2	second language
MD	mispronunciation detection
MDD	mispronunciation detection and diagnosis
NA	not available
NLP	natural language processing
ReLU	rectified linear activation unit
SD	standard deviation
TN	true negative
TP	true positive
UI	user interface
UX	user experience
VAD	voice activity detection
WER	word error rate

1 Introduction

Learning to speak a second language (L2) often involves practicing with native speakers through listening, repeating, and corrections [1]. For beginner L2 adult learners in Finland, practicing speaking Finnish outside the classroom can be challenging, especially since Finnish culture does not encourage small talk. They may also have limited free time for practicing Finnish due to work or study. Even during Finnish classes, teachers might not have resources to correct beginners' pronunciation mistakes, as it requires one-on-one practice. Consequently, there is a demand for a Finnish pronunciation mobile app that L2 learners can use to practice and receive feedback anytime, anywhere. This type of application is known as computer-assisted pronunciation training (CAPT), and the mobile app developed for this thesis is CaptainA. The app was inspired by earlier work by Rouhe et al. [2] under the same name.

While there are extensive global resources for learning foreign languages, most focus on major languages like English or Mandarin [3, 4, 5, 6]. They provide pronunciation practice, vocabulary, and other language learning opportunities. However, there are fewer similar applications for a smaller language, such as Finnish. Researchers have recently collaborated to develop new applications for L2 Finnish learners [7, 8, 9]. However, these applications mostly provide feedback in the form of a holistic score or the Common European Framework of Reference for Language (CERF) level. The holistic score might be more useful for intermediate or advanced L2 learners. For beginner L2 learners, who start with pronunciation, more detailed feedback is needed [10]. Specifically, from the pronunciation practice perspective, the CAPT system should resemble the feedback of a language teacher: identifying the error at the phoneme level and providing instruction to improve pronunciation [11]. In other words, a good CAPT should have the capabilities of mispronunciation detection and diagnosis (MDD). To our knowledge, no previous research has developed a similar app for Finnish L2 learners.

Furthermore, most applications for learning the Finnish language are developed under low-resource settings, as there is a lack of L2 Finnish corpora. Currently, the main dataset for L2 Finnish learners is Digitala [9]. The Digitala corpus, however, only has holistic scores, and the transcript does not have detailed annotations at the phoneme level. The lack of phoneme level annotation dataset makes developing a CAPT application with analytical feedback challenging.

Finally, this thesis also focuses on the practicality of the CAPT system. To provide a positive user experience (UX), the delay between the user's input and the CAPT feedback should be minimized. For example, in mobile web search, UX significantly declines when the latency surpasses the threshold of 7 - 10 seconds [12]. Although there is no reference for a mobile CAPT system, we aim to achieve a maximum response time of less than 10 seconds. Consequently, an essential part of this work is experimenting with smaller model sizes while striving for a reasonable level of performance.

Traditionally, MDD systems were built with acoustic-based Hidden Markov models and often required large-scale training datasets. Recent research in MDD has shown that end-to-end systems with Connectionist Temporal Classification (CTC) and Transformer-based architecture have promising results, even with low-resource settings [13, 14]. Our application uses a similar architecture but with a major challenge. While the research mentioned above is in popular languages (English, Mandarin) with detailed annotation from experts, we work on Finnish without the same level of detail. The main goals for this thesis are:

- Develop a read-aloud pronunciation practice application for beginner L2 Finnish speakers with detailed feedback at the phoneme level by using the forced alignment algorithm [15] and by leveraging the wav2Vec 2.0 architecture [16] for the low-resource setting. Also, address the most challenging problem of the work, developing an MDD application without a dataset annotated at the phoneme level.

- Design constructive feedback to motivate and attract L2 learners. Apply entropy regularization [17] during the training of our models to increase the feedback range. Add multimedia to the application to help learners with their self-study, based on the survey result of the CAPT application [11].

- Achieve fast response time for the app when developing our models and application. Ensure a reasonable accuracy level of the model by comparing it with a baseline.

The structure of this thesis is organized as follows. This section introduced the thesis's background, objectives, and challenges. Section 2 describes the foundation theories for machine learning and E2E approaches we chose for our work. The next section, Section 3, discusses the related research in MDD and how to evaluate its performance, as well as explains the theory of the main improvement made in the thesis. In Section 4, we explain our reasons for choosing pre-training, training, fine-tuning, and test corpora, including our findings about the advantages and disadvantages of each corpus. The specifications of our models and experiment are detailed in Section 5. The results of our experiments are summarized and investigated further in Section 6. For illustrations of the CaptainA app and its functionalities, please refer to 7. And finally, Section 8 summarizes our work, followed by considerations for potential work and discussions about its impact.

Throughout this thesis, we use alphabet letters instead of symbols from International Phonetic Alphabet (IPA) to represent Finnish letters and their respective phonetic pronunciation. Except for cases of “nk” with IPA [ŋk] and “ng” with IPA [ŋː], Finnish phonemes can be interchangeably represented with either alphabet letters or IPA symbols. Correspondingly, the long phoneme would be replaced with an extra duplicate letter instead of the symbol [ː].

2 Background

While automatic speech recognition (ASR) uses many different techniques and architectures, our project focuses on the end-to-end (E2E) system, which uses a model to handle the entire task instead of relying on multiple components or intermediate steps. Therefore, this section discusses the related architectures used in this paper and the common metrics used for MDD.

2.1 Machine learning and deep learning

Machine learning is a field within artificial intelligence that focuses on building predictive models by learning from available data. While artificial intelligence covers a wide range of techniques that allow machines to mimic human behavior, machine learning specifically focuses on the machine's ability to learn independently without explicit instructions. Machine learning is related to statistics, which the main purpose is to analyze, interpret and present data. It is also related to data mining, in which scientists try to extract meaningful observations from a large amount of data. One of the subfields of machine learning is deep learning (DL).

DL is a branch of machine learning that was initially inspired by the structure of the human brain, called artificial neural networks. These networks can automatically extract important features during training without requiring thoroughly designed algorithms, thus allowing machines to output optimal results with less human intervention. On the other hand, to achieve superior results, artificial neural networks require a significant amount of data and longer training time than other machine learning methods.

There are multiple algorithms in machine learning and artificial neural networks. In this paper, we focus on supervised learning, unsupervised learning, and the approach we used for our models: self-supervised learning.

2.1.1 Supervised learning

Supervised learning is a machine learning algorithm that allows machines to learn from labeled data. The labeled data has two components: the features of a data point and the corresponding correct label. The machine learning model then uses this information to learn the patterns in the data and establish a mapping between the features and labels. However, as high-quality labeled data requires manual work from experts in the field, supervised learning performance is often limited by the amount of data available for training.

2.1.2 Unsupervised learning

Unlike supervised learning, unsupervised learning utilizes unlabeled data for training. Without human intervention, unsupervised learning algorithms can discern hidden patterns in data and are primarily used for clustering tasks, association tasks, or dimension reduction. As unsupervised learning relies on unlabeled data, its models often require larger datasets for training and can yield lower accuracy than other algorithms. Although unlabeled data can be available in large quantities, training with a larger dataset requires more computing power.

2.1.3 Self-supervised learning

Self-supervised learning is a machine learning algorithm that bridges the gap between supervised learning and unsupervised learning. Similar to unsupervised learning, the algorithm is designed to learn from unlabeled data without human intervention. Instead, self-supervised learning generates the labeled data automatically from the data itself, hence the name self-supervised. Self-supervised learning has an advantage over supervised learning, as it can leverage unlabeled data for training. Moreover, self-supervised learning can perform classification tasks with limited labeled data.

2.2 End-to-end ASR

ASR is a subfield of computer science and natural language processing (NLP) focusing on developing machine learning models that can transcribe audio speech into written text. ASR systems can be classified into two categories: conventional ASR systems, which often consist of multiple components such as the acoustic model, language model, and decoder, and end-to-end (E2E) ASR system, which consist of a single model.

The E2E approach is a newer development in ASR, gaining popularity with the advancement of DL technology. The main difference between the traditional approach and E2E is that the latter directly maps appropriate speech representations to the final output. E2E ASR replaces multiple modules between the audio and the target text with artificial neural networks, simplifying the ASR system's pipeline.

2.2.1 Transformer

Initially introduced in 2017 as a new architecture for NLP, the Transformer [18] has since become popular in other fields, such as computer vision [19], due to its advantages. Transformer is a type of DL architecture that relies entirely on its self-attention mechanism for computing data representations. This self-attention mechanism allows the Transformer to avoid sequential algorithms or convolution, resulting in faster computation speed through parallel computing. In NLP, Transformer effectively extracts contextual information and, as a result, has made its place in many state-of-the-art models [16, 20, 21, 22, 23].

The Transformer architecture consists of two main blocks: the encoder and the decoder. Both the encoder and the decoder are formed by six fully connected layers, with identical structures within each block. The output of the encoder block is fed into the decoder block via the multi-head attention module. The basic structure can be seen in Figure 1.

Each layer in the encoder block starts with a multi-head attention module followed by a position-wise fully connected feed-forward network (FFN) module. The FFN module is formed with two linear layers and the rectified linear activation unit (ReLU)

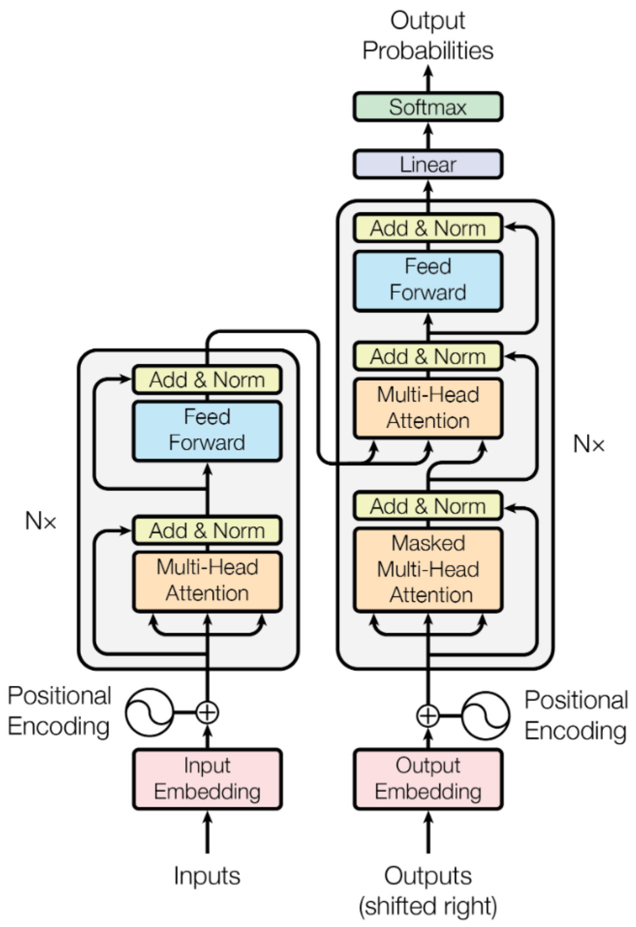


Figure 1: The basic architecture of a Transformer model [18].

activation function in the middle (Equation 1). Both the multi-head attention and FFN module are employed with a residual connection $x + f(x)$ where $f(x)$ is the module function; the result is then normalized.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{1}$$

The decoder has a similar structure with the addition of another multi-head attention module that takes the output of the encoder block. Each decoder module also includes a residual connection, followed by layer normalization.

The multi-head attention modules (left part of Figure 2) inside the encoder and the decoder allow the Transformer to, as the name implied, simultaneously attend to information from different perspectives of the input at various positions. The multi-head attention inputs are three parameters called value (V) with dimension d_v, key (K), and query (Q) with the same dimension d_k. These parameters are generated from the input sequence by the Embedding and Positional Encoding modules, as shown in Figure 1.

The original paper [18] used $h = 8$ parallel attention layers, with each attention layer being counted as one head, to form multi-head attention. All heads are concatenated

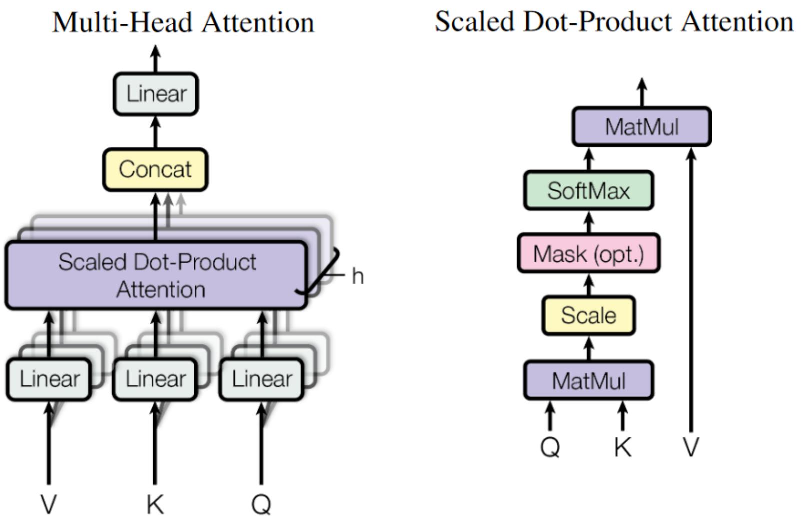


Figure 2: Multi-head attention and the scaled dot-product attention [18].

into the output with as follows:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2}$$

where matrix $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

For fast and efficient computing of the attention (head), the Transformer used dot-product attention and scaled the result by a factor of $\frac{1}{\sqrt{d_k}}$. The algorithm is therefore called scaled dot-product attention and can be seen in the right part of Figure 2:

$$Attention(Q, K, V) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{3}$$

And therefore, with parameters matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, each $head_i$ in Equation 2 is calculate as:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{4}$$

Finally, as mentioned previously, the Transformer model reduces computing complexity by not using recurrent layers or convolutional layers. However, this means the model cannot understand the order of the input sequence. To solve this problem, the Transformer model includes a Positional Encoding module together with the Input Embedding at the beginning of the encoder and decoder blocks. The Positional Encoding injects information about the position of the tokens into the model. In their paper, Vaswani et al. [18] use sine and cosine functions as follows:

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= sin(pos/10000^{2i/d_{model}}) \end{aligned} \tag{5}$$

where pos is the position and i is the dimension. Each dimension of the Positional Encoding corresponds to a sin wave, potentially allowing the Transformer model to extend to a longer sequence than those seen in the training data.

The Transformer is becoming increasingly popular in MDD applications and has demonstrated improvements over other E2E models and conventional methods [13, 14, 24]. Therefore, we aim to explore its effectiveness in the Finnish MDD task. It should be noted that, in contrast to other research in English and Mandarin that commonly utilizes datasets with detailed annotations at the phoneme level [13, 14, 24, 25], our Finnish learner's dataset is more limited.

2.2.2 CTC

Connectionist temporal classification (CTC) was first introduced as a novel method for sequence labeling with E2E models [26]. The CTC network is used to select the label with the highest probability from a given sequence.

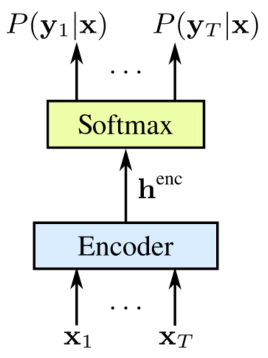

$P(\mathbf{y}_1|\mathbf{x})$
$P(\mathbf{y}_T|\mathbf{x})$
Softmax
$\mathbf{h}^{\text{enc}}$
Encoder
$\mathbf{x}_1$
$\mathbf{x}_T$

Figure 3: The basic architecture of a CTC network.

With $\mathbf{x}$ being the utterance input and $\mathbf{y}$ being the target label, the conditional probability of output sequence given input sequence $\mathbf{x}$ is:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\hat{\mathbf{y}} \in \mathcal{B}(\mathbf{y},\mathbf{x})} \prod_{t=1}^{T} P(\hat{y}_t|\mathbf{x}) \tag{6}$$

where $\mathcal{B}(\mathbf{y}, \mathbf{x})$ is the set of all possible labels plus the blank token, with length $|\mathbf{x}| = T$. $P(\hat{y}_t|\mathbf{x})$ is the conditional probability of the labels at each time step, estimated by the encoder as depicted in Figure 3.

The CTC is simpler than other E2E approaches and has been shown to outperform traditional ASR models when combined with other E2E models [27]. Additionally, when used together with a language model, CTC has demonstrated comparable performance to other E2E approaches [28]. For an illustration of how the CTC decoder work, see Figure 4. In MDD, the CTC has been used in conjunction with the Transformer attention module to deliver outstanding results [13, 15, 24].

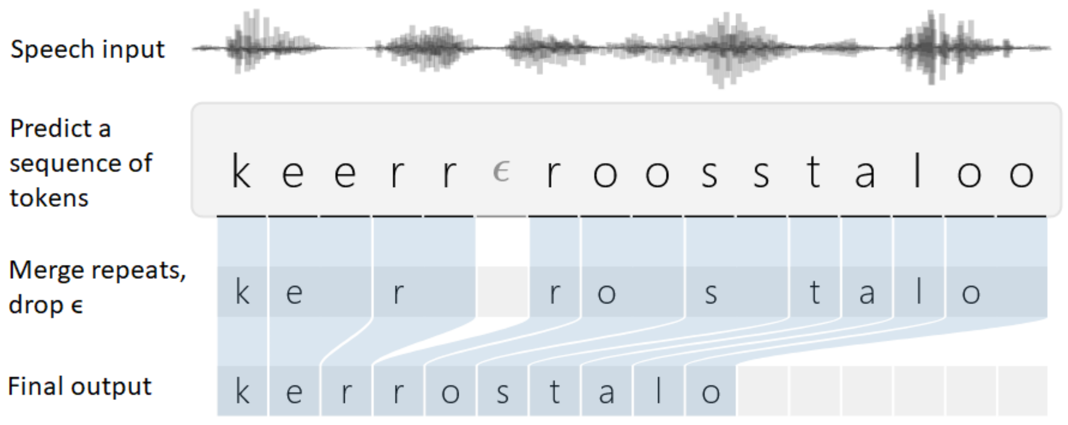


Figure 4: Illustration of CTC decoding [27]. The decoded word was "kerrostalo". The blank token is ϵ and was critical in decoding two letters "r".

2.2.3 Wav2vec

Wav2vec is a neural network architecture that leverages the amount of unlabeled data available in speech recognition. Wav2vec aims to address the issue faced by ASR models when labeled data is scarce. Instead of investing resources in acquiring more data, Schneider et al. [29] proposed a different approach by pre-training the model with a substantial amount of labeled or unlabeled generic datasets. The results show that pre-trained models perform better on downstream tasks than models without pre-training on unlabeled data.

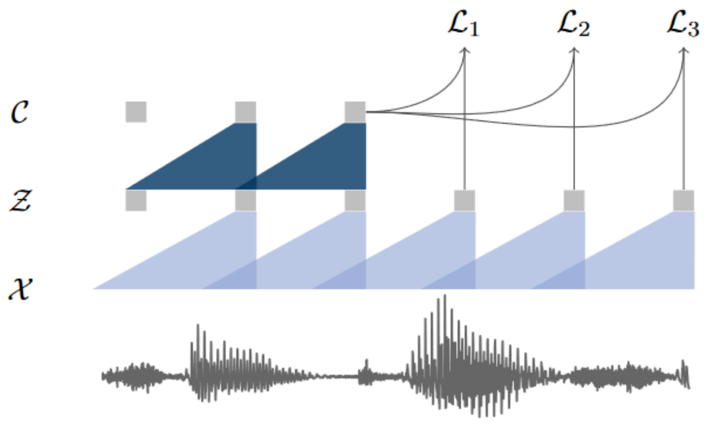


Figure 5: Illustration of wav2vec pre-training framework [29].

The goal of pre-training is to learn general representations from raw, unlabeled audio $\mathcal{X}$. As illustrated in Figure 5, the pre-trained wav2vec model is used as input for the ASR model through the encoder network $\mathcal{Z}$ $f : \mathcal{X} \to \mathcal{Z}$, and the context network $g : \mathcal{Z} \to C$. The wav2vec model is optimized to predict future samples from a given audio signal. By minimizing contrastive loss $\mathcal{L}$ for each step $k = 1, \ldots, K$, the model is trained to identify the true sample $\mathbf{z}_{i+k}$ from the distractors $\tilde{\mathbf{z}}$. With sigmoid $\sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i))$ as the probability of $\mathbf{z}_{i+k}$ being the true sample, we have:

$$\mathcal{L}_k = -\sum_{i=1}^{T-k}\left(\log\sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i)) + \lambda \underset{\tilde{\mathbf{z}}\sim p_n}{\mathbb{E}}\left[\log\sigma(-\tilde{\mathbf{z}}^\top h_k(\mathbf{c}_i))\right]\right) \tag{7}$$

This distinguishes the wav2vec model from the conventional ASR models that rely on acoustic features such as log-mel filterbanks. As the time of its introduction, the wav2vec model demonstrated superior performance compared to the best character-based model, using notably less labeled data [29]. Its outstanding performance highlights the importance of the pre-training approach in ASR, particularly in low-resource settings.

2.2.4 Wav2vec 2.0

After self-supervised learning appeared as a successful solution for NLP in low-resource languages, wav2vec 2.0 was introduced as a self-supervised framework that leverages the unlabeled raw audio data for pre-training [16]. As a successor to wav2vec, wav2vec 2.0 share a similar concept but with the addition of the Transformer network for better context representations and utilizes CTC for downstream tasks. The wav2vec 2.0 are fully E2E without requiring an external module or additional step. Since its introduction, wav2vec 2.0 have demonstrated superior performance in various NLP tasks [13, 14, 30, 31], especially for L2 speakers in low-resource settings [9, 24, 32].

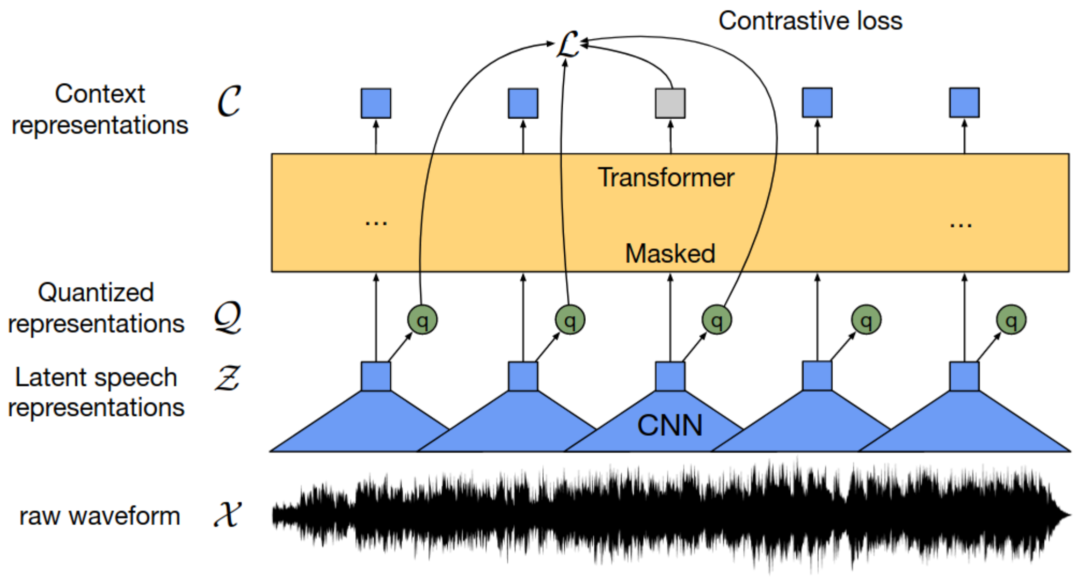


Figure 6: Illustration of wav2vec 2.0 pre-training framework.

Similar to its predecessor, wav2vec 2.0 leverages the abundance of unlabeled data available with its self-supervised pre-training capability. The structure of the wav2vec 2.0 model can be seen in Figure 5. During pre-training, the model learns speech representation directly from raw audio waveform $\mathcal{X}$ with a feature encoder. The feature encoder is formed from multiple temporal convolution blocks, followed by a normalization layer and a Gaussian error linear unit (GELU) activation function [33]. It takes standardized (zero mean and unit variance) raw audio $\mathcal{X}$ as input. The encoder outputs are T time-steps latent speech representations from $\mathbf{z}_1$ to $\mathbf{z}_t$ as $f : \mathcal{X} \rightarrow \mathcal{Z}$.

The speech representation outputs are then used to form corresponding context representations $\mathbf{c}_1, ..., \mathbf{c}_t$ using the Transformer architecture (2.2.1) $g : \mathcal{Z} \rightarrow C$. To encode relative positional information, wav2vec 2.0 replace the fixed positional embeddings with a convolutional layer resembling the relative positional embedding described in [34]. GELU activation function and layer normalization are also applied to the convolution output.

Furthermore, $\mathcal{Z}$ is also used for self-supervised training $\mathcal{Z} \rightarrow Q$ with quantization [35]. The quantization output $\mathbf{q} \in \mathbb{R}^f$ is chosen from the transformation $\mathbb{R}^d \rightarrow \mathbb{R}^f$ of vectors $e_1, ..., e_G$. With G representing the number of codebooks, and V representing the number of entries $e \in \mathbb{R}^{V \times d/G}$. To determine the probabilities for choosing entry v in codebook g, $p_{g,v}$, Gumbel softmax [36] is used with the following formula:

$$p_{g,v} = \frac{\exp(l_{g,v} + n_v)/\tau}{\sum_{k=1}^{V} \exp(l_{g,k} + n_k)/\tau} \tag{8}$$

where τ is the softmax temperature, n is independent and identically sampled from $n = -\log(-\log(u))$ with u is uniform distribution (0,1).

During pre-training, part of the latent speech representation $\mathcal{Z}$ is masked before being fed into the Transformer network. The masked time step t, corresponding to the context network $\mathbf{c}_t$, is used for learning the representation of speech audio by solving for a contrastive loss $\mathcal{L}_m$. The contrastive loss objective is to identify the true quantized latent speech representation $\mathbf{q}_t$ from a set of 101 $\tilde{\mathbf{q}} \in \mathbf{Q}_t$:

$$\mathcal{L}_m = -\log \frac{\exp(sim(\mathbf{c}_t, \mathbf{q}_t)/\kappa)}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}})/\kappa)} \tag{9}$$

where Baevski et al. [16] used $\kappa = 0.1$, and cosine similarity is calculated as $sim(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$

In the original setup, the configuration was $G = 2$ codebooks with $V = 320$ entries for a total of $V^G = 102400$ codewords. Considering the quantity, it's crucial to ensure each entry in each codebook has the same probability of being chosen for contrastive loss optimization. In the wav2vec 2.0 introduction paper, diversity loss $\mathcal{L}_d$ was implemented as follows:

$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v} \tag{10}$$

where $\bar{p}_g$ is the average softmax distribution of the code codebook entries for each codebook, and $H(\bar{p}_g)$ is the entropy. The objective of diversity loss is to maximize the entropy, which ensures $\bar{p}_g$ is uniformly distributed.

Therefore, the final loss function for the pre-training task is the total of contrastive loss $\mathcal{L}_m$ and diversity loss $\mathcal{L}_d$:

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d \tag{11}$$

where α is a hyperparameter that was set to 0.1 in the original setup.

After pre-training, the wav2vec 2.0 model needs to be fine-tuned for downstream tasks with labeled data. During fine-tuning, the weights of the feature encoder $\mathcal{Z}$ are frozen to prevent losing the speech representations learned during pre-training. The knowledge gained during pre-training is valuable as the process typically uses a considerable amount of data and time. Since the downstream tasks typically have limited labeled data, freezing the feature encoder can prevent overfitting the model to a small dataset. Instead of updating the feature encoder, for the ASR task, wav2vec 2.0 is fine-tuned with an additional linear projection on top of the network. The linear layer is used for the classification task, with the number of classes representing the vocabulary size. The model is optimized by minimizing CTC (mentioned previously on 2.2.2) loss:

$$\mathcal{L}_{ctc} = -\log(P(\mathbf{y}|\mathbf{x})) \tag{12}$$

In the paper, because the wav2vec 2.0 was fine-tuned with few labeled data, in addition to freezing the feature encoder, a masking strategy was applied, similar to SpecAugment [37].

At the beginning of this subsection, we mentioned that we selected the wav2vec 2.0 architecture primarily due to its outstanding performance for MDD in low-resource settings. Given the amount of data for Finnish L2, which is even more limited compared to other MDD research in English L2, we believed that the pre-trained wav2vec 2.0 models would be particularly beneficial for our project. In addition, there are multiple open-source wav2vec 2.0 models available in Hugging Face [38], with various sizes and pre-training, fine-tuning corpora. Considering the practicality of our thesis, we could quickly assess the latency response from each model size to select the optimal size for our CaptainA app. Since this is a pilot work with no baseline for the Finnish MDD task, we utilized a state-of-the-art Finnish ASR model from Hugging Face as our baseline. And we also used the hyperparameter configurations and dataset selections from those models as sources of inspiration to develop our own models.

3 Mispronunciation detection and diagnosis

3.1 Related work

Most of the MDD research has been centered around the English language, as evidenced by several studies [13, 14, 24, 39, 40, 41]. Furthermore, recent efforts have been made toward developing MDD models for Mandarin [42, 43, 25]. With the goal of helping L2 speakers learn the language, their MDD research often consists of two parts: the first focuses on the detection of mispronunciations made by the L2 speaker (the MD task), while the second analyzes the mispronunciation errors (the diagnosis task).

The lack of MDD research on other languages may come from the scarcity of relevant public datasets. Even for popular languages like English and Mandarin, the resources for MDD are limited. Most MDD research uses at least one detailed L2 dataset annotated to the phoneme level by multiple experts in the field. The annotations are tailored for the MDD, with each phoneme annotated manually, and usually include mispronunciation tags and other diagnoses.

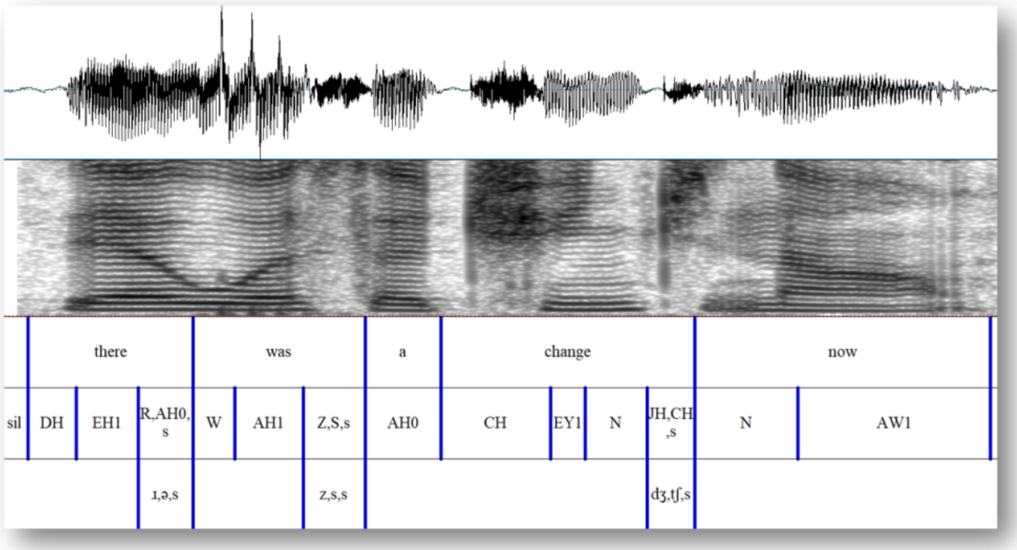


Figure 7: An example of the detailed annotations in the L2_ARCTIC corpus [44], includes speech waveform, spectrogram, words, phonemes, error tags, and comments from the annotator.

For example, the popular public L2-ARCTIC corpus [44] contains English samples made by L2 speakers from multiple backgrounds. Aside from the orthographic transcription at word and phoneme levels, the corpus also contains a subset that was manually annotated by experts. The annotations include the corrected word and phoneme boundaries; substitution, deletion, and insertion phoneme error tags (Figure 7). Similarly, in Mandarin, when there were no public MDD dataset available, researchers also built their own MDD corpus with phoneme-level annotations by language experts [42, 25].

For the Finnish language, to the best of our knowledge, there has been no prior research that has developed a comparable MDD application for L2 learners. The most recent work from Al-Ghezi et al. [9] focuses on automatic speech assessment. However, their pronunciation feedback is provided as a holistic score for the entire task without MDD on the phoneme level. There is also an older application developed by Rouhe et al. [2], but it was primarily developed to collect data and only provides pronunciation statistics after the training session. It is worth noting that both of these applications are only available via web interfaces for testing or demonstration purposes, and have not yet been made available to the general public.

In terms of mobile applications, as far as we know, there is no such app for the Finnish language. We found a mobile app, Duolingo [45], which offers Finnish lessons, but it does not provide detailed Finnish pronunciation practice. In English, we found ELSA Speak [3] is the most comparable to our project's target. It offers English pronunciation practice for L2 learners and is available on mobile devices. ELSA Speak can provide phonetic ratings and detailed diagnoses for pronunciation made by users, with later versions having more advanced features to help users practice speaking English. BoldVoice [46] is another MDD app for English and is also similar to ELSA Speak, with both charging a subscription fee to users.

In English, we have also noticed a gap between academic research and practical application in the field of MDD. Most of the academic research on MDD did not produce a publicly available MDD app. On the other hand, the research team behind popular pronunciation practice apps, such as ESLA Speak or Duolingo, did not publish the performance of their MDD systems.

Therefore, our research in Finnish MDD for L2 speakers is novel. Furthermore, we not only conducted academic research but also developed a practical mobile application utilizing the optimal model we discovered, thereby increasing the project's practicality and usefulness.

3.2 Goodness of Pronunciation

While we use an E2E approach for MDD, our approach closely resembles the traditional Goodness of Pronunciation (GOP) method [47]. The GOP algorithm can be explained with the block diagram in Figure 8. First, the speaker's audio signal is processed by a feature extractor. For a predetermined text, a dictionary provides the model with the target phonemes to measure GOP. Each target phoneme is aligned to a specific acoustic segment by applying the forced alignment algorithm on the feature sequence extracted from the audio file. We can determine the posterior probability of the phoneme being pronounced, using a conventional ASR model, such as the hidden Markov model. The GOP score of a target phoneme is the log of the probability we just found, normalized by its duration. To be accepted as correct pronunciation, the GOP score must be higher than a subjective threshold, called ”strictness” by Witt and Young [47].

In Finnish, each phoneme is represented by exactly one grapheme, except “nk” [ŋk] and “ng” [ŋː]. We can use graphemes directly to represent phonemes, thus eliminating the need for a pronunciation dictionary. As explained above, the wav2vec 2.0 model can independently extract speech representations without an additional

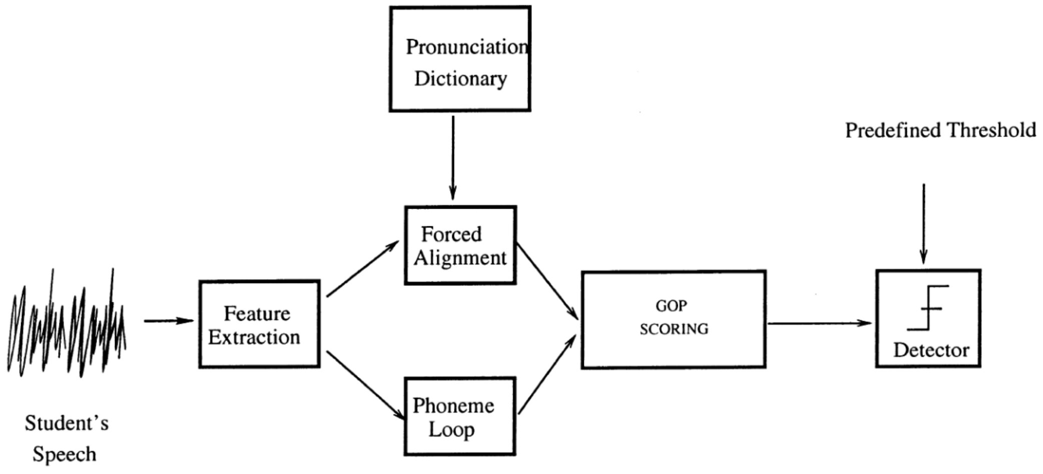


Figure 8: Block diagram of the GOP pronunciation scoring system.

feature extractor. Furthermore, the final output of the fine-tuned wav2vec 2.0 model is an array containing the log probability of all phonemes in the model's vocabulary. This multi-dimensional covers all frames of the audio input. By applying forced alignment to the output of the wav2vec 2.0 model, we can directly obtain all target phonemes scores.

It should be noted that traditional GOP uses phoneme score and a fixed threshold to determine mispronunciation, while our CaptainA uses the E2E model to extract phoneme scores and only provide them as feedback to the speaker. We could not scientifically validate our threshold due to the limitations of our dataset (read Section 4 for more detail). Additionally, the entropy regularization used in Section 3.4 altered the phoneme scoring scale, making it impossible to determine a universally accepted threshold for all models.

For comparison purposes, we use the CTC decoder result, the ASR prediction transcript, and the Levenshtein algorithm [48] to determine MDD. Since CaptainA's phoneme score does not affect MDD results, we use the maximum (instead of normalized) probability over the phoneme's audio frames. The use of maximum probability offers users more lenient feedback, primarily aimed at encouraging them to practice more. Our decision to adopt lenient scoring is also based on our findings, which will be discussed further in section 6, and other studies on the pronunciation feedback [10, 11, 49, 50].

3.3 Performance metrics

There are two different metrics to measure the performance of MDD. The first focuses on the accuracy of mispronunciation detection (MD), and the second measures the accuracy of the diagnosis. In both cases, the predicted phone sequence (made by the MDD model) is aligned with the ground truth (annotated by linguists) using the Levenshtein algorithm [48] or the Needleman-Wunsch algorithm [51]. Both algorithms generally yield the same result, but we chose Levenshtein for our project as it has a shorter execution time [52]. The quantity and position of correct pronunciation and errors (either insertion, substitution, or deletion error) are used to evaluate the performance of MDD.

Table 1: Confusion matrix for MD.

		Ground truth	
		Mispronunciation	Correct pronunciation
Model prediction	Mispronunciation	True positive (TP = CD + DE)	False positive (FP)
	Correct pronunciation	False negative (FN)	True negative (TN)

Recall, Precision, and F_1 are popular metrics for evaluating the performance of MD[13, 14, 24, 25, 41, 40]. Based on the confusion matrix in Table 1, with correct MD counted as true positive (TP), those metrics are calculated as follows:

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

$$F_1 = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{15}$$

From the formula, Recall measures the percentage of mispronunciations that our model correctly detected. Precision measures how many percent of mispronunciations detected by the model that are actual mispronunciations. The two metrics typically exhibit an inverse relationship, as we can aim for higher Recall by increasing the strictness of the model to detect more mispronunciation; however, as a consequence, Precision will decrease. It is more meaningful to look at both Recall and Precision by using F_1, which is the harmonic mean of Recall and Precision.

Furthermore, depending on the contexts, the social cost of Recall and Precision carries different weights [49]. From a language learning perspective, Precision is considered more practically impactful than Recall. Telling the speaker that they mispronounced a phoneme when they pronounced it correctly (FP) will have a more detrimental effect on their learning than failing to detect their mispronunciation (FN) [50]. In other words, higher Precision encourages users to practice more with CaptainA, but higher Precision can lead to lower Recall, as reducing FP would have the tradeoff effect of increasing FN. Consequently, even though CaptainA focuses on detecting mispronunciation with the Recall metric, as long as Recall is higher than a baseline, we chose the model that offers higher Precision and a more lenient phoneme score.

For mispronunciation diagnosis, we also use a common metric, diagnosis accuracy rate (DAR), as follows:

$$DAR = \frac{CD}{CD + DE} \tag{16}$$

where correct diagnosis (CD) and diagnostic error (DE) are cases of correct MD, so that TP = CD + DE, with CD being when the predicted phoneme is the same as the ground truth, and DE being when the predicted phoneme is different from the ground truth.

In MDD, the gold standard for the dataset is that the phonemes used as ground truth are annotated by language experts. If the annotator is not a language expert, it will cast doubt on the reliability of the performance metrics. In our case, the experts only rated the overall task and did not annotate individual phonemes. We rely on the transcripts made by the transcribers to calculate MDD metrics; however, the transcribers were not guaranteed to be language experts and were not required to annotate individual phonemes (see Section 4 below). It becomes more complex when there is a disagreement between the transcriber and the models regarding the phoneme diagnosis.

(In this section and generally, we use graphemes to explain the correct and incorrect pronunciation instead of the IPA. In the Finnish language, graphemes can generally be used to represent phonemes. The short and long vowels, therefore, are replaced with double graphemes.)

For example, the word "pyöreä" has three front vowels (y, ö, and ä) that are often mispronounced as back vowels. Knowing the target word, the transcriber could quickly mark the mispronunciation as three back vowels "puorea". For wav2vec 2.0 models, however, without prior knowledge of the target word, their common predictions are "purea". From the MD perspective, it does not change the performance metrics, as "purea" has three mispronunciation errors at the same position (one deletion and two substitutions). Nevertheless, the diagnosis metrics DAR is lower, even though the transcriber and models agree that the speaker misspoke with the back vowels.

Another example is "ruokapöytä", with the mispronunciation marked by the transcriber as "ruokapyotä". Our models also agree with the transcriber but marked the mispronunciation as "ruokapuutä". In this case, it is difficult to tell which is better, as they sound similar. One remedy is constructing a confusable phone list for phonological error analysis between native speakers and L2 speakers, as done by Wang, Feng, and Meng [39]. The confusable phone list would allow two easily confusable pairs to be recognized as the same MDD error. Another option is to fine-tune our models with the transcript, thus allowing us to learn the annotation style of the transcriber. Both options were not possible due to the lack of data, but they are feasible solutions in the future when we collect more data.

Therefore, we only used DAR as a reference and did not use it to select the optimal model for CaptainA.

3.4 Entropy regularization

We use the output of the fine-tuned wav2vec 2.0 models for phoneme scoring. At that stage, the models are trained with CTC loss. However, this tends to produce overconfident results, where phoneme predictions are either 100% or 0%, and the rest is predicted as blank, creating a spiky output shown in Figure 9. This peaky behavior of CTC output is a common problem [53]. As a result, the phoneme score derived from CTC can only determine whether the user's pronunciation is correct or incorrect, without recognizing any improvement efforts made during practice. This problem is noticeable in difficult sounds, like the Finnish [r], in which a speaker unfamiliar with the rolled R can only make small improvements but will not be able to pronounce it correctly. If CaptainA can not recognize learner's efforts, it can severely discourage them from practicing [11, 49].

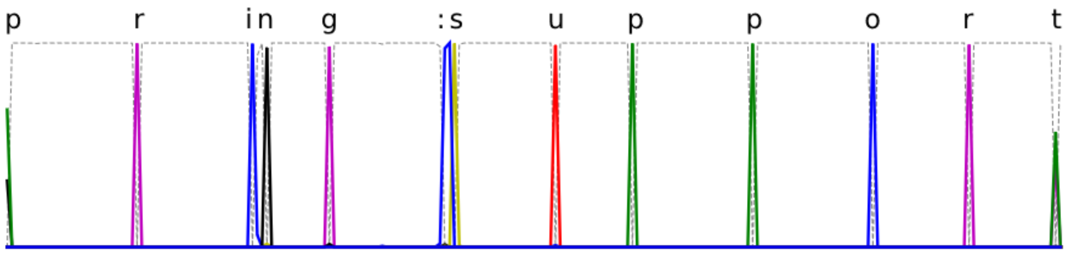


Figure 9: The CTC output [54]. The color lines represent the probability output of individual labels, and the grey dots indicate the blank token.

To make the feedback more robust, we need to reduce the overfitting problem of CTC. Liu, Jin, and Zhang [17] proposes adding a negative maximum conditional entropy regularization term to the CTC loss function L_{ctc}. This regularization term reduces the dominance of one specific path, allowing more exploration and generalization. Recall Equation 6, the set of all paths mapping $\mathbf{x}$ to $\mathbf{y}$ is $\mathcal{B}(\mathbf{y}, \mathbf{x})$, with $\mathbf{x}$ being the phoneme input and $\mathbf{y}$ being the target phoneme, plus the blank token. The calculation of CTC can be rewritten as:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}(\mathbf{y},\mathbf{x})} P(\pi|\mathbf{x}) \tag{17}$$

where the probability of path π is define as:

$$P(\pi|\mathbf{x}) = \prod_{t=1}^{T} P(\hat{y}_t|\mathbf{x}) \tag{18}$$

Because there are many combinations of feasible paths π, once the CTC algorithm finds a path that decreases the loss quickly, it will raise the probability of that path and ignore others, causing peaky behavior. By introducing entropy, other less dominant paths can have a higher probability of being explored. The entropy of the feasible paths knowing input label $\mathbf{x}$ and target label $\mathbf{y}$ is defined as follows:

$$
\begin{aligned}
H(P(\pi|\mathbf{y},\mathbf{x})) &= -\sum_{\pi\in\mathcal{B}(\mathbf{y},\mathbf{x})} P(\pi|\mathbf{x},\mathbf{y}) \log P(\pi|\mathbf{x},\mathbf{y}) \\
&= -\frac{1}{P(\mathbf{y}|\mathbf{x})} \sum_{\pi\in\mathcal{B}(\mathbf{y},\mathbf{x})} P(\pi|\mathbf{x}) \log P(\pi|\mathbf{x}) + \log P(\mathbf{y}|\mathbf{x})
\end{aligned}
\tag{19}
$$

Adding entropy to the CTC loss in Equation 12 as a negative term would prevent the CTC loss from converging quickly:

$$
\mathcal{L}_{enctc} = \mathcal{L}_{ctc} - \beta H(p(\pi|\mathbf{y},\mathbf{x})) \tag{20}
$$

with the hyperparameter β controlling the strength of the entropy regularization.

Lower entropy means the CTC algorithm focuses on only one single path and ignores the others. Higher entropy, on the other hand, results in lower regularized CTC loss and implies more uncertainty and randomness in selecting the optimal paths. Entropy regularization, thus, would slow the convergence to one specific path.

The benefit of entropy regularization is not limited to pronunciation scoring. As entropy increases generation, it also reduces overfitting. Since wav2vec 2.0 models often work in low-resource settings, training could result in overfitting and would benefit from using entropy regularization.

For the MDD objective, it is highly practical to encourage exploration of the less dominant paths. For example, the Finnish vowel harmony rules require that the front vowels (ä, ö, y) and the back vowels (a, o, u) are not used together in a single word. Therefore, a model trained with a native Finnish corpus would overlook the paths containing both front vowels and back vowels. L2 language speakers, however, can break the harmony rule and form impossible paths. Hence, the MDD model should be stimulated to explore those trivial paths in order to detect the learner's mistake.

An example of the effect of entropy regularization on the wav2vec 2.0 model can be seen in Figure 10, with both models having exact same hyperparameters except for entropy regularization. The target word was "ruokapöytä" but the speaker mispronounced it as "ruokapöy<u>d</u>ä". While both models reached the same MDD conclusion that the speaker mispronounced "t" as "d", the model without entropy was overly confident in its assessment, giving 100% probability for all phonemes in "pöydä". In contrast, the entropy $\beta = 10\%$ model also predicts with 100% probability for "p", "ö", "y" and "ä", but only gives 75% for "d", with the remaining 25% recognized as "t". From the learner's viewpoint, the former model would give them a score of 0 for the phoneme "t", and it would take a more significant improvement for their effort to be recognized. On the other hand, the later model can recognize a speaker's effort to pronounce the phoneme "t", even though it is clear that the speaker mispronounced it.

Because entropy regularization reduces the peaky behavior, it correspondingly reduces the blank labels predicted by the model, as we can see on the right part of Figure 10. The [PAD] label is the blank label used in our wav2vec 2.0 model; with a higher value of β, the probabilities of the [PAD] tokens are lowered. The CTC algorithm relies on the blank token to decode the predicted labels from the probability

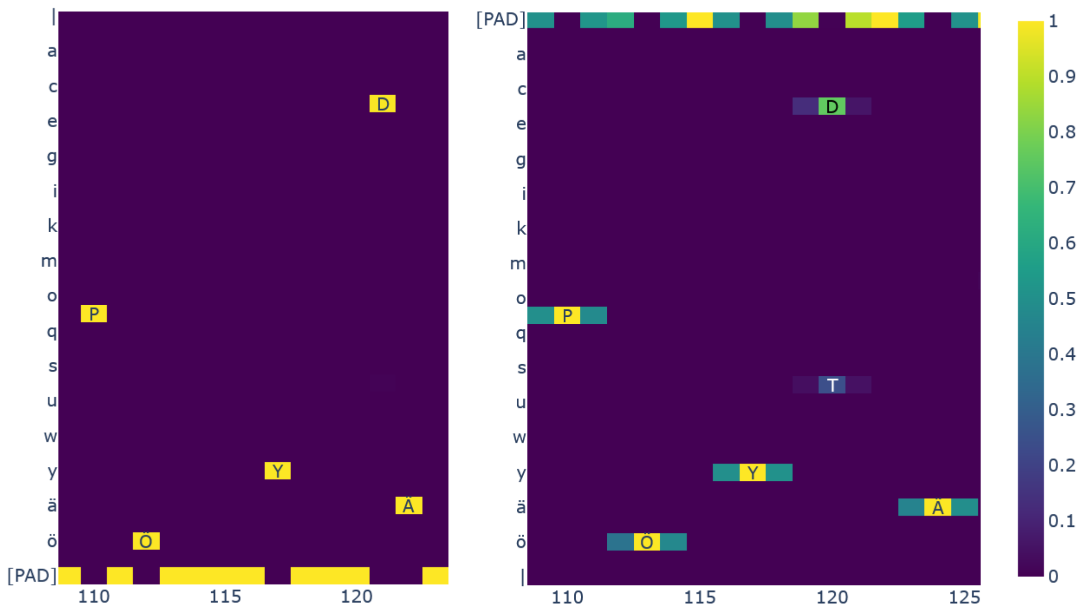


Figure 10: Wav2vec 2.0 labels probability output of an audio sample that mispronounced “pöytä” as “pöydä”. The vertical axis is all labels in the vocabulary, with [PAD] as padding or blank label. The horizontal axis is the frame number in the audio. The color represents the probability of individual label per frame. The left output is from X-G0 with no entropy $\beta = 0$, and the right output is from X-G10 with entropy $\beta = 10\%$. For more information on each model configuration, see Table 6.

output. Therefore, there is an upper limit for β until the benefit from generalization is offset by the error in decoding. Liu, Jin, and Zhang [17] suggested $\beta = 20\%$, however, their experiment was not applicable to our use case, and therefore we need to find the optimal β value for our MDD task.

4 Dataset

The main challenge of this project is the lack of spoken L2 corpora for the Finnish language. Moreover, for MDD purposes, the L2 corpus needs to be rated by language experts, who should also provide annotations of mispronunciation at the phoneme level. There are multiple L2 corpora with detailed phoneme annotation for English [44, 55, 56, 57]. Notable, the L2_ARCTIC and the speechocean762 are publicly available and include phoneme pronunciation annotations by experts for MDD tasks (see Figure 7 for an example of the annotation).

However, for the Finnish language, the availability of L2 speech dataset is more limited. The only dataset we have comes from the Digitala project [9], with the majority of the samples from Digitala being free-form spontaneous speech, which is unsuitable for pronunciation practice for beginners. Furthermore, Digitala samples were not annotated at the phoneme level, which makes MDD more challenging. Additionally, we did not have the resources to create our own MDD dataset.

As a result, the selection of datasets becomes an experimental and deliberate task. For datasets, there are two distinguished groups: the publicly available pre-training and training data from native-level speakers in multiple languages without any rating on their pronunciation, and the fine-tuning and test set from L2 Finnish speakers in the Digitala project, with task-level rating.

As we investigated and explained below, the corpora used in this project were not the perfect match for the MDD task, but they were the best alternative options available. After the pilot phase to gather users' opinions, we could collect better data from L2 speakers, potentially improving CaptainA's performance in the future.

4.1 Pre-trained model

Since we use the wav2Vec 2.0 model for our project to benefit from the speech representation learned during pre-training, we need to select our pre-training dataset. There are two options: XLS-R [58] and Uralic.

Table 2: The seven most popular languages in the XLR-R unlabeled pre-trained model, in thousand hours.

	English	German	French	Spanish	Italian	Polish	Dutch
Hours	69.5	25.4	24.0	22.3	21.9	20.9	20.1

The XLS-R corpus contains approximately 436,000 hours of unlabeled speech data from multiple language groups, with the majority of data from VoxPopuli [59]. The VoxPopuli corpus comprises samples from European parliament speeches, resulting in a significant number of samples being at a native or near-native level and used in the political domain. While XLS-R mainly includes high-resource languages (see Table 2), it also covers other low-resource languages for a total of 128 languages.

The Uralic pre-trained data in Table 3 is another subset of VoxPopuli that only covers the Uralic language family, including Hungarian, Finnish, and Estonian,

Table 3: Uralic pre-trained model's unlabeled data in thousand hours.

	Hungarian	Finnish	Estonia	Total
Hours	17.7	14.2	10.6	42.5

totaling approximately 42,500 hours. In terms of quantity, the Uralic dataset is only approximately one-tenth of the XLS-R. However, Uralic focuses solely on the Finnish-related group and could potentially provide better results for Finnish NLP tasks.

We did not pre-trained the models by ourselves. Instead, we used pre-trained models from Facebook for both XLS-R[1] and Uralic[2] models. We primarily experimented with large models (those with about 300 million parameters) as the larger models (1 billion parameters or more) require significantly more computing and memory resources to be delivered to our users. Instead, we used public models from an independent group, Finnish-NLP [60], for our baseline. The 1BIL model[3] has 1 billion parameters and was reported as the best model for Finnish ASR. The 300M model[4] has about 300 million parameters, the same size as our models, and was used for comparison purposes. These models were also pre-trained with XLS-R. While Finnish-NLP used a language model for decoding, their performance in the Common Voice 7.0 [61] test set was also superior when used without the language model. As our thesis is for MDD and not ASR, no language model was used in our work.

4.2 Training dataset

Since we did not have enough L2 data, we instead used the native Finnish dataset to train our wav2vec 2.0 model. Our rationale is based on the fact that the wav2vec 2.0 model can learn speech representations from the dataset. By learning speech features from native Finnish speakers, the models are expected to be able to identify deviations from the standard speech; thus, they can detect mispronunciations made by L2 speakers. However, not all deviations are from mispronunciation. They could arise from differences in spontaneous speech and read-aloud speech [62], differences in dialects, or differences in speakers' backgrounds. Therefore, it is crucial to select a corpus that aligns with our target use case of short read-aloud samples made by young speakers.

There were two spoken Finnish corpora available for consideration. The first one is the Lahjoita puhetta ("Donate Speech") [63]. While the corpus contains native and non-native speakers, we can filter by the metadata to select the best samples for our purpose. However, the dataset was not considered for our project, for several reasons. Firstly, the dataset comprises spontaneous spoken Finnish, which has significant differences compared to read-aloud Finnish [62]. Secondly, the recording duration for

[1]https://huggingface.co/facebook/wav2vec2-xls-r-300m
[2]https://huggingface.co/facebook/wav2vec2-large-uralic-voxpopuli-v2
[3]https://huggingface.co/Finnish-NLP/wav2vec2-xlsr-1b-finnish-lm-v2
[4]https://huggingface.co/Finnish-NLP/wav2vec2-xlsr-300m-finnish-lm

this dataset is longer, with a median value of about 40 seconds, and over 50% of the samples are longer than one minute. For reference, CaptainA aimed to limit recordings to a maximum of 8 seconds due to server limitations. Thirdly, speakers who donated their speech to the corpus were encouraged to use their colloquial speech. Using the Lahjoita puhetta dataset could make classifying the speech deviations made by L2 speakers into mispronunciation and other errors more complicated.

After careful consideration, we selected the Finnish Parliament corpus [64] for training our wav2Vec 2.0 models. The corpus and speakers' metadata are publicly available together, allowing us to selectively choose samples to match our target use case:

- The dataset does not have statistically significant dialect problems [65]. This would help our model provide feedback based on the standard Finnish accent.
- We only take samples with a maximum duration of 15 seconds to fit with CaptainA's maximum recording length of 8 seconds.
- We filter for read-aloud samples by choosing only slow or average speech samples.
- We select the speaker's age (publicly available) to fit with our target users (50 years old or younger).

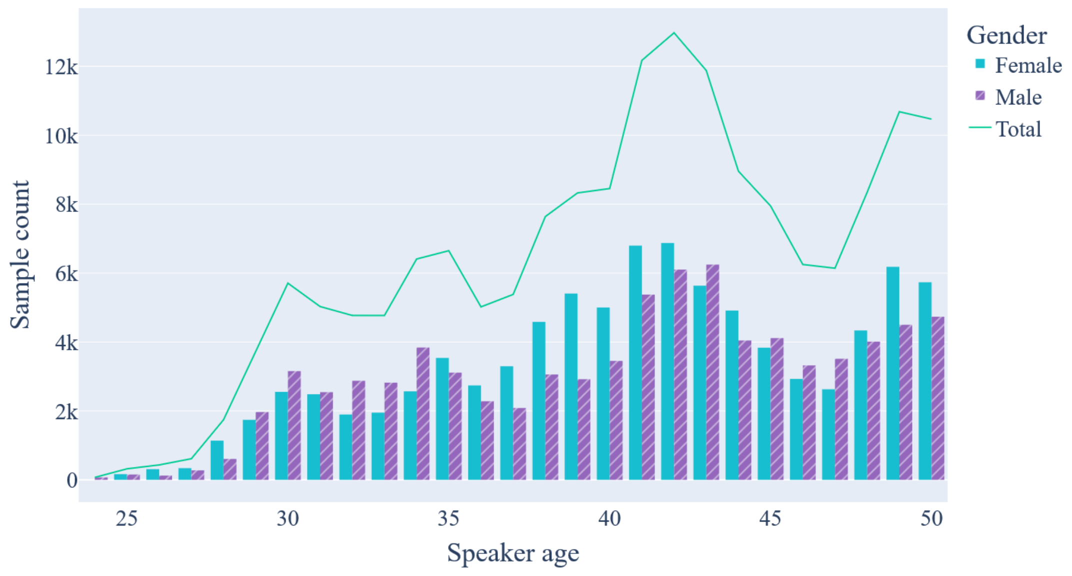


Figure 11: Training set distribution by gender.

After filtering, we observed that there was no significant difference in the number of samples between genders (Figure 11). In total, we had approximately 281 hours of the Finnish Parliament corpus for training, split into a 75% training set and a 25% evaluation set. Our training data have a median duration of 5.59 seconds, and the interquartile range (the difference between the 75% and 25% of the samples) of audio length fell within [2.95, 8.34] seconds.

Training with the Finnish Parliament corpus means our model inherits the same biases identified by Virkkunen et al. [65]. Regarding ASR, we expect female speakers, younger speakers, and speakers with a Master's degree or higher will be recognized better by our models. While ASR performance is measured in word error rate (WER) or character error rate (CER), it could be indirectly related to Recall and Precision. Lower CER signifies better speech recognization, which could result in higher Precision in samples with multiple mistakes. As beginner Finnish learners tend to make more pronunciation errors, we expect the corpus biases to affect them the most.

Furthermore, over 50% of the samples in the Finnish Parliament corpus have a duration of 5 to 10 seconds, with samples shorter than 5 seconds making up less than 15% of the corpus [65]. We expect our MDD model to work best for sentence pronunciation or long words. Short word pronunciation that takes 2 to 3 seconds might not have the same performance as longer samples. Unfortunately, the Finish Parliament corpus was the best alternative available at the time. The fact that our models work best with long audio samples was taken into account when developing CaptainA, and we provide multiple examples that encourage the user to practice with longer phrases or sentences.

The Finnish Parliament corpus also has various linguistic and editorial practices in parliamentary reporting [66]. While the main principle is to minimize alterations, the transcripts contain several changes from spoken language to written language to maintain consistency, comprehensiveness, and flexibility in the parliamentary reports. A summary of the types of speech changes is in Table 4. Because the editorial rules in the Finnish Parliament data affect the wav2vec 2.0 model output and subsequently influence MDD results, it is critical to thoroughly understand these rules.

The first two rules practically convert all spoken varieties of Finnish into their formal written form. Alhough CaptainA aims to provide MDD for read-aloud written Finnish, the result might confuse users who are not aware of these conversions. For example, a user could try to practice pronouncing "mennään", but they may receive an incorrect assessment as the model tends to understand the sound as "menemme" (as it turned out, the final "n" in "mennään" is often omitted by the MDD model). Furthermore, spoken Finnish is currently taught at beginner levels [67], but any effort to practice spoken Finnish will be rated as their written form ("mä" will be automatically converted as "minä").

The remaining rules cause other problems for short words. Short words like "tääl", "oli", or "tota" are intentionally ignored in MDD. However, even for a word that is not changed by the Finnish Parliament practices, it could potentially be misunderstood as self-corrections or blunders when the speaker is not fluent in pronouncing them. The word "yritys" is one clear example; as the combination of "y" and "r" into [yr] is difficult to pronounce for an L2 beginner, the speaker may sound similar to making a slip of the tongue or self-correction. The MDD model often ignores the speaker's effort to pronounce the word "yritys". The problems are more obvious in short words; as the speaker uses longer words or a sentence, it becomes clear that they are not making any mumbling sound or self-correction.

Table 4: Changes between speech and transcript, as summarized from language-regulatory practices in Finnish parliamentary reporting [66].

What is changed	Example
Regional dialect in phonological features	mä, mää, mie → minä ("I")
Some morphological features more typical of spoken varieties	me mennään → me menemme ("we go")
Certain syntactic features of spontaneous speech	tääl oli edustaja Turunen epäili ("here was MP Turunen suspected") → edustaja Turunen epäili ("MP Turunen suspected")
Omission of selected particles	tota ("kind of"), niinku ("like")
Self-corrections	päästöttö-...päästöllisen ("one witho-...with emission" → päästöllisen ("one with emission")
Minor blunders and slips of the tongue	sukupuolennvaihdos ("sex change") → sukupolvenvaihdos ("generational turnover")

Aside from the reporting practices, the construction of the Finnish Parliament corpus also causes some problems for the MDD task. As there are multiple ways to pronounce numbers in spoken Finnish, the corpus assumes that every speaker only uses the standard pronunciation for numbers. While this is standard procedure for ASR, it could cause problems in MDD if speakers want to use spoken language. For example, when pronouncing 5.80 using spoken Finnish as "viis kaheksankyt", the models predict the audio sample as standard Finnish "viisi kahdeksankymmentä". This affects the probability output of the wav2vec 2.0 models, and as a result, both the MD and diagnosis will not be correct.

These disadvantages are not limited to MDD. ASR models primarily trained with the Finnish Parliament corpus will presumably exhibit similar behaviors in their predictions, especially since ASR is more likely to process spoken Finnish than MDD. Nevertheless, the combination of parliamentary reporting practices and the lack of speech fluency by L2 speakers make these problems become more noticeable in MDD. As we explained in Section 3.3, for CaptainA, high FP errors and low Precision can frustrate L2 speakers and discourage learning.

It is important to note that even though we discussed the drawbacks in MDD of the Finnish Parliament records and the construction of the corresponding corpus, it does not mean we criticize any of those works. They provide an incredibly valuable dataset for the public, and the difficulties in MDD arise because they have different objectives. The Finnish Parliament record strives to accurately record in written form what happened in the parliamentary sessions, while the corpus construction mainly seeks to publicly provide the largest Finnish transcribed speech data for the ASR task. Ideally, we would prefer to train the model with a short, read-aloud dataset constructed specifically for MDD. In the early steps of constructing our MDD corpus, the Finnish

Parliament corpus was the nearest alternative we could find. Also, the initial goal of this project is mainly for read-aloud Finnish, whereas most of the problems mentioned above stem from using spoken Finnish in MDD.

4.3 Fine-tune and test data

For fine-tuning and testing, we use data from the Digitala project [8]. The corpus consists of data from digitized language tests taken by L2 students in Finnish and Swedish. The Digitala project has different phases, and we mainly use newer data described in Al-Ghezi et al. [9], as these data have short Finnish read-aloud samples that fit with our target. The Digitala dataset consists of free-form speech, long read-aloud speech, and short read-aloud speech. Each audio sample has holistic scores made by a group of specialized human raters on the following criteria: CEFR, fluency, and pronunciation. It is important to mention that the scores, even for pronunciation, are for the whole task, not individual phonemes.

4.3.1 Test set

Since the mispronunciation app requires the user to read a written word/sentence aloud, we only use the short read-aloud part as our test set. The CEFR levels are not applicable for all read-aloud tasks, and fluency is only available when reading aloud a phrase or sentence. For reference, we only use the median pronunciation score, rounded down into four levels from 1 to 4, with 4 being the best (Table 5). As data is not annotated at the phoneme level by experts, we could not use the pronunciation level in the Digitala corpus to determine our MDD pronunciation ratings. However, the dataset was also transcribed by transcribers. While we cannot guarantee the transcribers are language experts, it should be noted that the transcripts did capture mispronunciations made by the speakers.

Table 5: Test set distribution for overall pronunciation level, with 4 being the best.

Level	Number of phonemes
1	2,183
2	7,097
3	6,934
4	576

In total, we used 768 short read-aloud samples from the Digitala dataset as our test set, with a total duration of about 60 minutes. It consists of the following read-aloud tasks:

- kahviautomaatti.
- pyöreä ruokapöytä.
- korkea kerrostalo.

- fazer toimii kahdeksassa maassa.
- myöhemmin yritys on kasvanut huimasti.
- pupu puhuu puppua.
- tuuli tuli kotiin.

Besides the problem with the phoneme annotation, we also noticed several future improvements that can increase the quality of our test set. The first problem we noticed is that the short read-aloud test set did not cover all phonemes in the Finnish language; specifically, the test set is missing “nk” [ŋk] and “ng” [ŋː]. The first three tasks in the test set were the most suitable for testing read-aloud pronunciation for L2 beginners. And the 4th and 5th tasks were used to expand our application to lengthy sentences for advanced learners. However, the last two tasks focused on tongue twisters or phonetically challenging phrases instead of proper read-aloud tests and were not suitable for testing read-aloud pronunciation. Despite that, as we lack data for testing model performance, we included all tasks in our test set but noted that the test set quality could be improved further with more L2 data collected.

We also found cases where different models, trained with different datasets, agreed with each other but disagreed with the transcript. Checking the audio sample, we tend to agree with the models’ prediction when paying attention to individual phonemes. For example, studying a sample when an L2 speaker tried to pronounce the word “pyöreä”, the wav2vec 2.0 models predicted “puoria” while the transcript was “puorea”, with a distinct “i” sound pronounced in the record. In this particular case, the models and transcript agree on the basis of the pronunciation mistakes (front vowels and back vowels), but on the phoneme level, the transcript is less reliable as it missed another mispronunciation.

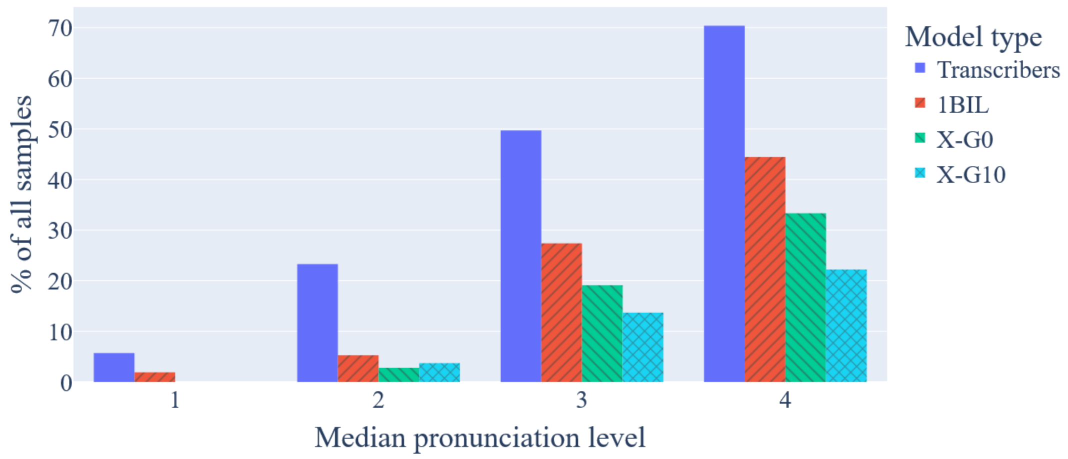


Figure 12: Percent of read-aloud samples without any mispronunciation marked, grouped by overall pronunciation level. The lower percentage indicates a stricter level. Models are defined in Table 6.

When studying the quality of the transcript for MDD purposes, we note that the transcribers are more lenient in detecting mispronunciation mistakes. This is expected, as it was not their primary task. A comparison of the strictness of transcribers and wav2vec 2.0 models is shown in Figure 12. We can see that while wav2vec 2.0 models have differences, they are noticeably stricter than human transcribers. Because strictness directly affects performance metrics, it is expected that all models will have low Precision and high Recall.

4.3.2 Fine-tuning set

For the test set, the quality of the phonemes error detected by the transcribers is less impactful, as all models would share the same problem. However, it is meaningful to investigate the quality of the test set for MDD, as we would use the rest of the Digitala corpus for fine-tuning. For example, mispronunciations for [r] are often ignored by transcribers but are easily detected by our models. Fine-tuning models with those samples would potentially lead to our models not detecting mispronunciation for [r] made by L2 speakers.

Therefore, choosing which samples from the rest of the data in Digitala for fine-tuning also depend on the quality of the phoneme annotations. For reference, from Figure 12, we can look at the number of samples without any errors for median pronunciation level 2. Median pronunciation level 2 means that, even with disagreement, experts would rate the sample with a maximum score of 3 (cases where one expert rates a pronunciation score of 1 and the other rates 4 rarely occur in the test set). This means that for the median pronunciation level 2, according to the experts, the samples must have some problems (score of 2) or at least minor problems (score of 3) in pronunciation. Instead, the transcripts indicate at least 20% of those samples did not contain any mistakes.

The results in Figure 12 are from the short read-aloud samples, where the transcribers only needed to pay attention to a few seconds of recording and had the knowledge of the target transcript. The free-form spontaneous tasks in Digitala are typically longer than 30 seconds without any target transcript, and thus potentially contain more unmarked mispronunciations. Fine-tuning with unmarked mispronunciations could cause our models to fail to detect mispronunciations. As a result, we could not use the free-form speech, a large part of the Digitala corpus, for fine-tuning.

Consequently, we can only select the best pronunciation samples, with a median pronunciation level of 4, from the long read-aloud speech in Digitala for fine-tuning. The long read-aloud samples generally have a duration of 6 to 8 seconds, with the shortest task being "fazerin tuotevalikoima on monipuolinen" with about 4 seconds of recording. After selection, the fine-tuning dataset contains 364 samples made by L2 speakers whose samples were not included in the short read-aloud test set. All samples have a pronunciation rating of 4, totaling about 42 minutes, split into a 90% fine-tuning set and a 10% evaluation set.

However, note that even the pronunciation samples with the highest rating still have mispronunciations. As seen in Figure 12, for rounded-down median pronunciation level 4 (the samples with the best score), the lenient transcribers also mark about 30%

samples with mispronunciations. This is mainly because the rubrics for the Digitala project did not require perfect pronunciation for the highest pronunciation score.

5 Experiment

Recent works have shown that E2E systems are effective in ASR in general and MDD for L2 speakers [13, 25, 68, 69]. As we explained in Section 2, while there are multiple approaches to E2E ASR and MDD systems, we focus on Transformer, CTC, and the state-of-the-art wav2vec 2.0 model for our application. The main reason is that the Digitala dataset has limited data and would benefit from wav2vec 2.0 advantages in a low-resource setting.

Since our work in MDD for Finnish L2 speakers is novel, we could not find a baseline to evaluate our models' performance. Instead, we use the reported best ASR model at the time of this thesis, the 1BIL model mentioned in Section 4 as our baseline. We also use the 300M model, a similar model with the same number of parameters as our models, to estimate the impact of the different configurations on the MDD performance.

The short read-aloud test set was not available at the start of the project, and initially, the long read-aloud set was used as both the fine-tuning set and the test set. With the short read-aloud data from Digitala, we have a larger and better-matching test set, and subsequently can arrange a larger fine-tuning set for our experiment.

We experimented with multiple combinations for MDD, mainly pre-training dataset, vocabulary type, and entropy strength β%. Descriptions of the models' names and their configurations are summarized in Table 6.

Table 6: Models configuration summary

Model	Type	Pretrain	Vocabulary	Parameters	Entropy β
1BIL	Baseline	XLS-R	Grapheme	1bil	0%
300M				300mil	
X-G0	Train	XLS-R	Grapheme	300mil	0%
X-G5		XLS-R	Grapheme		5%
X-G10		XLS-R	Grapheme		10%
X-G20		XLS-R	Grapheme		20%
X-P10		XLS-R	Phoneme		10%
X-H10		XLS-R	Hybrid		10%
U-G10		Uralic	Grapheme		10%
U-P10		Uralic	Phoneme		10%
X-G10-FT	Fine-tune	XLS-R	Grapheme	300mil	10%

We started our experiment with Facebook's XLS-R pre-trained model available in HuggingFace [38]. According to the model performance comparison made by Finnish-NLP [60], we found that the Uralic pre-trained models outperformed the XLS-R in Finnish ASR, with or without a language model. Therefore, we also conducted our experiment with the Uralic models to evaluate their performance in MDD.

We also experimented with a smaller model size[5], the 95 million parameters

[5]https://huggingface.co/facebook/wav2vec2-base

BASE model used in the wav2vec 2.0 paper [16]. The model was pre-trained with about 53,000 hours of unlabeled English speech from Libri-light [70]. However, the lack of language diversity, smaller model size, and less pre-trained data resulted in significantly lower performance in MDD. The results of the 95 million parameters models were not included in this thesis, as we did not find a similar pre-trained model that could be compared with the XLS-R or Uralic pre-trained models.

Our experiment aims to find the most practical model for MDD. The model size is not for consideration, as the 1 billion parameters model required too many computing resources, and the 95 million parameters model underperformed. Our wav2vec 2.0 model size is the large model with about 300 million parameters. As explained in Section 3.3, our main metrics are Recall, Precision, and F_1 on the short read-aloud test set. As Finnish words can generally be formed by compounding multiple words together, we use CER instead of WER for a better understanding of the MDD models. We also look at DAR for reference, but it is not used to select the optimal model. There was no ultimate metric to select the model, as we also need to consider the ability to recognize the learner's efforts, measured by the strength of the entropy regularization (explained in Section 3.4). Eventually, we decided that the optimal model is the one that has a reasonable Recall (at least comparable to the baseline), with the best combination of Precision and high β‰ entropy parameter.

5.1 Vocabulary

To take advantage of the phonetic character of the Finnish language, and to ensure compatibility with other ASR models, we first developed our models using graphemes. Using phonemes, however, would give users a better understanding of their pronunciation. We also noted that, while the foreign alphabet like "z" and "x" are included in the grapheme models, they rarely had high probability and instead were predicted as other letters. For example, "x" was often predicted as "ks" with Finnish samples. Our objective was Finnish beginner pronunciation practices without the need for practicing loanwords. Moreover, changing to phonemes would reduce our model vocabulary size by swapping foreign letters with their alternative Finnish counterpart. The rules to change from grapheme to phoneme are as follows:

- add "ŋ" to represent "nk" (ŋk) and "ng" (ŋŋ).
- "å" will be replaced with "oo".
- remove "c".
- "q" will be replaced with "k".
- "w" will be replaced with "v".
- "x" will be replaced with "ks".
- "z" will be replaced with "ts".

The rules are applied to all datasets used in the project. As a result, we reduced our vocabulary size from 33 to 27 for phoneme models. After our preliminary result, we also tested with a hybrid model, which is similar to grapheme models but with the symbol “ŋ” added to the vocabulary. The inclusion of “ŋ” was recommended by Finnish teachers in the Kiebuusti project and is used to provide correct phonetic feedback to learners.

We also removed other extra tokens used in ASR models, like the “start of sentence” token and the “end of sentence” token. A summary of vocabulary for each model type can be found in Appendix B. The grapheme models have a similar vocabulary to the 1BIL and 300M models for comparison purposes, whereas the phoneme models would have the minimal vocabulary possible for Finnish MDD, to reduce computing power.

5.2 Hyperparameters & Entropy regularization

Our hyperparameters are the same as the 1BIL model made by Finnish-NLP [60] whenever applicable. Firstly, we want our models to be comparable with the baseline models. Secondly, the Finnish-NLP group applied several dropout settings in their hyperparameters. Similar to entropy regularization, higher dropout reduces overfitting during training and encourages generalization. As our models are mainly trained on native Finnish samples and tested on unseen L2 Finnish samples, we found that the dropout applied by Finnish-NLP improves our MDDs performance. Dropout reduced overfitting and expanded the scoring range for our application (see Figure 13). Unless stated otherwise, all models use the same hyperparameter configuration. The detail of hyperparameters is included in Appendix A.

For entropy regularization, we experimented with several configurations for 300 million parameters models, with $\beta = 5\%$, $\beta = 10\%$, and $\beta = 20\%$. Initially, we prefer a bigger β value, as they provide a better rating system for CaptainA. However, we noticed that a small amount of entropy regularization also improves MDD evaluation. For our work, we found that β within the range of $[5\%, 10\%]$ yields the best overall result in MDD. Considering the main purpose of entropy regularization is to provide an extra scoring range, and the added benefit of better MDD performance, we experiment mainly with $\beta = 10\%$.

The entropy regularization loss $H(p(\pi|\mathbf{y}, \mathbf{x}))$ from Equation 19 is based on the code provided by Liu, Jin, and Zhang [17]. We implemented the entropy loss into the model by using a customized model, extended from the standard wav2vec 2.0 model in the Transformers library [71]. The loss, $\mathcal{L}_{enctc}$ is the standard CTC loss function combined with entropy regularization and β following Equation 20. During training and fine-tuning, the customized model minimizes the $\mathcal{L}_{enctc}$ loss instead of the standard $\mathcal{L}_{ctc}$ loss.

We also tested entropy regularization with the 1 billion parameter model. However, we found that the optimal range of β may not be the same. Due to the limitation of computing resources and the impracticality of a bigger model, we did not tune for optimal β value. It should be noted that models with different configurations may need different β values to obtain the optimal result.

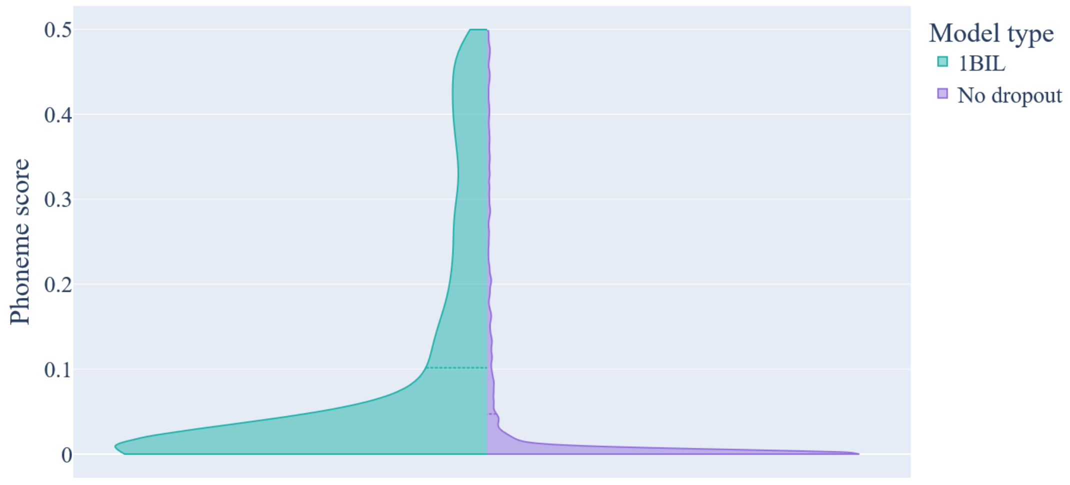


Figure 13: Phoneme score distribution for short read-aloud phonemes with a score less than 0.5. The model without dropout is an internal model with the same size as 1BIL, trained on the Lahjoita puhetta dataset. The model without dropout tends to be overconfident and mainly gives close to 0 to all mispronunciations, whereas 1BIL with dropout can seldom give different scores.

5.3 Training and fine-tuning

The training for our project was performed using computer resources within the Aalto University School of Science "Science-IT" project. Due to the amount of training data, we used the Nvidia Tesla A100 with 80GB memory. One model would generally take two days for training with entropy regularization. For comparison, we fixed the number of epochs in training to 10 epochs for all models. Larger epochs also did not improve the models' performance in MDD, as it led to overfitting.

We also fine-tuned our trained models with the long read-aloud dataset from Digitala, and included the result for a fine-tuned model (X-G10-FT) in Table C1. Because the data for fine-tuning is limited, the resources we spend on fine-tuning are minimal. Appendix A contains details on the hyperparameters we used for fine-tuning. Generally, we tried to use the same hyperparameters, but since our data is smaller, we also used a smaller batch size and epochs. We found that large epochs in fine-tuning, while improving the model's CER, did not significantly improve MDD performance. Since the fine-tuning dataset potentially has undetected mispronunciations, the fine-tuned model has much lower Recall compared with the additional gain in Precision. Since the fine-tuning dataset contains possible wrong labels, we would prefer our models retain the knowledge they learn during training with native Finnish speakers. Therefore, we also experimented with a much lower learning rate; however, we experienced the same trade-off behavior between Recall and Precision.

5.4 Practical metrics

As we mentioned in Section 3.3 and Section 4.3, our dataset was not ideal for MDD, especially for providing phoneme diagnosis. From our experiment, the DAR of our models shows that the diagnostic accuracy was not high, and we would want to avoid providing incorrect diagnoses to the users. Furthermore, when looking at the type of error from the Levenshtein algorithm, we found that it is more practical to focus on substitution errors for diagnosis. The insertion error is generally caused by skipping a target phoneme, or pronouncing a short phoneme instead of the long phoneme. Similarly, deletion errors are often caused by pronouncing extra phonemes not in the target word, or pronouncing the phoneme longer than they should.

The Finnish language has distinct short and long durations for vowels and consonants, and MDD applications should be able to deliver diagnoses on those types of errors. However, beginner speakers tend to easily make those mistakes, as they often speak too fast or too slowly. When providing corrective feedback in CAPT, Engwall and Bälter [11] recommended focusing on the main error type. We, therefore, focus on the diagnosis quality of mispronouncing a phoneme as another phoneme, e.g. front vowels mispronounced as corresponding back vowels, and skip other errors. In other words, we should concentrate on diagnostic accuracy for substitution errors with a practical metric:

$$DAR_S = \frac{CD_S}{CD_S + DE_S} \tag{21}$$

where CD_S and DE_S are similar to CD and DE, but only in case of substitution errors.

We found that, for our case, DAR_S is higher than DAR, and the mispronunciation diagnosis based on substitution errors is more reliable. The diagnosis function provided by CaptainA is optional, so we only show the diagnoses in obvious cases when we detect substitution errors from the mispronunciation.

Readers should be reminded that, while CaptainA's diagnosis is only for substitution errors, other errors are still identified with the basic MD. For example, if the user pronounces only one "a" instead of the target "aa", the mispronunciation is marked in the second "a"; however, CaptainA will not provide any detailed diagnosis for that type of mispronunciation.

We also use another set of practical metrics to measure the ability to give positive feedback by CaptainA. As explained in Section 4, the dataset only provides binary ground truth, with either correct or incorrect pronunciation. And in Section 3.4, we use entropy regularization to be able to give additional score feedback as a middle ground to the user, to encourage them to practice. However, we could not verify the correctness of the new score range as the dataset does not provide such information. Instead, we practically aim for the distribution of phoneme scores with smaller peaks in scores of 0 and 1. For example, in Figure 13, we would prefer the distribution on the left to the distribution on the right.

For that purpose, we calculated the standard deviation (SD), kurtosis, and skewness of the phoneme score distribution for each model. SD is the square root of the variance, and both measure the general dispersion of phoneme scores. We would prefer a large SD as it indicates the pronunciation scores are not concentrated. However, as the distribution is concentrated on the two tails (near 0 and 1), SD might not correctly capture the score dispersion. Kurtosis and skewness metrics offer quantitative information about the shape of our score distribution [72]. Compared to a normal distribution which has kurtosis of 3, we would prefer smaller kurtosis as it indicates lighter (thin) tails, and heavier (thicker) shoulders. The preferred skewness values are dependent on the type of pronunciation. We use Skew_M to measure the skewness of the score distribution of all mispronounced phonemes. By definition, the phoneme scores would be less than 0.5, and the distribution would be skewed to the right, concentrated on 0; we would prefer a lower skewness number, as it indicates less skewness to the right [73]. On the other hand, we use Skew_P to measure the skewness of the score distribution of all correct phonemes. The score distribution is skewed to the left, concentrated on 1; and we prefer a bigger skewness value. Effectively, for skewness on mispronunciation score distribution and correct pronunciation score distribution, the closer the value to 0 (normal distribution skewness), the better.

6 Results

6.1 Settings comparison

All results of our experiment are summarized in Table C1 and Table C2. As we experimented with multiple settings, it is also beneficial to look at the result with a change in configuration. Table 7 shows the difference in MDD performance from different pre-trained models and different vocabulary.

Table 7: Speech models' performance with different pre-train models and different vocabulary settings.

Model	**CER**	**Recall**	**Precision**	**F_1**	**DAR**
X-G10	**21.2%**	63.1%	**29.4%**	**40.1%**	**55.3%**
U-G10	30.4%	**64.3%**	23.4%	34.3%	40.3%
X-P10	**21.3%**	63.2%	**27.3%**	**38.1%**	**54.9%**
U-P10	29.6%	**66.8%**	22.6%	33.8%	40.3%

The Uralic pre-trained models' significant underperformance in both ASR and MDD was unexpected. Although U-G10 and U-P10 showed a slightly higher Recall rate, this could be an unintended consequence of their significantly worse CER rather than better MD. We believe the XLS-R outperformed Uralic because of its exposure to non-Uralic languages. The test set contains multiple L2 speakers with different backgrounds, with the majority not familiar with the Uralic family. Because we trained our models with a native Finnish dataset, our models can only be exposed to the foreign language groups via the pre-trained model. Recall the literature review on the wav2vec 2.0 model in Section 2; the features learned during pre-training are frozen during training and fine-tuning. Therefore, the XLS-R models retained some non-Uralic features learned from the pre-training. The pre-trained Uralic model does not offer that valuable exposure.

A similar situation occurred when we looked at the performance of phonetic models in Table 7. The grapheme model X-G10 outperformed the phoneme model X-P10 in MDD, with distinctly higher Precision. We were expecting a similar performance, as there were no significant differences between grapheme and phoneme in the test set. Investigate the training data, we found that Finnish parliament speakers often use foreign phonetics for foreign words, e.g., the "z" in "Fazer" is pronounced as Finnish [ts], but both the "z" in "Zyskowicz" can be pronounced as English [z]. Therefore, even though uncommon alphabet like "z" or "c" were not used in the test set, leaving them in the vocabulary helped models learn the correct features of Finnish phonetics during training.

On the positive side, we also benefited from the generalization of entropy regularization. From Table 8, we can clearly see that a small amount of entropy β within [5%, 10%] increases the MDD performance. The increase in Recall is promising, as we can see that the X-G5 and X-G10 did not have worse CER, suggesting the higher Recall is from better MDD and not worse ASR. The most valuable and practical increase is in Precision, as we considered the cost of FP to be more severe than FN (see

Table 8: Speech models' performance with different β entropy strengths.

Model	CER	Recall	Precision	F_1	DAR
X-G0	20.9%	61.1%	26.7%	37.2%	59.4%
X-G5	**19.5%**	63.1%	**30.0%**	**40.6%**	**59.9%**
X-G10	21.2%	63.1%	29.4%	40.1%	55.3%
X-G20	29.5%	**66.6%**	23.3%	34.5%	40.3%

Section 3.3). However, X-G20 showed that higher β value results in worse performance in MDD. X-G20 has significantly worse Precision, and the higher Recall could be caused by higher CER. As mentioned in the previous section, one possible explanation is that the stronger entropy parameter can remove the blank token predictions, while the CTC algorithm needs the black tokens for decoding (see Figure 4). It could also be possible that bigger entropy strength needs different hyperparameters than what we used for our experiments.

Table 9: Advantages of higher entropy regularization in providing diverse scores.

Model	DAR_S	SD	Kurtosis	$Skew_M$	$Skew_P$
1BIL	**71.8%**	0.14	3.53	1.37	-3.73
300M	63.3%	0.14	3.60	1.40	-3.50
X-G0	68.9%	0.14	3.80	1.45	-3.50
X-G5	69.2%	**0.16**	2.71	1.07	-3.09
X-G10	68.9%	**0.16**	**2.54**	**0.99**	**-2.75**
X-H10	67.7%	**0.16**	2.59	1.04	-3.16

It should be noted that, while the entropy $\beta = 5\%$ yielded the best MDD performance, we instead chose the higher entropy $\beta = 10\%$ models. Recalling that the initial purpose of entropy regularization is to obtain a better scoring range for our MDD application, the small difference in performance between X-G5 and X-G10 allows us to focus more on the practicality of those models. Looking at Table 9, we can see that all models with entropy have higher SD, indicating entropy disperses the phoneme scores. Looking at kurtosis combined with $Skew_M$ and $Skew_P$, we can see X-G10 has a better score distribution shape than X-G5, with light tails and fat shoulders. The $Skew_M$ for mispronunciation scores is lower in X-G10, indicating the model gives fewer 0 scores in mispronunciation (less right-skewed) compared with other models. This would allow us to provide more positive feedback to the users. On the other hand, the $Skew_P$ for correct pronunciation scores is higher in X-G10, indicating the model gives fewer 100% scores for correct pronunciation (less left-skewed). Overall, the X-G10 with higher entropy provides the best score ranges for CaptainA, allowing us to give the user an additional score in the middle of "incorrect" and "correct" pronunciation.

The difference is not visually observable between X-G5 and X-G10. Therefore, the impact of entropy in the score distribution is illustrated in Figure 14 and Figure 15 between X-G0 and X-G10. The only difference between the two models is the entropy regularization. And we can clearly see that the score distribution for mispronunciations

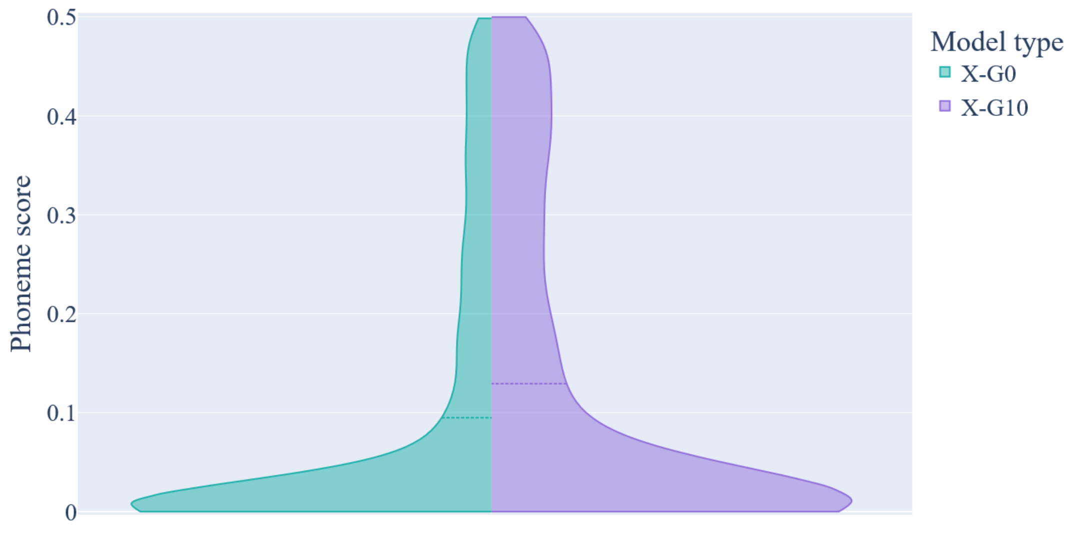


Figure 14: Phoneme score distribution for short read-aloud phonemes with a score less than 0.5.

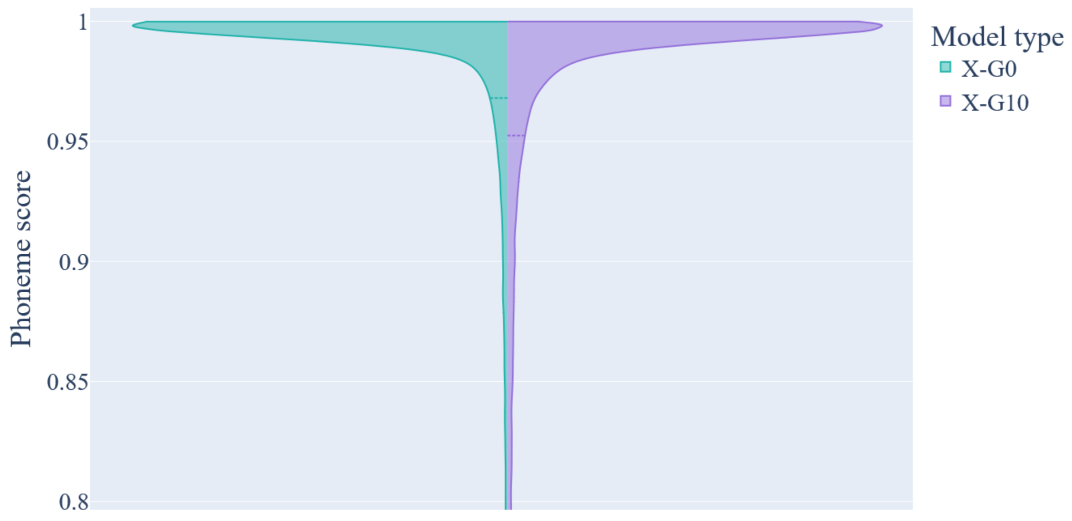


Figure 15: Phoneme score distribution for short read-aloud phonemes with a score greater than or equal to 0.8.

is more dispersed with X-G10, while X-G0 is more concentrated near 0. X-G10 has lower Skew$_M$, and is less skewed to the right, with lower kurtosis indicating a thinner tail. Both measures indicate less concentration on the tail near 0 compared with

X-G0. For the score distribution of correct pronunciations, as most of the samples are pronounced well, both models are concentrated near 1. However, we can still see that the X-G10 is less concentrated than X-G0.

The hybrid model, H-G10, has an additional symbol, "ŋ", for pronunciation of "nk" and "ng". It is our target model for CaptainA as we expect the H-G10 would have similar performance to X-G10, while also providing better pronunciation feedback for phoneme [ŋ]. Looking at Table C1, we can see that X-H10 is better in ASR than X-G10, with lower CER by 2%. However, all other metrics between the two models are similar, even for practical metrics in Table C2. This is because the test set did not have any test samples containing "nk" or "ng" (see Section 4.3). Even though we could not evaluate whether X-H10 has any advantages over X-G10 or not, the X-H10 has similar MDD performance to X-G10, so we can use X-H10 for CaptainA. We will be able to evaluate the effectiveness of the X-H10 with phoneme [ŋ] with more L2 data collected in the future.

6.2 Baseline comparison

It is also important to compare our models with the baselines of the state-of-the-art models in Finnish ASR. In Table 10, it can be observed that the performance of the 300M and the X-G0 is similar, with the exception of the 300M being slightly worse in L2 ASR, and potentially leading to higher Recall. Given that both models have the same size and hyperparameters, it appears that our careful selection of read-aloud training samples may not be as important as we initially believed. This suggests that instead of restricting ourselves to the read-aloud samples of the Finnish Parliament data, we can expand our training set to include all samples in the dataset, or other spontaneous corpora like Lahjoita puhetta. If spontaneous speech does not reduce MDD performance, using the Lahjoita puhetta corpus could help provide better evaluation for spoken Finnish, as mentioned in Section 4.

Table 10: Speech models' performance compared with baselines.

Model	CER	Recall	Precision	F_1	DAR
1BIL	**15.4%**	59.8%	**33.3%**	**42.8%**	**64.9%**
300M	22.3%	**65.0%**	26.1%	37.2%	57.5%
X-G0	20.9%	61.1%	26.7%	37.2%	59.4%
X-G10	21.2%	63.1%	29.4%	40.1%	55.3%
X-H10	19.2%	63.7%	29.3%	40.1%	58.5%

Compared with the 300M, the X-G10 and X-H10 have outstanding performance in MDD. While 300M has better Recall, the result could arguably be due to lower ASR performance. The notable improvement is in Precision while maintaining a reasonable level of Recall. With Recall higher than the baseline set by the 1BIL model, Precision is more valuable as we want to avoid FN detections (correct pronunciations being treated as mispronunciations).

The higher Precision was an unexpected but positive consequence of applying entropy. When looking at the practical metrics in Table 9, we can see that the diagnosis based solely on substitution errors has better accuracy, closely on par with the baseline from 1BIL. This is one of the reasons we only provide diagnoses for substitution errors, as we find DAR$_S$ to be more reliable. The kurtosis and skewness metrics show the main advantages of our model compared with the baseline. While our models' performance is lower than the bigger model 1BIL, we can provide a broader range of scoring to the users as feedback. From the practical point of view, recognizing user's progress is also an important factor for a CAPT system [11].

Table 11: MD metrics for the pronunciation levels from 1 to 4 in the short read-aloud test set. Pronunciation levels are rated by experts for the whole task. Metrics for level 4 are less reliable due to the lack of samples.

Model	Level	CER	Recall	Precision
1BIL	1	**26.9%**	72.6%	**38.7%**
X-G10		33.5%	**78.5%**	36.8%
1BIL	2	**20.0%**	61.5%	**32.7%**
X-G10		24.7%	**63.2%**	28.9%
1BIL	3	**11.6%**	42.4%	**27.2%**
X-G10		17.6%	**45.7%**	22.2%
1BIL	4	**6.0%**	18.8%	**20.0%**
X-G10		13.6%	**25.0%**	10.0%

Furthermore, our target user group is beginner Finnish L2 learners, and we found that our models have slightly better performance for that group. Examining the performance metrics for each pronunciation level in Table 11, we can observe that the X-G10 model outperformed the 1BIL model in Recall and performed similarly in Precision for pronunciation level 1. However, as the pronunciation level increases, the X-G10 model starts to lose ground in Precision compared to the 1BIL model. While X-G10 continues to maintain its superiority in Recall, the gap in Precision between X-G10 and 1BIL becomes more pronounced with increasing pronunciation levels. This is expected as X-G10 has a much higher CER, which contributes to the widening difference in Precision as the level increases.

Overall, considering the computing resources required for the 1 billion parameters models and the practicality of our MDD application, the X-G10 and its alternative X-H10 demonstrated significant improvements compared to our baseline. The performance of the 300M indicated that, if the baseline had the same model size, our models would outperform in all important metrics. These models would be well-suited for use in the initial public release of our CaptainA MDD app and set a new baseline for further development. As more data is collected from the app, we can develop better models in the future.

6.3 Fine-tuned performance

The performance of the fine-tuned model, X-G10-FT, is reported in Table C1 and Table C2 in Appendix C. The model demonstrated superior performance in multiple metrics, including CER, Precision, DAR, and DAR$_S$, even when compared to the 1BIL. Despite its impressive results, we ultimately decided against using it for our application due to its leniency in detecting mispronunciations, as evidenced by the significantly lower Recall rate. Compared with the baseline model 1BIL, the Recall rate is also lower, while CER is similar, suggesting the much lower Recall is not because the model has better ASR. This is likely the trade-off result between Recall and Precision as the model adapted to the lower strictness level of the transcribers. Model's Skew$_P$ is also the lowest, indicating the X-G10-FT is very generous in the scoring of correct pronunciations, with more scores concentrated around 1.

Investigating the results in the test set, we found that the fine-tuned model tended to ignore the errors caused by accents, as intended, but also failed to detect MD caused by clearly different phonemes, particularly in the pronunciation of Finnish [r]. We found multiple cases in which X-G10-FT rated the [l] pronunciation as the correct pronunciation of Finnish [r], whereas other models and even the transcribers marked the [l] as mispronunciation. There are also some cases where the fine-tuned model could not detect the differences between the front vowels and the back vowels, while other models and the transcribers did.

The initial plan was to use the X-G10-FT for beginner L2 speakers struggling with the Finnish [r], while the X-G10 or X-H10 would be used for other users. However, due to the limitations of our server, we could only use one model, and thus we could not use the less reliable X-G10-FT.

We attempted to use a lower learning rate to prevent the model from adapting to unmarked mispronunciations. Although the impact was less severe, the fine-tuned model still exhibited the same behavior. Using a lower learning rate is a naive approach to ensure the high quality of the native Finnish model is not compromised by the undetected Finnish L2 samples. A better approach would be to implement the Kullback-Leibler (KL) divergence during fine-tuning. Based on the work done by Zhuang et al. [74], the fine-tuning model is constrained by the KL divergence to maintain the similarity between the probability outputs before and after fine-tuning. Instead of minimizing the KL divergence, we can set up our fine-tuning so that any changes that cause the divergence to be larger than a predetermined threshold are rejected. However, we did not have enough resources to implement the KL divergence and tune the threshold to achieve the best result. If we cannot find or collect higher-quality fine-tuning data in the future, KL divergence will be our potential solution to improve the fine-tuning models.

7 Computer-Assisted Pronunciation Training app

Initially, our project aimed to develop a simple mobile application for demonstration purposes. However, as we progressed, we realized that it was feasible to provide a reliable MDD application with the resources available to us. A functional MDD application would also allow us to gather more L2 data and improve our models. Moreover, since there was no equivalent CAPT for the Finnish language, CaptainA could serve as a practical learning tool for L2 speakers.

Therefore, we focused more on the practicality of the MDD application and devoted a significant portion of our efforts to the development of the CaptainA mobile app. The flowchart of CaptainA is illustrated in Figure 16. It is comprised of three main components: a server responsible for MDD and other processing tasks, a modern user interface (UI) mobile app for Android and iOS devices, and instructional multimedia content that is partly integrated into the mobile app and partly hosted on an ad-free video platform.

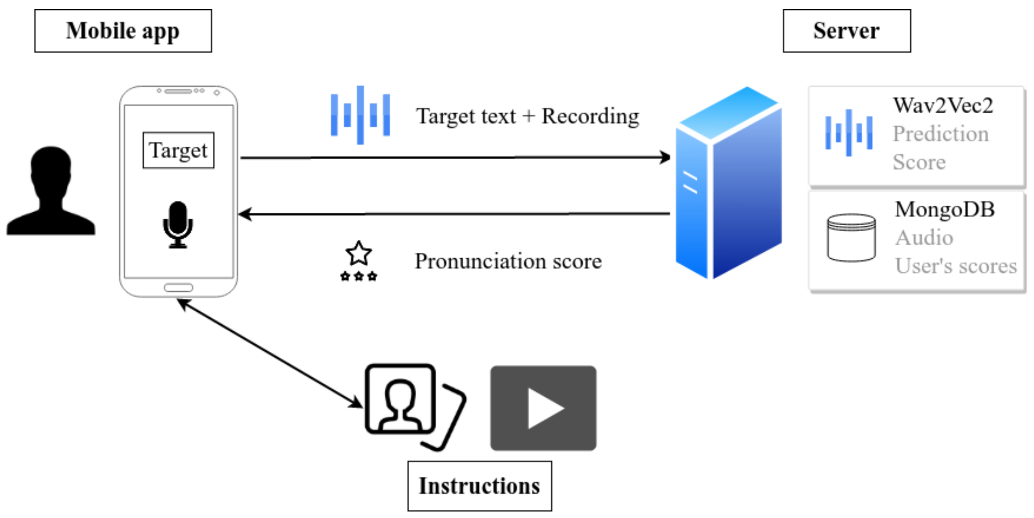


Figure 16: CaptainA's flowchart.

7.1 Unity development platform

Our CaptainA was first prototyped with Android Studio [75] and for Android mobile devices only. As the development continued further, there was a demand for support on other platforms, such as iOS or Windows. Furthermore, as CaptainA provided instructional materials in multiple formats, the integration of those multimedia made the app development became more complex. Consequently, we switched our main development tool to the Unity engine [76]. Compared with others, Unity has multiple benefits that help us swiftly develop our CaptainA.

The Unity Editor allows the CaptainA app to reach a broader audience with cross-platform development, including but not limited to Android, iOS, and Windows. Apps developed by Unity can be effortlessly converted and published to other platforms using the same interface and settings. Unity also supports development on different devices, such as smartphones, tablets, or computers. Additionally, the Editor comes with a device simulation function, allowing us to test CaptainA's interface on various handheld devices with different resolutions.

Unity's main programming language is C#, which is efficient in application development [77]. Moreover, Unity Editor is capable of drag-and-drop UI, facilitating rapid development for our mobile app. With Unity Engine, we can design a modern UI for CaptainA swiftly, without the need to devote significant time to updating our knowledge with the latest mobile development practices. As we will discuss later in this section, CaptainA contains a large number of audio, photos, and videos as instructional materials. Unity's multimedia integration tools also streamline the process of adding and managing the multimedia in our app.

In addition to the aforementioned benefits, Unity comes with an animation system, which allows us to rapidly animate our photos for mobile users. Animation is a lightweight and faster alternative to video materials, which permits users to quickly refer to our visualization guides when using CaptainA. Through animation, we can also draw the user's attention to the important part of articulation configuration, and help them understand the difference between correct and incorrect pronunciations (see Figure 19, 20).

7.2 Server

Our back end is built with Nginx, an open-source web server with its main advantage being fast response speed in handling simultaneous requests [78]. Our main programming language is Python [79], with core libraries including Levenshtein [80], Pytorch [81], and Transformers [71]. Python allows us to utilize those libraries for MDD, without the need to implement our own wav2vec 2.0 model architecture. CaptainA server is deployed using the open-source container engine Podman [82]. By using a container engine, we improve CaptainA mobility, and can quickly deploy our back end to a more powerful server if the number of users unexpectedly increases. The container engine also supports CaptainA smoothly integrating with MongoDB [83] for data collection in the later stages of this project.

The server of CaptainA handles all requests from the mobile app (see Figure 16). At the pilot stage, a request consists of a recording in waveform audio file format (WAV), and a target text indicating the user's intended pronunciation. The server's wav2vec 2.0 model processes the recording to get the rating for individual phonemes in the target text. The server then sends the results, which include transcript prediction from the wav2vec 2.0 model, phoneme ratings, and other assessments, back to the mobile app.

Additionally, the server detects potential biases caused by the nature of the Finnish Parliament corpus (Section 4. It also identifies potential problems caused by the difference between read-aloud and spontaneous Finnish. The most notable issue is the potential boundary gemination cases ("loppukahdennus" or "rajageminaatio") for spoken Finnish. Our model does not differentiate boundary gemination between written or spoken language and treats both pronunciations as correct. For example, "mene pois" (go away) is pronounced as "meneppois" [menepːois], but our model would recognize both "meneppois" and "mene pois" as correct pronunciations. Whenever the server detects such complicated case, it sends a warning to the user.

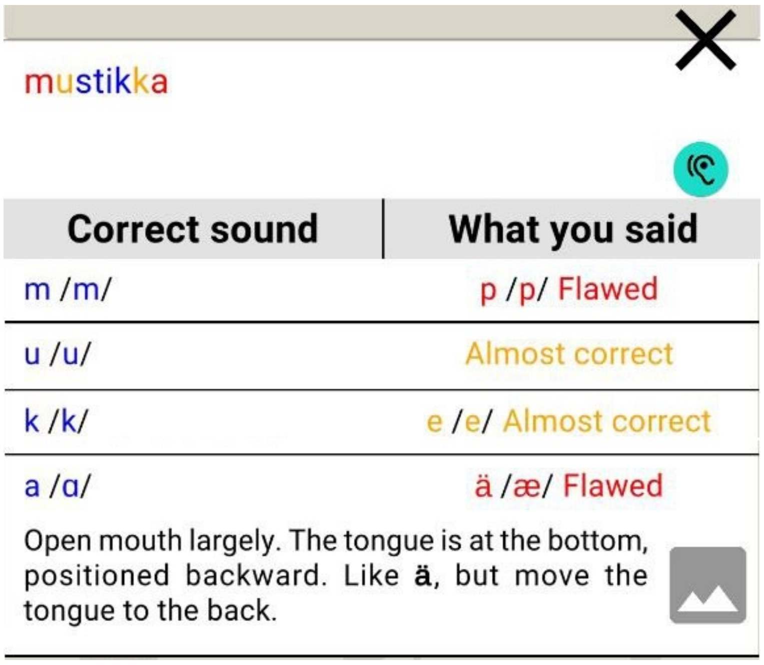
It is important to note that, at the time of this thesis work, no data is being collected. All data is deleted from the server after processing. However, the server is capable of collecting data with MongoDB software. In the later phase of the project, with the user's permission, we intend to collect valuable L2 data to improve our model performance in MDD. The model already receives the audio recording and the target transcript for the MDD task, and its output includes the user's pronunciation score, thus collecting those data would be straightforward.

Since we provide the CaptainA application as a free product, we used an idle server with limited hardware. The audio input was processed by Intel® Xeon® X5670 2.93GHz, launched in early 2010. For a model with 300 million parameters, the server time complexity is proportional to the audio length with a constant factor of approximately 1 (it takes about five seconds to process one sample with a five-second duration). CaptainA has both word and sentence pronunciation practice and has a cap of eight seconds maximum recording length. A single request sent to the server, therefore, takes around 2 to 8 seconds to process. For reference, with mobile web search, UX greatly decreases when the delay is more than seven seconds [12]. Since the latency for one user was just acceptable, using models with more than 300 million parameters was not practical. We also experimented with even smaller model sizes, but the results were not encouraging.

7.3 Mobile application

In section 5, we mentioned that Finnish is a phonetic language, which allows for easy conversion from graphemes to phonemes. We leveraged this characteristic in CaptainA's "Freestyle" mode, making it possible for users to practice any words or sentences they can type, as long as the recording time is not too long. The app automatically handles the necessary conversion between grapheme and phoneme (mainly for "ng" and "nk"). Compared to other applications, allowing users to practice any word or sentence they want is a unique feature of CaptainA. It enables users to extend the app's usability beyond the number of samples provided by us. However, it also means that users may use CaptainA in an undesirable way, such as practicing spoken Finnish while the app focuses on written Finnish. Moreover, the "Freestyle" mode lacks audio samples and translations, which need to be added manually to the app. Improving MDD for spoken language and providing automatic translations and examples are potential future updates for CaptainA.

Meanwhile, users can also practice with more tailored exercises in "Topic" mode, which often includes translations and audio samples from native speakers. The audio samples are natural human voices from fluent Finnish speakers, manually selected and edited from open-source or public-domain data sources: Finnish Parliament data [64], Mozilla Common Voice [61], and LibriVox [84]. Aalto University Language Centre also contribute practice samples from their language courses. In total, CaptainA offers approximately 180 practice samples, with around 60 samples having example audio, divided into categories such as Easy, Hard, Greetings, Pair (two similar words with contrasting pronunciations), or AudioBooks. The AudioBooks category is particularly interesting, as it introduces learners to classic Finnish literature in the public domain, along with hand-picked audio samples for practice.



mustikka

Correct sound	What you said
m /m/	p /p/ Flawed
u /u/	Almost correct
k /k/	e /e/ Almost correct
a /ɑ/	ä /æ/ Flawed

Open mouth largely. The tongue is at the bottom, positioned backward. Like **ä**, but move the tongue to the back.

Figure 17: Detail feedback on user's pronunciation of the target word "mustikka" as "pustikeä". The screenshot is cropped to reduce vertical length.

After users record their practice attempt, CaptainA sends the audio recording and the target transcript to the server. The app then waits for a maximum of 20 seconds to receive the pronunciation score information from the server. This number is subject to change depending on the server hardware and the volume of server requests. If the app receives the result, it displays the target word or sentence, with each phoneme having a different color depending on its rating (see the word "mustikka" in Figure 17).

The blue color indicates a pronunciation close to the level of a native Finnish speaker, while red marks flawed pronunciation. Yellow represents a phoneme pronunciation that falls in between. The yellow rating serves as positive feedback to encourage users in their practice and is the result of entropy regularization. Initially, we targeted four different ratings. However, measuring the accuracy of the rating scale was challenging, as we only had a binary test set with correct or incorrect ratings. From a practical standpoint, it can be difficult to distinguish four different colors on a phone screen.

CaptainA also provides an experimental pronunciation diagnosis whenever the user presses on the colored text. A small window will appear, containing detailed feedback on their pronunciation (see Figure 17). Users can see what they are supposed to say (correct sound) and compare it with what a native Finnish speaker would hear (what you said). We based the diagnosis on the actual prediction of the recording made by our model and the Levenshtein distance [48] between the target and the prediction.

As explained in Section 5.4, since we lacked suitable data to improve the diagnosis in the pilot phase of the app, we only used substitution errors to identify the pronounced phonemes. For the X-H10 model, the tested accuracy of such diagnosis was 67.7%.

The detailed feedback also includes a brief explanation of the appropriate manner of articulation to produce the accurate phoneme, along with a button link to the instructional multimedia with more information. In the future, with L2 data collected by CaptainA, we intend to implement more practical diagnoses, such as explaining common pronunciation mistakes based on the user's background, and providing personalized comparisons between the target phoneme and the user's actual pronunciation.

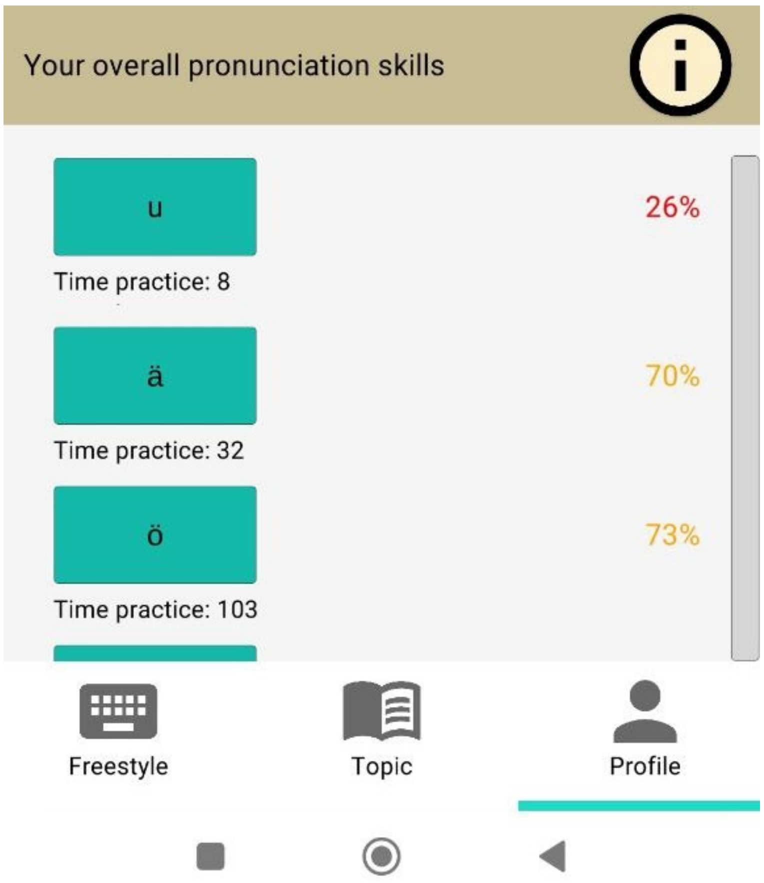


Figure 18: Learners' average score for each phoneme is saved on their phone. The score is colored based on their proficiency (red, yellow, blue). The screenshot is cut in the middle to reduce vertical length.

The user's average score for each phoneme (Figure 18) is stored locally on their device. Each phoneme is also colored so the user can easily see their strengths and weaknesses. Each entry also includes the timestamp when the user practiced. The data can be collected later, with the user's permission, to evaluate the app's effectiveness in pronunciation practice.

As the app records audio, implementing a voice activity detection (VAD) system was necessary. An efficient VAD system would help the server deliver feedback quickly and reduce its workload. However, we could not employ advanced VAD methods due to limited server capacity. On the mobile side, with limited hardware, implementing a good VAD system in Unity would have also required extended work. As our CaptainA will not charge any fees, using a third-party system was not an option. After considering all the factors, we opted for a manual VAD system, where users control the recording time by pressing and holding the record button. During internal tests, we noted that the manual VAD system was not intuitive, as people are accustomed to recording without having to hold a button. A more user-friendly VAD system would only require the user to press the record button and automatically detect when to end the recording. To address this issue, we implemented more audio and visual cues to help users become familiar with our manual VAD system. Despite its limitations, it was the best solution, given our limited server capacity and resources.

7.4 Instructional multimedia

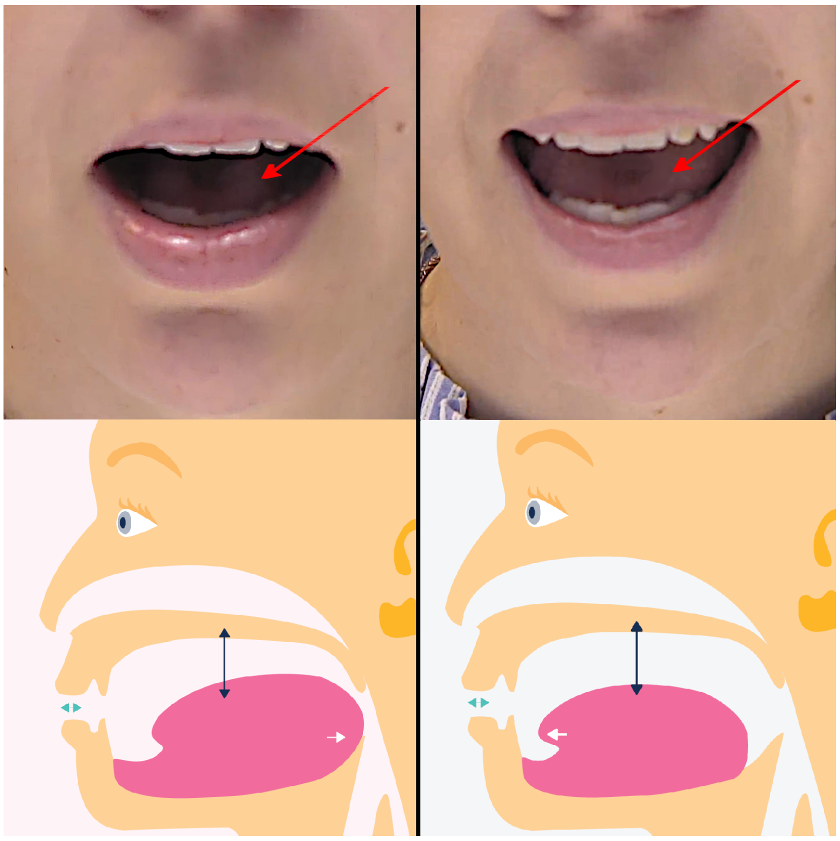


Figure 19: Phoneme assistant for A (left) and Ä (right), with a flashing red arrow pointing at the tongue to show the main difference.

A good CAPT app should not only detect pronunciation errors made by the speakers but also provide instructional material to help learners understand and self-correct their mistakes [11]. During our app development, we also found that users might not be able to correct their own mistakes without assistance. Consequently, with the help of the Aalto University Language Center and members of the Kielibuusti project, we added several photos, animations, and video links to help users understand the correct articulation configuration (the shape and position of speech organs such as lips, mouth, and tongue). Studies on the CAPT system have suggested that a combination of textual instruction and articulatory animations or videos would be beneficial to L2 learners [11, 85].

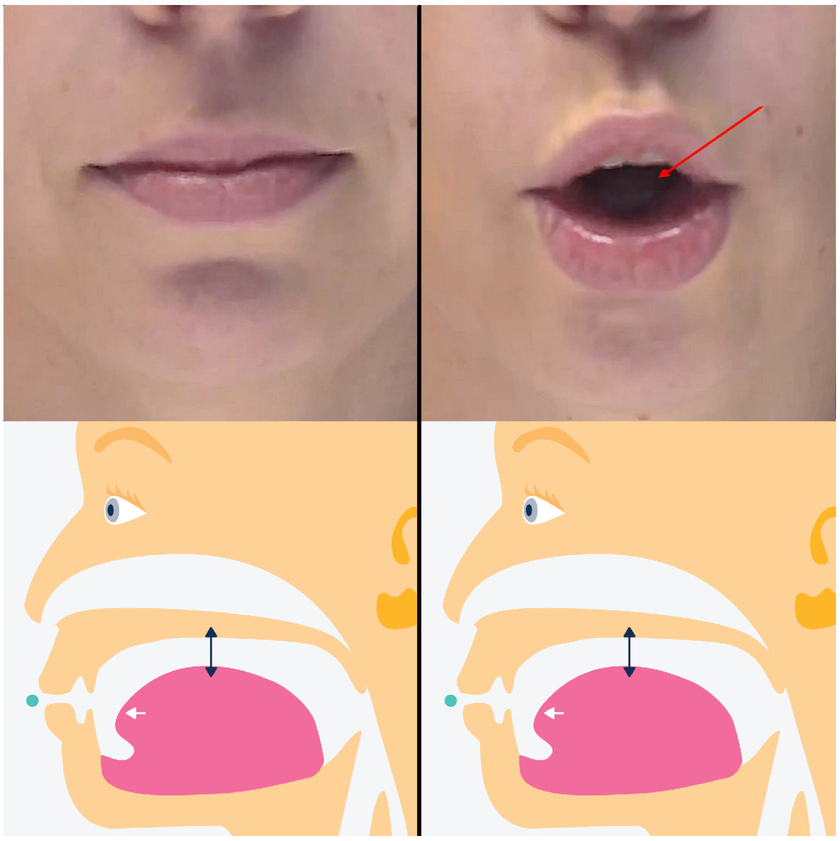


Figure 20: Phoneme assistant for Ö. The animation shows the mouth changing to the correct shape (from the left picture to the right picture). A flashing red arrow pointing at the tongue reminds the user that Ö is a front vowel.

The text instruction and photos are included inside the app. The first version of CaptainA has materials covering all Finnish phonemes, primarily focusing on vowels. The lateral view illustrations of correct articulatory configuration are created by Aino Huhtaniemi[6]. The author has permitted us to use and edit those illustrations for this project. As mentioned in subsection 7.1, we also animate some instructions to help users see the critical points in their articulation. For example, A, Ä, O, and Ö will have a flashing red arrow to focus users on the correct tongue position, while O, Ö, U, and Y will show the mouth movement to form the correct shape (see Figure 19 and 20). It should be noted that at this stage, we only show instruction for individual phonemes.

[6]https://ainohuhtaniemi.com/

With potential data to be collected from users, we could show a combination of photos to help users better understand their personal pronunciation problems.

To keep the size of CaptainA compact, instead of integrating the instruction videos directly into the app, we have uploaded them onto Panopto [86]. Panopto is supported by Aalto University and allows us to provide high-quality videos to the public free of charge or advertisements. While other video hosting platforms are also possible, they come with the risk of advertisements being added to the video without our consent, or additional costs to maintain the videos. Redirecting users to external links may negatively impact UX. However, considering our goal is to have at least one video per phoneme and the possibility of adding other open-source video materials later, it is more favorable to maintain a small app size with external videos.

8 Conclusion

Our pilot study investigated the feasibility of developing an MDD app for Finnish L2 learners, using low-resource data without detailed annotation at the phoneme level. By training with native Finnish samples and combining state-of-the-art wav2vec 2.0 models with regularization techniques, we were able to generate a more diverse rating system beyond a simple binary decision, providing users with more positive feedback. By designing our app with data collection capabilities, we could potentially gather more L2 data and improve the model's performance in the future.

Our model was trained solely with the Finnish Parliament corpus and achieved 63.7% Recall and 29.3% Precision for the MD task, and a practical diagnosis accuracy of 67.7%, with about 30% size of the baseline model. Furthermore, with entropy regularization, our model provided a wider range of scoring, allowing us to offer users extra ratings besides just correct/incorrect. The extra rating encourages users to practice and continue using the app while also reducing the social cost of incorrect MD.

Despite encountering some challenges with short and mumbled recordings, spoken Finnish, and number pronunciation, CaptainA has achieved excellent results in evaluating read-aloud written Finnish for L2 beginner speakers, thus meeting the initial target of our project. The performance is impressive, given the limited data available for MDD tasks. However, since we allow users to practice with any Finnish word they want in "Freestyle" mode, we found that, in practice, they also want to test their spoken Finnish pronunciation. Additionally, without fine-tuning with an L2 corpus, CaptainA could not recognize pronunciation from heavy accent L2 speakers or L2 speakers with speech difficulties. The diagnosis functionality is still limited to substitution errors and does not consider the speaker's background. We are confident that with more samples and metadata from L2 speakers, we can improve our model's performance and minimize those drawbacks.

The results obtained from our experiments during this project also provide valuable insights for similar research. By training our models on native language speakers' corpus and testing on L2 corpus, we demonstrated the feasibility of building an MDD app for low-resource languages with minimal L2 data. We discovered that applying entropy regularization, with β% around 5%-10%, improved not only MD but also ASR results. We found that using pre-trained data from multiple languages, such as XLS-R, can be more beneficial for MDD involving speakers with different backgrounds. In contrast, using pre-trained data from one language family, such as Uralic, may be more useful for ASR tasks involving only speakers from that language family.

The slight improvement observed when using a grapheme vocabulary, including both native Finnish letters and non-native ones, indicates that a multilingual model may be beneficial for MDD. By employing a multilingual model, we could leverage the L2 corpus from different languages, which is particularly valuable in low-resource settings. An intriguing research topic would be to explore the benefits of training an MDD model with similar languages that share a significant portion of their phonetics, such as developing an MDD app for Finnish using datasets from corpora in the Uralic language family.

In addition to exploring different datasets, there are several potential research directions to improve our models. One potential approach is to decrease model size with quantization [87], which could lead to faster app response times. Another interesting avenue is to investigate the MDD performance of models trained on a combination of Finnish corpora, such as the Lahjoita puhetta dataset [63]. While using the Lahjoita puhetta dataset could improve performance for spoken Finnish, it may have an adverse effect on read-aloud samples. Another intriguing area of research is the combination of audio and video for MDD, as changes in the shape and position of lips and tongue result in different phonemes. The visual cues are more obvious in vowel pairs such as A-Ä, E-I, E-Ö, or O-Ö and are valuable in mispronunciation diagnosis.

Due to the time limitation of the project, CaptainA only provides very limited functionality, with English translation and practice samples manually added to the app. We noted that users also want to learn about different forms of the word, and examples of word usage in a full sentence. Such requirements can be fulfilled automatically by using Verbix [88] or Wiktionary [89]. Furthermore, integrating vocabulary learning tools like Anki [90] and pronunciation practice within CaptainA would appeal to L2 learners and increase the app's usefulness for higher proficiency speakers. CaptainA's unique feature of enabling users to practice with their own set of vocabularies could further enhance app usability while reducing our workload. If CaptainA gains more public interest, we could devote more resources to implementing more practical functions in CaptainA during future updates.

With the publication of the CaptainA app for Finnish L2 learners and the promising initial results of the MDD model, we hope to garner more support for developing language learning tools for L2 students, not only in Finnish but also for other low-resource languages. The Finnish CaptainA lays a foundation for creating similar CAPT apps, such as Finland Swedish CAPT. Ultimately, CaptainA's effectiveness will be measured by its usefulness in assisting immigrants to learn Finnish and successfully integrate into Finnish society. The project's success can be evaluated in the future, upon the public release of CaptainA.

References

1. O'malley JM, O'Malley MJ, and Chamot AU. Learning strategies in second language acquisition. Cambridge university press, 1990
2. Rouhe A, Karhila R, Elg A, Toivola M, Khandelwal M, Smit P, Smolander AR, and Kurimo M. Captaina: Integrated pronunciation practice and data collection portal. Conference of the International Speech Communication Association 2018 Jan :1051–2. DOI: 10.21437/interspeech.2018-3015. Available from: https://research.aalto.fi/files/29058322/ELEC_rouhe_et_al_Captaina_interspeech.pdf
3. Becker K and Edalatishams I e. ELSA Speak - Accent Reduction. Pronunciation in Second Language Learning and Teaching Proceedings 2019; 10
4. Kacetl J and Klimova B. Use of smartphone applications in english language learning—A challenge for foreign language education. Education Sciences 2019 Jul; 9:179. DOI: 10.3390/educsci9030179. Available from: https://www.mdpi.com/2227-7102/9/3/179/pdf?version=1562847655
5. Hasan M, Hoon TB, et al. Podcast applications in language learning: A review of recent studies. English language teaching 2013; 6:128–35
6. Hu R and Xu X. Mobile experience in learning Chinese: Review and recommendations. Impacts of mobile use and experience on contemporary Society 2019 :182–92
7. Zansen Av and Huhta A. Developing automated feedback on spoken performance: Exploring the functioning of five analytic rating scales using many-facets Rasch measurement. *Digital Research Data and Human Sciences*. Jyväskylän yliopisto. 2022
8. Kallio H, Hilden R, Kurimo M, Vainio M, Karhila R, and Lindroos E. Developing a high-stake digital spoken language proficiency assessment: Results from pilot tests. *International Technology, Education and Development Conference*. 2016
9. Al-Ghezi R, Getman Y, Voskoboinik E, Singh M, and Kurimo M. Automatic Rating of Spontaneous Speech for Low-Resource Languages. *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2023 :339–45
10. Jakobson L. Holistic perspective on Feedback for adult beginners in an online course of Swedish. Apples: journal of applied language studies 2015; 9
11. Engwall O and Bälter O. Pronunciation feedback from real and virtual language teachers. Computer Assisted Language Learning 2007; 20:235–62
12. Arapakis I, Park S, and Pielot M. Impact of Response Latency on User Behaviour in Mobile Web Search. *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*. 2021 :279–83
13. Wu M, Li K, Leung WK, and Meng H. Transformer Based End-to-End Mispronunciation Detection and Diagnosis. *Interspeech*. 2021 :3954–8

14. Xu X, Kang Y, Cao S, Lin B, and Ma L. Explore wav2vec 2.0 for Mispronunciation Detection. *Interspeech*. 2021 :4428–32

15. Kürzinger L, Winkelbauer D, Li L, Watzel T, and Rigoll G. CTC-segmentation of large corpora for german end-to-end speech recognition. *International Conference on Speech and Computer*. Springer. 2020 :267–78

16. Baevski A, Zhou Y, Mohamed A, and Auli M. wav2vec 2.0: A framework for self-supervised learning of speech representations. Advances in Neural Information Processing Systems 2020; 33:12449–60

17. Liu H, Jin S, and Zhang C. Connectionist temporal classification with maximum entropy regularization. Advances in Neural Information Processing Systems 2018; 31

18. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, and Polosukhin I. Attention is all you need. Advances in neural information processing systems 2017; 30

19. Han K, Wang Y, Chen H, Chen X, Guo J, Liu Z, Tang Y, Xiao A, Xu C, Xu Y, et al. A survey on vision transformer. IEEE transactions on pattern analysis and machine intelligence 2022; 45:87–110

20. Raisi Z, Naiel MA, Younes G, Wardell S, and Zelek JS. Transformer-based text detection in the wild. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021 :3162–71

21. Kanda N, Ye G, Gaur Y, Wang X, Meng Z, Chen Z, and Yoshioka T. End-to-End Speaker-Attributed ASR with Transformer. Conference of the International Speech Communication Association 2021 Apr. DOI: 10.21437/interspeech.2021-101

22. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, et al. Language models are few-shot learners. Advances in neural information processing systems 2020; 33:1877–901

23. Chowdhery A, Narang S, Devlin J, Bosma M, Mishra G, Roberts A, Barham P, Chung HM, Sutton C, Gehrmann S, Schuh P, Shi K, Tsvyashchenko S, Maynez J, Rao A, Barnes P, Tay Y, Shazeer N, Prabhakaran V, Reif E, Du N, Hutchinson B, Pope R, Bradbury JT, Austin J, Isard M, Gur-Ari G, Yin P, Duke T, Levskaya A, Ghemawat S, Dev S, Michalewski H, Garcia X, Misra V, Robinson KG, Fedus L, Zhou D, Ippolito D, Luan D, Lim H, Zoph B, Spiridonov A, Sepassi R, Dohan D, Agrawal S, Omernick M, Dai AM, Pillai TS, Pellat M, Lewkowycz A, Moreira E, Child R, Polozov O, Lee KJ, Zhou Z, Wang X, Saeta B, Diaz MA, Firat O, Catasta M, Wei JZ, Meier-Hellstern K, Eck D, Dean J, Petrov S, and Fiedel N. PaLM: Scaling Language Modeling with Pathways. 2022. DOI: 10.48550/arxiv.2204.02311. Available from: http://arxiv.org/pdf/2204.02311

24. Jiang SWF, Yan BC, Lo TH, Chao FA, and Chen B. Towards robust mispronunciation detection and diagnosis for L2 English learners with accent-modulating methods. *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE. 2021 :1065–70

25. Shen Y, Liu Q, Fan Z, Liu J, and Wumaier A. Self-Supervised Pre-Trained Speech Representation Based End-to-End Mispronunciation Detection and Diagnosis of Mandarin. IEEE Access 2022; 10:106451–62

26. Graves A, Fernández S, Gomez F, and Schmidhuber J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the 23rd international conference on Machine learning*. 2006 :369–76

27. Hannun A. Sequence Modeling with CTC. Distill 2017. https://distill.pub/2017/ctc. DOI: 10.23915/distill.00008

28. Prabhavalkar R, Rao K, Sainath TN, Li B, Johnson L, and Jaitly N. A Comparison of sequence-to-sequence models for speech recognition. *Interspeech*. 2017 :939–43

29. Schneider S, Baevski A, Collobert R, and Auli M. wav2vec: Unsupervised Pre-Training for Speech Recognition. Conference of the International Speech Communication Association 2019 Sep. DOI: 10.21437/interspeech.2019-1873

30. Pepino L, Riera P, and Ferrer L. Emotion Recognition from Speech Using wav2vec 2.0 Embeddings. Conference of the International Speech Communication Association 2021 Apr. DOI: 10.21437/interspeech.2021-703

31. Tak H, Todisco M, Wang X, Jung Jw, Yamagishi J, and Evans N. Automatic Speaker Verification Spoofing and Deepfake Detection Using Wav2vec 2.0 and Data Augmentation. *Proc. The Speaker and Language Recognition Workshop (Odyssey 2022)*. 2022 :112–9. DOI: 10.21437/Odyssey.2022-16

32. Bannò S and Matassoni M. Proficiency assessment of L2 spoken English using wav2vec 2.0. *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2023 :1088–95

33. Hendrycks D and Gimpel K. Gaussian Error Linear Units (GELUs). 2020. arXiv: 1606.08415 [cs.LG]

34. Mohamed A, Okhonko D, and Zettlemoyer L. Transformers with convolutional context for ASR. 2020. arXiv: 1904.11660 [cs.CL]

35. Jegou H, Douze M, and Schmid C. Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence 2010; 33:117–28

36. Jang E, Gu S, and Poole B. Categorical reparameterization with Gumbel-Softmax. International Conference on Learning Representations 2016 Nov. Available from: https://openreview.net/pdf?id=rkE3y85ee

37. Park DS, Chan W, Zhang Y, Chiu CC, Zoph B, Cubuk ED, and Le QV. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *Proc. Interspeech 2019*. 2019 :2613–7. DOI: 10.21437/Interspeech.2019-2680

38. HuggingFace. Filter wav2vec2 models by Finnish language. 2023. Available from: https://huggingface.co/models?language=fi&other=wav2vec2 [Accessed on: 2023 Feb 20]

39. Wang L, Feng X, and Meng HM. Mispronunciation detection based on cross-language phonological comparisons. *2008 International Conference on Audio, Language and Image Processing*. IEEE. 2008 :307–11

40. Qian X, Meng H, and Soong F. Capturing L2 segmental mispronunciations with joint-sequence models in Computer-Aided Pronunciation Training (CAPT). *2010 7th International Symposium on Chinese Spoken Language Processing*. IEEE. 2010 :84–8

41. Lee A and Glass J. A comparison-based approach to mispronunciation detection. *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2012 :382–7

42. Zhang F, Huang C, Soong FK, Chu M, and Wang R. Automatic mispronunciation detection for Mandarin. *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2008 :5077–80

43. Li W, Chen NF, Siniscalchi SM, and Lee CH. Improving Mispronunciation Detection for Non-Native Learners with Multisource Information and LSTM-Based Deep Models. *Interspeech*. 2017 :2759–63

44. Zhao G, Sonsaat S, Silpachai A, Lucic I, Chukharev-Hudilainen E, Levis J, and Gutierrez-Osuna R. L2-ARCTIC: A non-native English speech corpus. *Interspeech*. 2018 :2783–7. DOI: 10.21437/Interspeech.2018-1110. Available from: http://dx.doi.org/10.21437/Interspeech.2018-1110

45. Shortt M, Tilak S, Kuznetcova I, Martens B, and Akinkuolie B. Gamification in mobile-assisted language learning: A systematic review of Duolingo literature from public release of 2012 to early 2020. Computer Assisted Language Learning 2021 :1–38

46. BoldVoice. BoldVoice | Pronunciation App for Non-Native English Speakers. 2023. Available from: https://www.boldvoice.com/ [Accessed on: 2023 Mar 18]

47. Witt SM and Young SJ. Phone-level pronunciation scoring and assessment for interactive language learning. Speech communication 2000; 30:95–108

48. Levenshtein VI et al. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*. Vol. 10. 8. Soviet Union. 1966 :707–10

49. Bachman LF et al. Fundamental considerations in language testing. Oxford university press, 1990

50. Eskenazi M. An overview of spoken language technology for education. Speech Communication 2009; 51:832–44

51. Likic V. The Needleman-Wunsch algorithm for sequence alignment. Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne 2008 :1–46

52. Aung KMM. Comparison of levenshtein distance algorithm and needleman-wunsch distance algorithm for string matching. PhD thesis. MERAL Portal, 2019

53. Zeyer A, Schlüter R, and Ney H. Why does CTC result in peaky behavior? 2021. arXiv: 2105.14849 [cs.LG]

54. Graves A and Graves A. Supervised sequence labelling. Springer, 2012

55. Meng H, Lo YY, Wang L, and Lau WY. Deriving salient learners' mispronunciations from cross-language phonological comparisons. *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*. IEEE. 2007 :437–42

56. Weinberger S. Speech Accent Archive. George Mason University, 2015. Available from: http://accent.gmu.edu

57. Zhang J, Zhang Z, Wang Y, Yan Z, Song Q, Huang Y, Li K, Povey D, and Wang Y. speechocean762: An Open-Source Non-native English Speech Corpus For Pronunciation Assessment. 2021. DOI: 10.48550/ARXIV.2104.01378. Available from: https://arxiv.org/abs/2104.01378

58. Babu A, Wang C, Tjandra A, Lakhotia K, Xu Q, Goyal N, Singh K, von Platen P, Saraf Y, Pino J, Baevski A, Conneau A, and Auli M. XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale. *Proc. Interspeech 2022*. 2022 :2278–82. DOI: 10.21437/Interspeech.2022-143

59. Wang C, Riviere M, Lee A, Wu A, Talnikar C, Haziza D, Williamson M, Pino J, and Dupoux E. VoxPopuli: A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021 Aug :993–1003. DOI: 10.18653/v1/2021.acl-long.80. Available from: https://aclanthology.org/2021.acl-long.80

60. Tanskanen A, Rasmus T, and Vehviläinen T. Finnish-NLP. 2023. Available from: https://huggingface.co/Finnish-NLP [Accessed on: 2023 Feb 27]

61. Mozilla. Mozilla Common Voice is an initiative to help teach machines how real people speak. 2023. Available from: https://commonvoice.mozilla.org [Accessed on: 2023 Feb 15]

62. Lennes M. Segmental features in spontaneous and read-aloud Finnish. Phonetics of Russian and Finnish 2009

63. Moisio A, Porjazovski D, Rouhe A, Getman Y, Virkkunen A, AlGhezi R, Lennes M, Grósz T, Lindén K, and Kurimo M. Lahjoita puhetta: a large-scale corpus of spoken Finnish with some benchmarks. Language Resources and Evaluation 2022 :1–33

64. Aalto University, Department of Signal Processing and Acoustics. Aalto Finnish Parliament ASR Corpus 2008-2020, version 2. text corpus. Available from: http://urn.fi/urn:nbn:fi:lb-2022052002

65. Virkkunen A, Rouhe A, Phan N, and Kurimo M. Finnish parliament ASR corpus: Analysis, benchmarks and statistics. Language Resources and Evaluation 2023 :1–26. DOI: 10.1007/s10579-023-09650-7

66. Voutilainen ERJ. The regulation of linguistic quality in the official speech-to-text reports of the Finnish parliament. CoMe: Studies on Communication and Linguistic and Cultural Mediation 2017 :61–73

67. Kuparinen K and Tapaninen T. Oma Suomi 1. Otava, 2015

68. Vazhenina D and Markov K. End-to-end noisy speech recognition using Fourier and Hilbert spectrum features. Electronics 2020; 9:1157

69. Li J et al. Recent advances in end-to-end automatic speech recognition. APSIPA Transactions on Signal and Information Processing 2022; 11

70. Kahn J, Riviere M, Zheng W, Kharitonov E, Xu Q, Mazaré PE, Karadayi J, Liptchinsky V, Collobert R, Fuegen C, et al. Libri-light: A benchmark for ASR with limited or no supervision. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020 :7669–73

71. Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, Cistac P, Rault T, Louf R, Funtowicz M, Davison J, Shleifer S, Platen P von, Ma C, Jernite Y, Plu J, Xu C, Scao TL, Gugger S, Drame M, Lhoest Q, and Rush AM. Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020 Oct :38–45. Available from: https://www.aclweb.org/anthology/2020.emnlp-demos.6

72. DeCarlo LT. On the meaning and use of kurtosis. Psychological methods 1997; 2:292

73. Groeneveld RA and Meeden G. Measuring skewness and kurtosis. Journal of the Royal Statistical Society: Series D (The Statistician) 1984; 33:391–9

74. Zhuang F, Cheng X, Luo P, Pan SJ, and He Q. Supervised representation learning: Transfer learning with deep autoencoders. *Twenty-fourth international joint conference on artificial intelligence*. 2015

75. Hagos T and Hagos T. Android studio. Learn Android Studio 3: Efficient Android App Development 2018 :5–17

76. Juliani A, Berges VP, Teng E, Cohen A, Harper J, Elion C, Goy C, Gao Y, Henry H, Mattar M, and Lange D. Unity: A General Platform for Intelligent Agents. 2020. arXiv: 1809.02627 [cs.LG]

77. Hejlsberg A, Wiltamuth S, and Golde P. C# language specification. Addison-Wesley Longman Publishing Co., Inc., 2003

78. Reese W. Nginx: the high-performance web server and reverse proxy. Linux Journal 2008; 2008:2

79. Van Rossum G and Drake Jr FL. Python tutorial. Vol. 620. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995

80. Bachmann M. Levenshtein module. 2021. Available from: https://maxbachmann.github.io/Levenshtein/levenshtein.html [Accessed on: 2023 Feb 15]

81. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 2019; 32

82. Gantikow H, Walter S, and Reich C. Rootless containers with Podman for HPC. *High Performance Computing: ISC High Performance 2020 International Workshops, Frankfurt, Germany, June 21–25, 2020, Revised Selected Papers*. Springer. 2020 :343–54

83. Banker K, Garrett D, Bakkum P, and Verch S. MongoDB in action: covers MongoDB version 3.0. Simon and Schuster, 2016

84. LibriVox. Free public domain audiobooks. 2023. Available from: https://librivox.org/ [Accessed on: 2023 Feb 15]

85. Neri A, Cucchiarini C, Strik H, and Boves L. The pedagogy-technology interface in computer assisted pronunciation training. Computer assisted language learning 2002; 15:441–67

86. Panopto. Stop Typing. Start Recording. 2023. Available from: https://www.panopto.com/ [Accessed on: 2023 Feb 12]

87. Peng Z, Budhkar A, Tuil I, Levy J, Sobhani P, Cohen R, and Nassour J. Shrinking bigfoot: Reducing wav2vec 2.0 footprint. arXiv preprint arXiv:2103.15760 2021

88. Verbix. Verbix Verb Conjugator. 2023. Available from: https://www.verbix.com/ [Accessed on: 2023 Feb 11]

89. Wiktionary. Wiktionary, the free dictionary. 2023. Available from: https://en.wiktionary.org/ [Accessed on: 2023 Feb 11]

90. Elmes D. Anki. Powerful, intelligent flash cards. Remembering things just became much easier. 2023. Available from: https://apps.ankiweb.net/ [Accessed on: 2023 Feb 11]

A Wav2vec 2.0 model hyperparameters

The following hyperparameters were used for training our 300 million parameters models, including X-G0, X-G5, X-G10, X-G20, X-P10, X-H10, U-G10, U-P10:

```
learning_rate: 5e-5
num_train_epochs: 10
per_device_train_batch_size: 128
per_device_eval_batch_size: 128
warmup_steps: 500
lr_scheduler_type: linear
save_steps: 500
eval_steps: 500
layerdrop: 0.041
activation_dropout: 0.055
mask_time_prob: 0.082
attention_dropout: 0.094
hidden_dropout: 0.047
feat_proj_dropout: 0.04
optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
seed: 1011
```

The following hyperparameters were used for fine-tuning our 300 million parameters model, X-G10-FT:

```
num_train_epochs: 5
per_device_train_batch_size: 16
per_device_eval_batch_size: 16
warmup_steps: 10
save_steps: 10
eval_steps: 10
```

B Vocabulary of each model type

The following are the vocabulary inside the “vocab.json” file of the wav2vec 2.0 models. Different model type have different vocabulary, and all related datasets are converted accordingly. [PAD] is the padding token, and [UNK] is the unknown token.

Grapheme model: "’": 1, "a": 2, "b": 3, "c": 4, "d": 5, "e": 6, "f": 7, "g": 8, "h": 9, "i": 10, "j": 11, "k": 12, "l": 13, "m": 14, "n": 15, "o": 16, "p": 17, "q": 18, "r": 19, "s": 20, "t": 21, "u": 22, "v": 23, "w": 24, "x": 25, "y": 26, "z": 27, "ä": 28, "å": 29, "ö": 30, "|": 32, "[UNK]": 31, "[PAD]": 0

Phoneme model: "a": 1, "ä": 2, "b": 3, "d": 4, "e": 5, "f": 6, "g": 7, "ŋ": 8, "h": 9, "i": 10, "j": 11, "k": 12, "l": 13, "m": 14, "n": 15, "o": 16, "ö": 17, "p": 18, "r": 19, "s": 20, "t": 21, "u": 22, "v": 23, "y": 24, "|": 25, "[UNK]": 26, "[PAD]": 0

Hybrid model: "ŋ": 1, "a": 2, "b": 3, "c": 4, "d": 5, "e": 6, "f": 7, "g": 8, "h": 9, "i": 10, "j": 11, "k": 12, "l": 13, "m": 14, "n": 15, "o": 16, "p": 17, "q": 18, "r": 19, "s": 20, "t": 21, "u": 22, "v": 23, "w": 24, "x": 25, "y": 26, "z": 27, "ä": 28, "å": 29, "ö": 30, "|": 32, "[UNK]": 31, "[PAD]": 0

C Models performance summary

Table C1: Speech models' performance in ASR and MDD on Digitala short read-aloud set.

Model	CER	Recall	Precision	F_1	DAR
1BIL	15.4%	59.8%	33.3%	**42.8%**	64.9%
300M	22.3%	65.0%	26.1%	37.2%	57.5%
X-G0	20.9%	61.1%	26.7%	37.2%	59.4%
X-G5	19.5%	63.1%	30.0%	40.6%	58.7%
X-G10	21.2%	63.1%	29.4%	40.1%	55.3%
X-G20	29.5%	66.6%	23.3%	34.5%	40.3%
X-P10	21.3%	63.2%	27.3%	38.1%	54.9%
X-H10	19.2%	63.7%	29.3%	40.1%	58.5%
U-G10	30.4%	64.3%	23.4%	34.3%	40.3%
U-P10	29.6%	**66.8%**	22.6%	33.8%	40.3%
X-G10-FT	**15.0%**	55.5%	**33.8%**	42.0%	**75.4%**

Table C2: Speech models' practical performance in diagnosis on Digitala short read-aloud set.

Model	DAR_S	SD	Kurtosis	Skew_M	Skew_P
1BIL	71.8%	0.14	3.53	1.37	-3.73
300M	63.3%	0.14	3.60	1.40	-3.50
X-G0	68.9%	0.14	3.80	1.45	-3.50
X-G5	68.5%	**0.16**	2.71	1.07	-3.42
X-G10	68.9%	**0.16**	2.54	0.99	-2.75
X-G20	54.9%	**0.16**	2.58	1.00	-2.05
X-P10	66.6%	**0.16**	2.63	1.05	-2.92
X-H10	67.7%	**0.16**	2.59	1.04	-3.16
U-G10	57.7%	0.15	3.15	1.22	-2.20
U-P10	57.1%	0.15	**2.53**	**0.98**	**-1.96**
X-G10-FT	**75.4%**	**0.16**	2.82	1.15	-3.88