Master Thesis in Statistics and Machine Learning
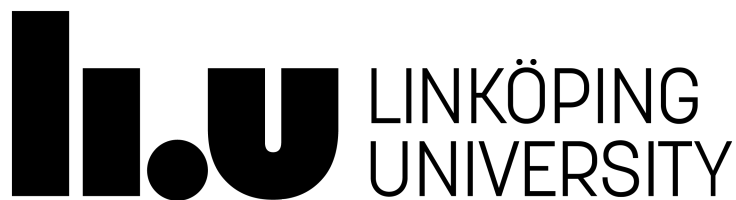
Improving Speech Recognition for Arabic language
Using Low Amounts of Labeled Data

Mohammed Bakheet

Supervisor: Filip Ekström
Examiner: Jose M. Peña
External Supervisor: Juan Verhook

LI.U LINKÖPING UNIVERSITY

Division of Statistics and Machine Learning

Department of Computer and Information Science

Linköping University

LIU-IDA/STAT-A–21/045—SE

2021

Upphovsrätt

Copyright

Abstract

The importance of Automatic Speech Recognition (ASR) Systems, whose job is to generate text from audio, is increasing as the number of applications of these systems is rapidly going up. However, when it comes to training ASR systems, the process is difficult and rather tedious, and that could be attributed to the lack of training data. ASRs require huge amounts of annotated training data containing the audio files and the corresponding accurately written transcript files. This annotated (labeled) training data is very difficult to find for most of the languages, it usually requires people to perform the annotation manually which, apart from the monetary price it costs, is error-prone. A supervised training task is impractical for this scenario.

The Arabic language is one of the languages that do not have an abundance of labeled data, which makes its ASR system's accuracy very low compared to other resource-rich languages such as English, French, or Spanish. In this research, we take advantage of unlabeled voice data by learning general data representations from unlabeled training data (only audio files) in a self-supervised task or pre-training phase. This phase is done by using wav2vec 2.0 framework which masks out input in the latent space and solves a contrastive task. The model is then fine-tuned on a few amounts of labeled data. We also exploit models that have been pre-trained on different languages, by using wav2vec 2.0, for the purpose of fine-tuning them on Arabic language by using annotated Arabic data.

We show that using wav2vec 2.0 framework for pre-training on Arabic is considerably time and resource-consuming. It took the model 21.5 days (about 3 weeks) to complete 662 epochs and get a validation accuracy of 58%. Arabic is a right-to-left (rtl) language with many diacritics that indicate how letters should be pronounced, these two features make it difficult for Arabic to fit into these models, as it requires heavy pre-processing for the transcript files. We demonstrate that we can fine-tune a cross-lingual model, that is trained on raw waveforms of speech in multiple languages, on Arabic data and get a low word error rate 36.53%. We also prove that by fine-tuning the model parameters we can increase the accuracy, thus, decrease the word error rate from 54.00% to 36.69%.

Acknowledgements

1 Introduction

Acoustic Speech Recognition (ASR) systems enable machines to recognize human speech and transcribe it into text. There are currently many problems facing ASR Systems, to mention a few: the first one is the speaker problem, and this could be the speaker emotions, gender variation, or age variation. Secondly, the vocal track length could also affect the way people pronounce words. Thirdly, the background and channel noise, this could appear as children noise in a car or a metro station, or channel noise when speaking over a phone or wireless devices in general. One of the challenging problems is the different accents within the same language (Arabic in this context) could vary depending on the speaker's country of origin [1].

In the last years, Automatic Speech Recognition systems have improved the state of the art significantly. Open-source collaborations such as Wav2Vec 2.0 [2] and Mozilla DeepSpeech [3] have brought the speech-to-text community together to allow anyone to train their models with a comparable performance of the best models 2 years ago with 100 times less data [4]. The problem of data collection is still significant and finding new ways of addressing these issues is critical for the speech-to-text models of the future.

In the initial stages, Hidden Markov Models (HMM) were used as a stochastic approach to solve the problem of speech recognition. HMM achieved good results, but it also had many limitations [5]. A few of these limitations were solved by HMM/Gaussian mixture models (HMM/GMM), in any case, numerous remained unsolved [6]. After that, a hybrid of deep neural networks (DNN) and HMM was shown to significantly improve ASR performance over the mixture of Gaussian mixture model and Hidden Markov Models (GMM-HMM) [7][8].

A problem that appears when using DNN in speech recognition is the enormous quantities of labeled data that is essential for training neural networks which, in many fields, is more difficult to find than unlabeled data. Thousands of hours of transcribed speech are required to achieve high model accuracy, this large number of hours is unavailable for most of the more than 7000 spoken languages [9]. Learning purely from labeled examples does not resemble language acquisition in humans: infants learn language by listening to adults around them - a process that requires learning good representations of speech [2].

People usually do not transcribe what they say, one must label many hours by themselves or hire someone to do this rather tedious job for them. Even though this could work in many cases, we still have the hurdle of spoken vs written languages as the same written word could have different pronunciations depending on different speakers. ASR systems have long been relying on labeled data for better performance, so, the more transcribed training data we get the higher the performance. This could mean, for languages with different accents, one must have hours of labeled training data for each accent to acquire high performance in that specific accent. A solution for this problem of labeled data scarcity is to have an unsupervised task where the model is pre-trained on huge amounts of unlabeled data (which is easier to find), and then finetune the pre-trained model on a few hours of labeled data in a supervised task, this solution is referred to as Self-supervised Learning (SSL) [10].

1.1 Background

Per capita, Sweden received the most asylum seekers globally in 2015. Beyond accepting refugees via its longstanding resettlement program, Sweden has taken in rising numbers of asylum seekers since 2012, exceeding 40,000 per year. Most of these recent arrivals have been Syrians, Afghans, and Iraqis [11]. The language barrier has been a big issue for those immigrants when they see a doctor or have a job interview. DigitalTolk is a company that provides interpretation and translation services to the public and private sectors in Sweden. Patients in emergency rooms, migrants in trauma units, or a job seeker who can not speak Swedish are ought to get the support they need for a better life.

DigitalTolk provides more than 200 languages and dialects support, it has translators and interpreters who speak all those languages. One has to book an interpreter, a person in the current setup, through DigitalTolk's booking system, then the interpreter will be assigned to the client based on an importance score. The current booking system is built on the basis of several importance algorithms, such as the interpreter's interests, experience, competence, or previous feedback from clients.

Even though, the current solution might be efficient for many scenarios, a faster and more efficient and effective way would be to build an Automatic Speech Recognition (ASR) systems by using state-of-the-art speech recognition technologies. This new ASR system interprets from Swedish to Arabic and from Arabic to Swedish in real-time, it allows non-Arabic-speaking healthcare workers to communicate with patients who are unable to speak Swedish. The ASR does the interpretation through a device that listens to a sentence in a language and then speaks that sentence in another language.

To build an ASR system, a huge amount of training and validation data is needed for the model to achieve high accuracy, and this poses a challenge for building this system as the problem of data annotation (labeled datasets) has been a major issue for ASR systems, and that could be attributed to the fact that individuals generally do not write as much as they speak, and the spoken language tends to be different than the written form of that language (different dialects). Because of this, a model that utilizes a few amounts of labeled (annotated) data, yet achieves high prediction accuracy is needed. Building an ASR with low amounts of annotated Arabic training data will subsequently lay the foundation for building other ASRs for different Arabic dialects or even other languages.

1.2 Research Questions:

This research answers the following questions:

- The problem of data collection is still significant and finding new ways of addressing these issues is critical for the speech-to-text models of the future. How can we address this problem?

- Can self-supervised learning be used to utilize unlabeled speech data for training an Arabic ASR system? If not, what other techniques could be used?

- Is it possible to achieve high accuracy using a few amounts of labeled data?

2   Theory

In the face of a wide diversification of sounds, both experimentally applied (music) and naturally occurring, human speech perception is vigorous. Significant strides are being made regularly to design a machine that mimics the human behavior of speech recognition. Over the last few years, advances in both machine learning algorithms and computer hardware have led to more efficient methods for training different speech recognition models.

2.1   How Speech is Digitally Represented

If audio is to be transcribed, going straight to characters is not feasible as characters do not map to different phonemes depending on their pronunciation and position in words. Figure 1 illustrates a waveform and its phonemes representation. A traditional ASR pipeline usually represents words as a sequence of "phonemes," For instance: hello = [HH AH L OW]. Phonemes are the perceptually distinct units of sound that distinguish words. Even though it is not clear how fundamental these phonemes are and despite their approximation, they are somehow standardized and there are different conventions for how to define them. One popular dataset on systems using phonemes is called TIMIT, that has a corpus of audio frames with an example of each of the phonemes [12], it is widely used by phone recognition research community because it is annotated at the phone level.

Once phoneme representation is acquired, it adds even more complexity to the ASR system, because now the ASR system does not associate audio features with words, it associates them with another kind of transcription. This brings us to the introduction of another component into the pipeline that tries to understand how to convert the transcriptions in phonemes into actual spelling, and this is done by using a phonemes dictionary that contains a mapping of each word and its respective phonemes representation.

To transform an acoustic signal to words, it is desired to first explore the elements of speech that are: **Segments** that are discrete units that can be identified, either physically or auditorily, in the stream of speech, **Words**, **Syllables** which are segments of a speech that consist of a vowel, and **Phonemes** that are unique concepts in phonetic to recognize words and phones. The figure below transcripts the sentence "she just had a baby" with phonemes. A segment is, on the other hand, one or multiple phonemes.
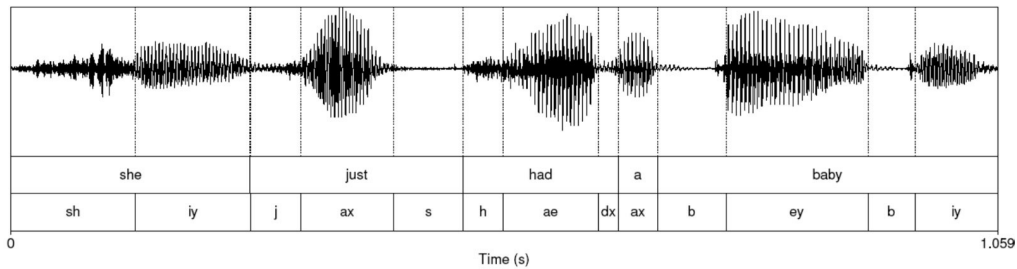
Figure 1: Waveform of someone saying: 'she just had a baby '

Audio is represented in a one-dimensional wave. Typical sample rates for speech are: 8KHz, 16KHz, and each sample is typically 8-bit or 16-bit. We can think of an audio signal as a one-dimensional vector $X = [x_1, x_2, ..., x_n]$ representing time and a vector $Y = [y_1, y_2, ..., y_n]$ representing the amplitude of the speech wave y_i at time x_i. However, this representation could change to a different form (a spectrogram) that is covered in section 2.6.1.

2.2 How ASR works

In a traditional ASR system, we would have an acoustic model whose job is to learn the relationship between a sequence of phonemes and the audio we are hearing. It accomplishes that by transforming speech features to a conditional probability distribution over phones, $p(O|W)$. For instance, the pronunciation of the letter "t" at the beginning of the word as "tea" and the end of it such as "spirit," so, what is the probability of this letter being pronounced? There are different elements involved in ASR systems apart from the acoustic model, that are:

Pronunciation dictionary transforms phones into words. Every word has a number of sounds representing it as in English dictionaries. This could be affected by accent variation. In addition to that, the way someone is speaking is also one of the factors affecting pronunciation (spontaneous speech vs. BBC radio news).

Language model represents a probability distribution over words sequence, $p(w)$. Language models solve the problem of listening without being able to transform sounds to words or meanings, it assumes audio is grammatically and semantically correct. It provides context to distinguish between words and phrases that sound similar.

Decoder aggregates all knowledge to a contract search space, in other words, the most probable word sequence. The job of the decoder is to find the sequence of words (W) that maximizes the probability of a particular sequence (w) given the audio (X).

$$\underset{W}{\operatorname{argmax}} P(W|X) \tag{1}$$

There have been many successful attempts to build acoustic models for each language [13] [14] [15]. However, the state-of-the-art is to build a language-independent acoustic model, but also to increase the accuracy of existing models. Speaker independent models are first trained from multiple speakers and then adapted to a specific speaker either before or during recognition. Analogously, language-independent modeling is a methodology that combines speech and models from multiple source languages and transforms them for recognition in a specific target language [16]. It is vital to mention that speaking the same language with different accents might affect the ability of that language's ASR system to recognize the speech.
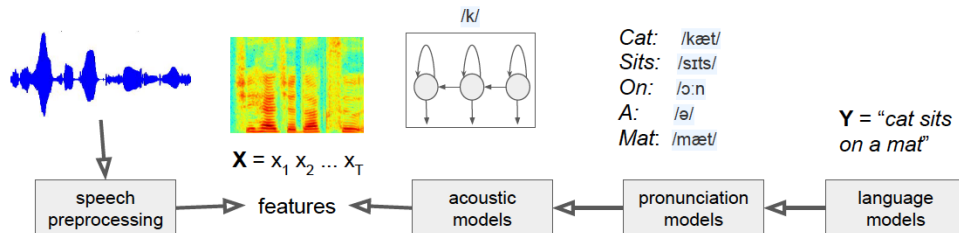
Figure 2: The classic way of building a speech recognition system (https://heartbeat.fritz.ai/)

Figure 2 illustrates the classic way of building speech recognition systems. A generative model of language is built, by producing a certain sequence of words from the language model (rightmost side), and then for each one of the words, the pronunciation model decides how this particular word is spoken by writing in the form of phonemes. The pronunciation models are then fed into an acoustic model which decides how the phoneme sounds. The acoustic model does the description through frames of audio features that are discussed next.

2.3 Speech Features Extraction

Speech signals have high variability for many reasons including gender, accent, emotion, and background noise. To reduce the variability of speech signals some feature extraction should be done. There are many feature extraction techniques used in speech recognition systems including Linear predictive analysis (LPC), Linear predictive cepstral coefficients (LPCC), Perceptual linear predictive coefficients (PLP), Mel-frequency cepstral coefficients (MFCC), Power spectral analysis (FFT), Mel scale cepstral analysis (MEL), Relative spectra filtering of log domain coefficients (RASTA), and First-order derivative (DELTA) [17].

Mel-Frequency Cepstral coefficients (MFCC) is arguably the most used feature for ASR including in deep learning (DL) systems. It is based on short-time analysis (around 20–50 ms of sound), assuming speech to be stationary during that period. MFCC shows good performances on clean conditions, but it is not robust against speech distortions such as background noise, reverberation, channel distortion, etc. One of the reasons is because the log function is overly sensitive for low energy spectra where much of speech information resides [18].

$$C(x(t)) = F^{-1}[log(F[x(t)])] \tag{2}$$

Equation 2 shows how we can compute *cepstrum* on top of which MFCC is built, where $x(t)$ is a signal in time domain (normal waveform), $F[x(t)]$ is the discrete fourier transform which generates a spectrum by moving from time domain to frequency domain [19]. Fourier Transform transforms a function of time $x(t)$ to a function of frequency $F[x(t)]$. Then we apply logarithm on the amplitude of the spectrum for more computational efficiency [20]. Finally, we apply an inverse fourier transform to the log amplitude spectrum to produce cepstrum.

2.4 Speech Classification

In ASR systems (illustrated in figure 2), structured sequence data is classified, where the label sequences are inferred from the observation sequences. An accurate ASR system is a system that performs with high accuracy in speech recognition tasks by classifying label sequences (sentences) correctly from the observation sequences (waveforms). We can model the posterior probability of the classes directly in a discriminative model, whereas, in a generative model Bayes' rule is used.

2.4.1 Generative vs. discriminative classifier

The model is to model the distribution of words in the word space. ASR systems infer the sentences' acoustic waveforms. Speech signals are sequential in nature, and that is where generative models come into view by using Bayes' rule to combine Hidden Markov Model (HMM) acoustic models and N-gram language models. On the contrary, the machine learning and natural language processing (NLP) research areas are increasingly dominated by discriminative approaches, where the class posteriors are directly modeled [21]. For instance, when a child is told to "bring that cup," the child does not actually know unless one points to the cup, discriminative models would tell the child the difference between the cup and the other objects that might exist.

Generative model classification use Bayes' rule e.g., for HMMs, and it can be split into a language model, the prior, $P(w)$ that yields a probability of any sentence, and

an acoustic model $p(O_{1:T}|\lambda^{(w)})$ which is the likelihood that a sentence w generated the observations $O_{1:T}$ with model parameters $\lambda^{(w)}$. We can, as a result, obtain the posterior by applying Bayes' rule:

$$P(w|O_{1:T}, \lambda) = \frac{p(O_{1:T}|w, \lambda^{(w)})P(w)}{\sum_w p(O_{1:T}|w, \lambda^{(w)})P(w)} \tag{3}$$

Whereas discriminative models directly model the posterior given the observation sequence:

$$P(w|O_{1:T}, \alpha) = \frac{1}{Z}\exp(\alpha^T \phi(O_{1:T}, w)) \tag{4}$$

where Z is the normalization constant to ensure a valid probability mass function over all sentences, and α the discriminative model parameters. $\phi(O_{1:T}, w)$ is called the feature function. Feature-function, shown in Figure 3, defines the relationship between the sequence of observations and the sentence label.
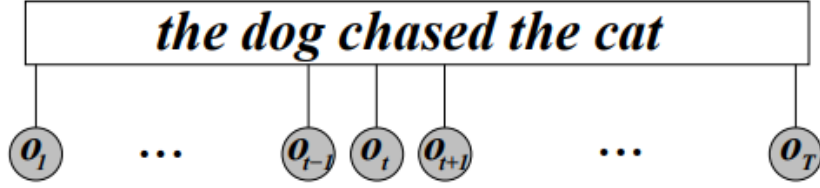
Figure 3: Graphical model for a simple discriminative model [13]

2.5 HMM-GMM in Speech Recognition

An acoustic model works as a human ear. Nearly four decades ago, the expectation maximization (EM) algorithm was introduced as a principal means for training HMMs which is a Markov Chain where the output symbols or probabilistic functions that describe them, are connected either to the states or to the transitions between states [22]. Both Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM) had been, before the era of deep learning, must-learn technologies when dealing with speech signals.

HMMs are used as acoustic models to calculate the likelihood of different classes generating observation sequences (sentences). HMM model consists of hidden variables (States) and observables, a Markov chain also contains the probability of transitioning from one state to another. The modeling part, which is done by HMM, is for transitioning between phonemes and the corresponding observable. GMM plays the role of emission probability distribution which results in mixture emissions. Figure 4 shows Markov process and the transitioning between different states.

Gaussian Mixture Models (GMMs) are, on the other hand, used to model the distribution of features for phonemes. They allow modeling features with a few possible values, which provides flexibility in speech variations. In Figure 5, it can be seen that when we try to model the phone "sh" using gaussian model, it fixes the utterance of that phoneme and does not allow for speech variations, whereas, when GMM is used for
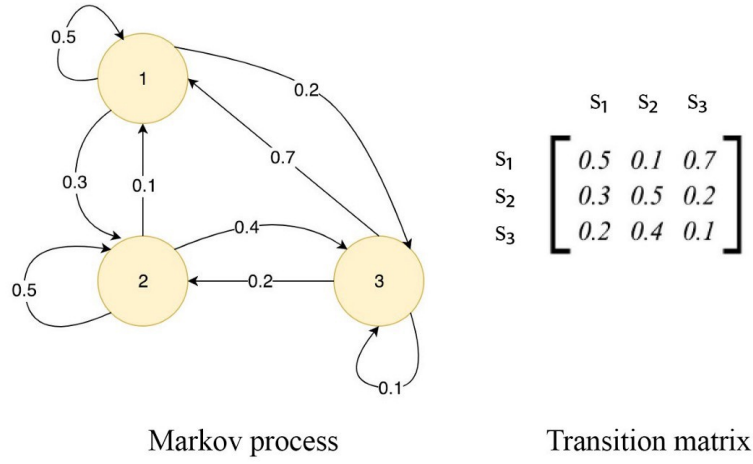
Figure 4: Markov Process and Transition Matrix

the same purpose, it allows different pronunciations for the same phone, which is more realistic for speech recognition.
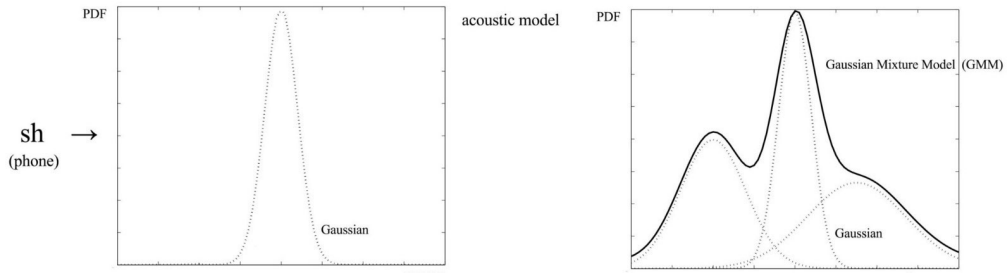
Figure 5: Gaussian vs GMM acoustic model when modeling "sh"

Pronunciation Model For obtaining a good estimate for the relationship between pronounced speech and the parts of words, statistical models require an adequate number of samples. In order to share information across words to avoid training statistical models on each word (which is not feasible), the pronunciation model is used.

$$W^* = \operatorname*{argmax}_{W} P(W|X) \tag{5}$$

By considering words as sequences of states Q

$$W^* = \operatorname*{argmax}_{W} P(X|Q,W)P(Q,W) \tag{6}$$

$$W^* \approx \operatorname*{argmax}_{W} P(X|Q) \sum_{Q} P(Q|W)P(W) \tag{7}$$

$$W^* \approx \operatorname*{argmax}_{W} \operatorname*{max}_{Q} P(X|Q)P(Q|W)P(W) \tag{8}$$

7

Where $P(Q|W)$ is the pronunciation model [23].

Language Model assigns a probability estimate to word sequences, it is referred to in equation 9, which is derived from equation 5, as $P(W)$.

$$W^* = \underset{W}{\operatorname{argmax}} P(X|W)P(W) \tag{9}$$

The maximum likelihood estimate is calculated for estimating sequences and is given by:

$$P(w_i|w_1, ..., w_{i-1}) = \frac{C(w_1, ..., w_i)}{\sum_v (C(w_1, ..., wi - 1v))} \tag{10}$$

Where $C(w_1, ..., w_i)$ is the observed count in the training data. For instance, if we want to know the probability of the word (*the*) being pronounced after the sentence (*its water is so transparent that*), this would give us:

$$P(the|\text{its water is so transparent that}) = \frac{C(\text{its water is so transparent that the})}{\sum_v C(\text{its water is so transparent that})} \tag{11}$$

We can, hence, calculate the probability of the whole sequence by using the chain rule as follows:

$$P(w_1, ..., wN) = \prod_{k=1}^{N} (w_k|w_{k-1}) \tag{12}$$

2.6   Deep learning in ASR

DNNs have proven more successful in speech recognition than Gaussian Mixture Model (GMM), and that is due to the fact that GMMs are unsuitable for modeling data that resides on nonlinear manifolds or near to them. For example, modeling the set of points that lie very close to the surface of a sphere only requires a few parameters using an appropriate model class, but it requires a very large number of diagonal Gaussians or a fairly large number of full-covariance Gaussians [24].

ASR systems rely on modern pipelines that are composed of many processing stages, including specialized input features [25], acoustic models [26], and Hidden Markov Models (HMMs) [27]. Deep learning takes the place of these processing stages and achieves higher performance than traditional methods on hard speech [3]. An end-to-end deep learning approach can replace entire pipelines of hand-engineered components with neural networks, end to end learning allows us to handle a diverse variety of speech including noisy environments, accents, and different languages [28]. When deep neural networks are used, a considerable improvement is acquired compared to that of GMM. An acoustic model can get between 10 and 20 percent relative improvement in accuracy [14], which is a huge jump in speech recognition and highly noticeable to a speaker. Achieving such high accuracy in speech recognition tasks by using neural networks is, in some sense, the first generation of deep learning in speech recognition.

The hybrid deep neural network (DNN)- hidden Markov model (HMM) has been shown to significantly improve speech recognition performance over the conventional Gaussian mixture model (GMM)-HMM [29]. This could be attributed to the fact that DNNs are capable of modeling complex correlations in speech features.

2.6.1 Convolutional Neural Networks for Speech Recognition

Convolutional Neural Networks (CNNs or ConvNet) are a class of deep neural networks that have many signal processing and computer vision applications including speech recognition [29]. Rather than using fully connected hidden layers that is similar to DNNs, CNNs present a network structure consisting of *convolutions* and *pooling* layers. The term convolution refers to combining two functions to yield a third function, it is applied in CNNs to the input data by using a kernel or filter to produce a feature map. A *kernel* refers to the sets of learnable parameters applied in convolution operation. Whereas, a *stride* is the distance between two successive kernel positions [30]. Figure 6 shows how a simple 3×3 kernel moves in a 5×5 input tensor with one stride [31], the kernel moves across the input tensor one step to the right (the number of strides) and it multiplies element-wise.
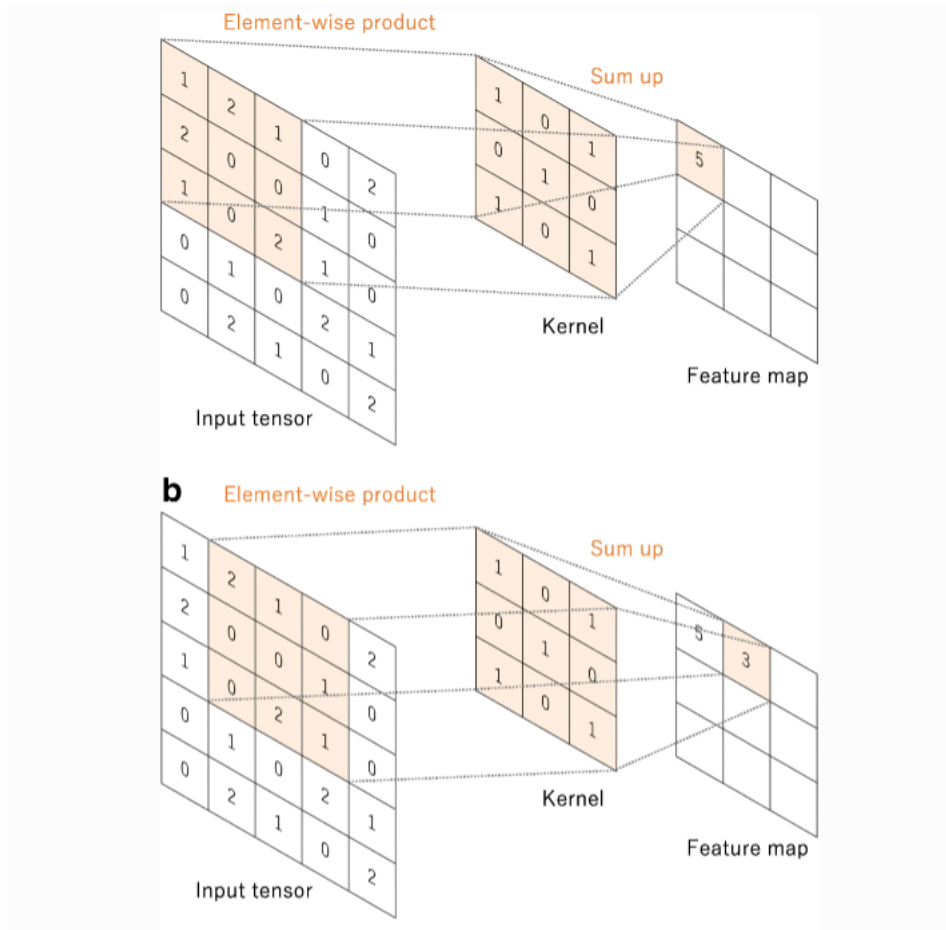

Figure 6: A 3×3 kernel with a 5×5 input tensor and one stride

When using CNNs for speech recognition, speech signals can either be transformed into spectrograms (images) or used as 1-d waves to organize speech features into feature maps that are suitable for CNNs. Assuming input feature maps are all one dimensional, each unit of one feature map in the convolution ply can be computed as:

$$q_{j,m} = \sigma\left(\sum_{i=1}^{I}\sum_{n=1}^{F} o_{i,n+m-1}w_{i,j,n} + w_{0,j}\right), \qquad (j = 1, ..., J) \qquad (13)$$

Where $o_{i,m}$ is the m-th unit of the i-th input feature map, O_i, $q_{j,m}$ is the m-th unit of the j-th feature map Q_j in the convolution ply, $w_{i,j,n}$ is the n-th element of the weight vector, $w_{i,j}$, that connects the i-th input feature map to the jth feature map of the convolution ply, F is the *filter size*, which determines the number of frequency bands in each input feature map that each unit in the convolution ply receives as input. [29]. Equation 13 can concisely be written in matrix form using the convolution operator $*$ as follows:

$$Q_j = \sigma\left(\sum_{i=1}^{I} O_i * \mathbf{w}_{i,j}\right), \qquad (j = 1, ..., J) \qquad (14)$$

Where O_i is the i-th input feature map and $\mathbf{w}_{i,j}$ represents each local weight matrix. Figure 7 shows a pair of a convolution ply (on the left side) and a pooling ply (rightmost), wherein, the mapping from either the input layer or the pooling ply to a convolution ply is based on equation 14 [29].
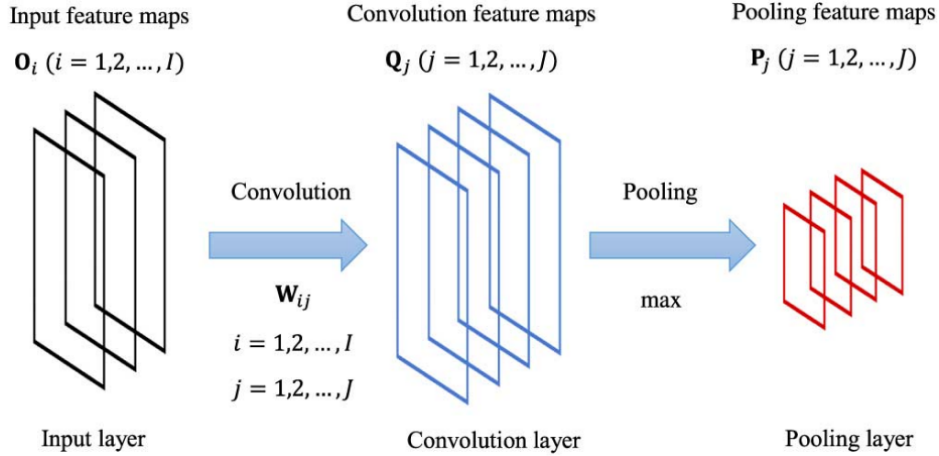
Figure 7: Two convolution plys and a pooling ply

The limitation of this feature map is seen when a different feature map is produced by a small stride (movement) in the feature position. This limitation is addressed by *downsampling*, and it is applied when important structural elements are included in a lower resolution signal which is generated from the input signal. To resolve this issue, a layer of pooling is added after the convolutional layer. Pooling selects an operation to be applied to the resulting feature maps, the result of this operation is a reduction in the size of each feature map. When using CNNs in image processing, the input is a (2-D) array of pixel values at x and y axes, whereas, in speech signals, the input could either be a spectrogram (Figure 8) or it could take a waveform. The temporal behavior of speech signals makes it a fertile area for Temporal Convolutional Network (TCN) that uses the convolutions for sequential speech data [32].
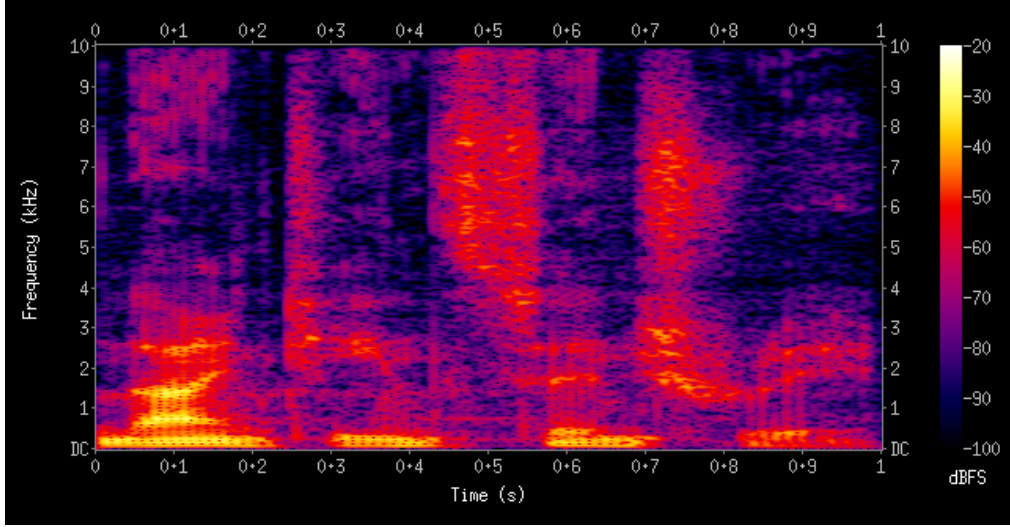
Figure 8: A visual representation of the spectrum of frequencies of speech signal varying with time (Spectrogram) that is fed to CNNs

2.6.2 Recurrent Neural Networks (RNNs) for Speech Recognition

An RNN is a class of Artificial Neural Networks (ANNs) that uses sequential data or time series data. RNNs do not assume independence between the current input and previous input, the output of RNNs depends on the memorized prior sequence. For a given input sequence $x = (x_1, ..., x_T)$, a standard recurrent neural network computes the hidden vector sequence $h = (h_1, ..., h_T)$ and outputs vector sequence $y = (y_1, ..., y_T)$ by iterating the following equations from $t = 1$ to T:

$$h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{15}$$

$$y_t = W_{hy}h_t + b_y \tag{16}$$

where the W terms denote weight matrices (e.g. W_{xh} is the input-hidden weight matrix), the b terms denote bias vectors (e.g. b_h is hidden bias vector) and $\mathcal{H}$ is the hidden layer function. $\mathcal{H}$ is usually a sigmoid function, nonetheless, the Long Short-Term Memory (LSTM) [33], which uses purpose-built memory cells to store information, proved to be better at finding and exploiting long range context than sigmoid function [34]. It is implemented by the following functions:

$$i_t = \sigma(W_{xi}X_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{17}$$
$$f_t = \sigma(W_{xf}X_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{18}$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{19}$$
$$o_t = \sigma(W_{xo}X_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{20}$$
$$h_t = o_t \tanh(c_t) \tag{21}$$

where σ is the logistic sigmoid function, and i, f, o and c are input gate, forget gate, output gate and cell activation vectors, respectively. Figure 10 illustrates the Bidirectional

11

RNNs (BRNNs) that computes the forward hidden sequence $\overrightarrow{h}$, the backward hidden sequence $\overleftarrow{h}$ and the output sequence y by iterating the backward layer from $t = T$ to 1, the forward layer from $t = 1$ to T, and then updating the output layer. If BRNN is combined with LSTM, it results in bidirectional LSTM which accesses both input directions.

$$\overrightarrow{h_t} = \mathcal{H}(W_{x\overrightarrow{h}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h_{t-1}} + b_{\overrightarrow{h}}) \tag{22}$$

$$\overleftarrow{h_t} = \mathcal{H}(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h_{t+1}} + b_{\overleftarrow{h}}) \tag{23}$$

$$h_t = W_{\overrightarrow{h}y}\overrightarrow{h_t} + W_{\overleftarrow{h}y}\overleftarrow{h_t} + b_y \tag{24}$$
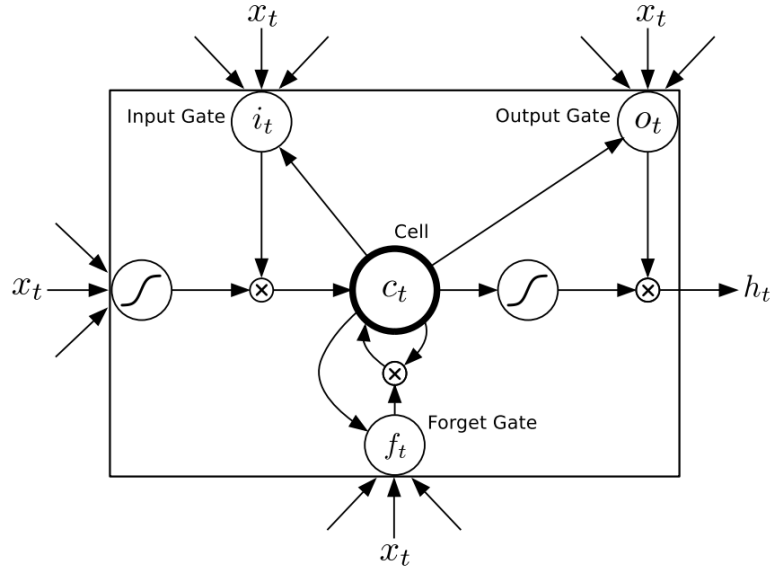

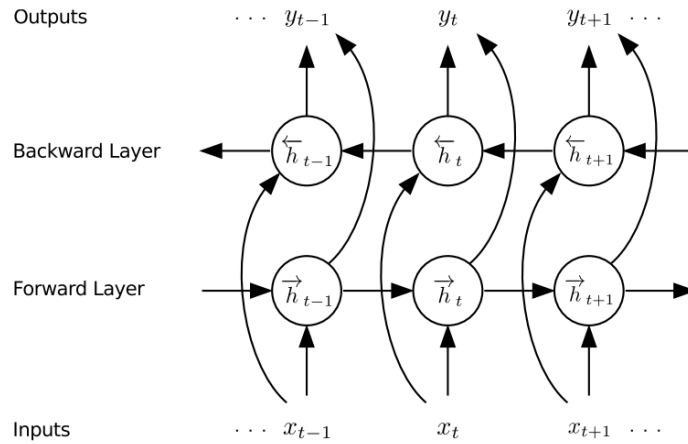Figure 9: Long Short-term Memory Cell

Figure 10: Bidirectional RNN

2.6.3 Transformers in Speech Recognition

The concept of transformers was proposed in 2017 for Natural Language Processing (NLP) [35]. It has, since then, been heavily used to solve different NLP tasks such as text classification and text analysis. Transformers achieved 28.4 Bilingual Evaluation Understudy (BLEU) on the Workshop on Statistical Machine Translation (WMT) dataset 2014, English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU [35]. In machine translation systems, BLUE is a performance evaluation technique that compares the candidate translation, which is done by the machine, and the reference translation or the human-generated translation. For speech recognition, Transformers have achieved competitive recognition accuracy compared to RNN-based counterparts within both end-to-end and hybrid frameworks [36]. Transformers are similar to RNNs in that they handle sequential input data, but they, yet differ from RNNs in that they do not require sequential data to be processed in order. They introduced an attention mechanism that learns contextual relations between words (or sub-words) in a text. The Transformer, in its most basic form, consists of an encoder and a decoder, the encoder reads the text input by mapping an input sequence of symbol representation $(x_1, ..., x_n)$ to a sequence of continuous representation $z = (z_1, ..., z_n)$ whereas, the decoder produces a prediction for the task by generating an output sequence $(y_1, ..., y_m)$ from the input z it receives from the encoder. As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. This bidirectional characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word) [37].

In speech recognition tasks, Transformers can achieve around 6% relative word error rate (WER) reduction compared to the bidirectional LSTM (BLSTM) baseline in the offline fashion, where the entire speech utterance is required as input for the encoder-decoder architecture. On the other hand, in the streaming fashion, where the entire speech is not available, Transformer-XL is comparable to latency-controlled BLSTM (LCBLSTM) with 800-millisecond latency constraint [36]. Transformer-XL was introduced in 2019 for the purpose of enabling learning dependency without the fixed-length limitation and by keeping the temporal coherence intact. It does that by using a segment-level recurrence mechanism and a novel positional encoding scheme [38].

2.7 Self-supervised Learning (SSL)

Neural networks depend on enormous quantities of labeled data to learn, and this scenario doesn't resemble the way humans learn. Self-supervised learning has arisen as a machine learning worldview to solve the problem of labeled data scarcity. It enables learning general data representations from unlabeled examples in a supervised learning task and fine-tunes the model on labeled data [2] by adding a predictor to the model that takes in the representations learned by SSL. SSL provides, when pre-training on unlabeled data, effective contextual audio representation. Additionally, by not relying on labels, SSL can avoid many problems related to labels or audio files corruption, and thus, increase model robustness. Moreover, self-supervision greatly benefits out-of-distribution detection on difficult, near-distribution outliers, to an extent that it exceeds the performance of fully supervised methods [39]. This has been proven successful for natural language processing [40]–[42] and is an active area of research in computer vision [2].

The elemental idea for SSL is to form some auxiliary pre-text task from the input data and then feed it to the model such that while solving the auxiliary task the model learns the underlying structure of the data (object structure in an image). The model is trained to distinguish a sample z_{i+k} that is k steps ahead of distractor samples $\hat{z}$ drawn from a distribution p_n, it performs this distinction by minimizing the contrastive loss

(covered in the next section) for steps $k = 1, ..., K$.

2.7.1 Contrastive Learning

In its simplest form, contrastive learning means for any positive pairs (different views of the same data) of input (x_1, x_2), there is a pair of output functions $(f(x_1), f(x_2))$, a neural network function in this context, that are similar to each other. And for a negative input x_3 both $f(x_1)$ and $f(x_2)$ are dissimilar to $f(x_3)$, Figure 11.
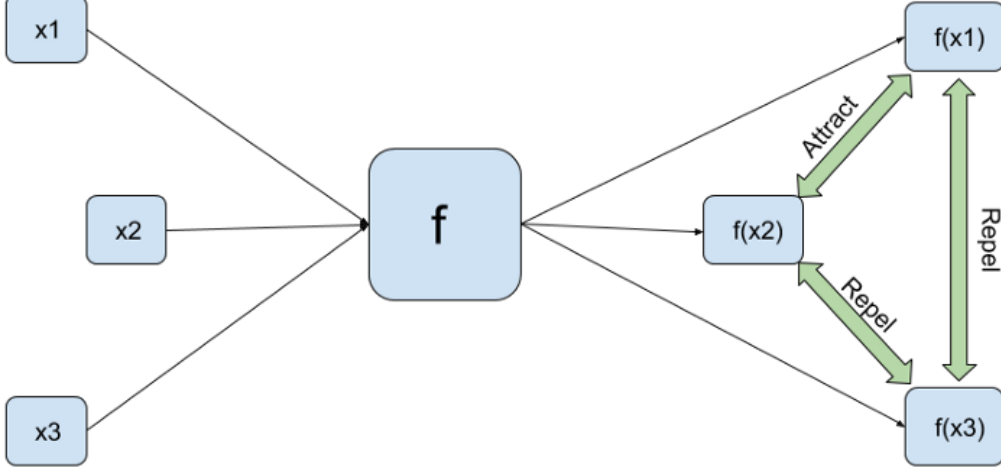
Figure 11: Contrastive Learning

In a speech signal context, for an output c_t, in order for the model to know the true quantized latent speech representation q_t in a set of $K + 1$ quantized candidate representations $\tilde{q} \in Q_t$ which includes q_t and K distractors. These distractors are masked uniformly sampled time steps of the same utterance. The loss is calculated as follows:

$$\mathcal{L} = -log \frac{\exp(sim(c_t, q_t)/k)}{\sum_{\tilde{q} \sim Q_t} \exp(sim(c_t, \tilde{q})/k)} \tag{25}$$

where

$$sim(c_t, q_t) = \frac{c_t.q_t}{||c_t|| \times ||q_t||} = \frac{\sum_{i=1}^{n} c_{ti}.q_{ti}}{\sqrt{\sum_{i=1}^{n} c_{ti}^2} \times \sqrt{\sum_{i=1}^{n} q_{ti}^2}} \tag{26}$$

$sim(c_t, q_t)$ and $sim(c_t, \tilde{q}_t)$ are the cosine similarity between context representations and quantized latent speech representations [43][44]. Low loss indicated high similarity between c_t and q_t, whereas high loss is a low similarity between them.

2.7.2 SSL in Speech Recognition

A speech file consists of raw audio that could be represented as an audio wave. In the framework used in this research (*wav2vec 2.0*) [2], a multi-layer Convolutional feature encoder $f : \mathcal{X} \to \mathcal{Z}$ is used to encode raw speech audio $\mathcal{X}$ and output latent speech representations $z_1, ..., z_T$ for T time-steps [45]. The output of the feature encoder is

discretized to q_t and, thereafter, fed to a Transformer network (discussed in section 2.6.3) $g : \mathcal{Z} \to \mathcal{C}$ to build representations $c_1, ..., c_T$ by reading the entire sequence of words at once (bidirectionally) [42]. The model is trained via a contrastive task discussed in section 2.7.1, defined over a quantization of the latent representation, where the true latent is to be distinguished from distractors [2].
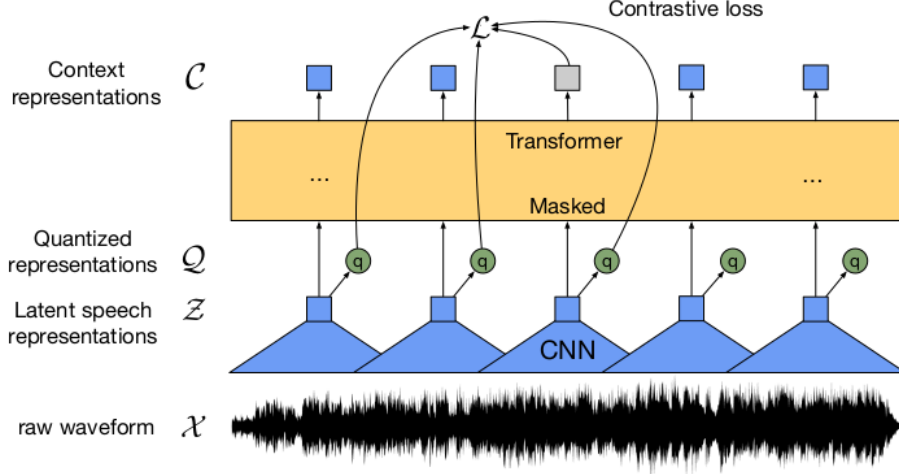
Figure 12: The framework which jointly learns contextualized speech representations and an inventory of discretized speech units. [1]

Figure 12 illustrates how the model learns contextualized speech representation. The model starts with raw audio (a 1-d waveform) that is used to feed a multi-layer feature encoder that learns the representations ($\mathcal{Z}$). These representations are vectors representing different wave segments and are, then, transformed into quantized representations ($\mathcal{Q}$) that are extracted from multiple codebooks using product quantization [46]. These codebooks (illustrated in figure 13) are vectors in a d-dimensional space containing vectors of codewords [47]. Given multiple codebooks G with V entries $e \in \mathbb{R}^{v \times d/G}$ one entry is chosen from each codebook, the resulted vectors $e_1, ..., e_G$ are concatenated and a linear transformation is applied $\mathbb{R}^d \to \mathbb{R}^f$ to obtain $q \in \mathbb{R}^f$ [2]. The extraction mechanism depends on the shortest distance between the z vector and q vectors. A Transformer is used to learn contextualized representation by masking out some of the vectors ($\mathcal{Q}$) and measuring the distance between the context ($\mathcal{C}$) and the quantized ($\mathcal{Q}$) representations of those vectors by calculating a contrastive loss.

Figure 13: Codebook generation using Gumbel-Softmax quantization

Figure 13 shows how Gumbel-Softmax which is a differentiable approximation of the

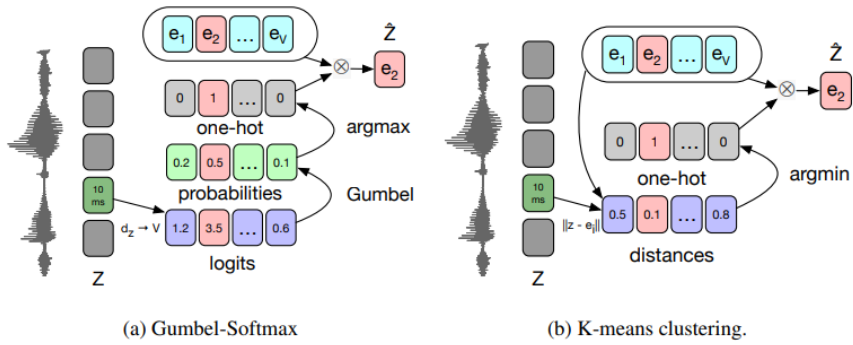argmax is used for calculating logits representing the codebook vectors. Gumbel-Softmax enables selecting discrete codewords in a differentiable way. Given a latent representation z, a linear layer is applied, followed by a ReLU [48] and another linear which outputs $l \in \mathbb{R}^V$ logits for the Gumbel-Softmax. The largest index in l is picked during inference. And the output probabilities for choosing the j-th variable during training are:

$$p_j = \frac{\exp(l_j + v_j)/\mathcal{T}}{\sum_{k=1}^{V} \exp(l_k + v_k)/\mathcal{T}} \tag{27}$$

Where $v = -\log(-\log(u))$ and u are uniform samples from $\mathcal{U}(0,1)$ [10]

2.7.3 Transfer Learning (TL)

Transfer learning is to use a model, that was used for a specific task, for a new related task as a starting point. It stores the knowledge it gained while solving the initial task, and then uses that knowledge for a different but related task. In speech recognition, we can refer to the initial task as "source language" and the new task as "target language," thus, we can initialize the network for the target language with a pre-trained model from the source language. TL has proven to be able to improve the performance and considerably decrease WER [49].

2.7.4 Pre-trained Models

2.7.4.1 WAV2VEC Pre-training

wav2vec is a model for unsupervised pre-training tasks in speech recognition developed by Facebook AI. It is a multi-layer convolutional neural network (CNN) that receives raw audio as input and outputs general representations for the raw audio, which can subsequently be used as input to ASR systems. WAV2VEC reduces WER of a strong character-based log-mel filterbank baseline [50] by up to 36 % when only a few hours of transcribed data is available [51]. Figure 14 shows the architecture of WAV2Vec that takes the raw audio as input to a fully connected CNN to solve the next time prediction task.

The parameterized encoder network $f : \mathcal{X} \mapsto \mathcal{Z}$ is applied on the raw audio samples $x_i \in \mathcal{X}$. With kernel sizes of (10, 8, 4, 4, 4) and strides of (5, 4, 2, 2, 2), The output of the encoder is a low-frequency feature representation $z_i \in \mathcal{Z}$ which encodes about 30 milliseconds of 16 kHz of audio and the striding results in representations z_i every 10ms [51].

The output of the encoder network is then fed into a context network $g : \mathcal{Z} \mapsto \mathcal{C}$ which can be seen in figure 12, and that has nine layers with kernel size three, stride one, and 210 ms receptive field. The context network outputs a single contextualized tensor $c_i = g(z_i...z_{i-v})$ from the multiple latent representations $z_i...z_{i-v}$ obtained from the encoder network. This context network follows the Transformer architecture [2]. An output layer is added on top of the transformer network to fine-tune the model on labeled data and make predictions.

2.7.4.2 Cross-lingual Speech Representation (XLSR)

As opposed to pre-training speech models on one language (monolingual pre-training), it is also feasible to make use of data from other languages for performance
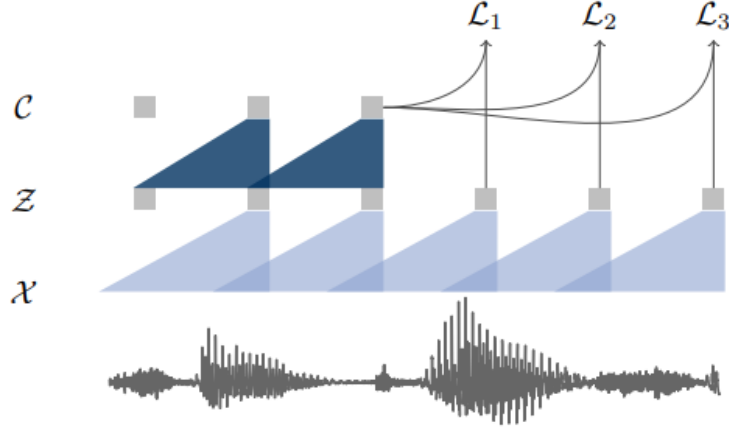
Figure 14: WAV2VEC Fully Convolutional Architecture [35]

improvement. This has worked well for natural language processing, by achieving state-of-the-art results on cross-lingual classification, unsupervised and supervised machine translation [52]. Learning cross-lingual speech representation in pre-training is not only possible, but it also outperforms monolingual models in many experiments. XLSR showed a relative phoneme error rate reduction of 72% compared to the best known results, and it also improved word error rate by 16% On BABEL corpus compared to a comparable system [53]. This pre-trained XLSR model is finetuned on individual languages which results in a different model for each language. This is particularly useful for languages with low resources as it uses other languages' resources plus the low amount of data from the target language for pre-training, then fine-tune it on the target language.
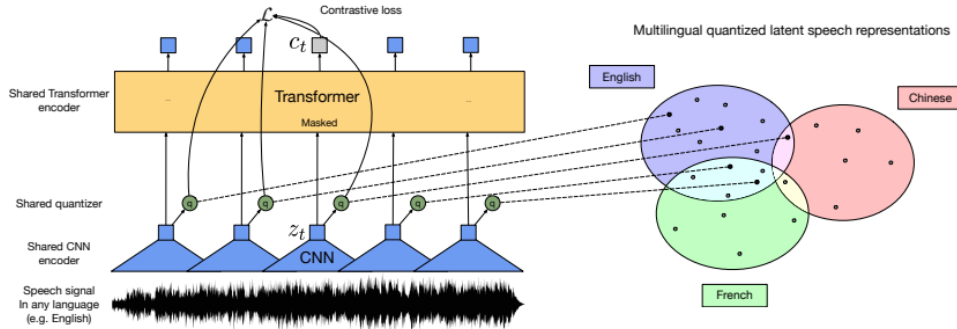
Figure 15: Cross-lingual Speech Representation (XLSR) Approach [36]

Figure 15 shows the approach of XLSR which is similar to that of WAV2VEC, and that is because of the fact that XLSR is built on top of WAV2VEC framework. Nevertheless, in XLSR architecture, it could be seen that the speech signals of different languages are fed to the shared encoder, which then produces the latent representations $\mathcal{Z}_{\sqcup}$. The latent representations for all languages are then quantized using a shared quantizer, these quantized representations $q_1...q_n$ are then fed to a shared transformer network (encoder) by masking some of the quantized vectors out. The contrastive loss is calculated after that in the same way it is calculated in WAV2VEC, except, in XLSR it is calculated for the output of the shared encoder. The representations produced by the transformer

network that are learnt with SSL are then used as input to an additional function that can be trained with regular supervised learning in the fine-tuning phase.

2.8   Connectionist Temporal Classification (CTC)

CTC is an algorithm used to solve the issue of the mismatch between the true written sentence and the predicted sentence, it uses the history of the target character without assuming conditional independence between characters. In the bottom of Figure 16 we can see how CTC classifies a speech signal, the shaded lines are the output activations, and they correspond to the probability of phonemes matches at particular times. The CTC network predicts only the sequence of phonemes (typically as a series of spikes, separated by 'blanks,' or null predictions), while the framewise network attempts to align them with the manual segmentation (vertical lines) [54].
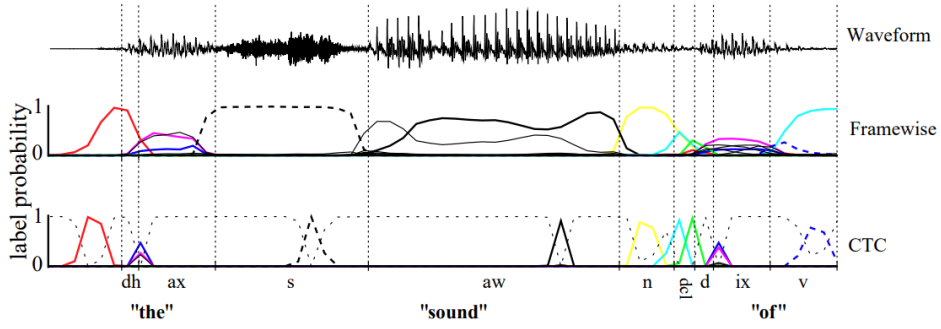
Figure 16: Connectionist Temporal Classification (CTC)

$$p(Y|X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X) \tag{28}$$

For a given input sequence $X = [x_1, ..., x_T]$ and a corresponding output sequence $Y = [y_1, ..., y_U]$, CTC assigns probabilities for each Y given an X. More specifically, it marginalizes over the set of valid alignments and computes the probability for a single alignment step-by-step. CTC does not require alignment between input and output sequences, nonetheless, it sums over the probability of all possible alignments to get the probability of output given an input.

Loss Function   CTC loss function is used when fine-tuning the model for evaluating the quality of the self-supervised pre-training phase, the aim is to maximize the probability of the correct alignment between input x_i and output y_i or to compute the conditional probability $p(Y|X)$. Then to infer a likely Y given X as follows:

$$Y^* = \underset{Y}{\mathrm{argmax}}\, P(Y|X) \tag{29}$$

CTC assigns probabilities for each Y given an X according to the alignments between inputs and outputs. Since alignment between input and output is not required by CTC, this makes it suitable for speech input which can have stretches or silence that do not correspond to output. CTC introduces a blank token ϵ to the set of allowed output, this ϵ is a placeholder and removed from the output. Figure 17 shows how ϵ is introduced in the word hello, where repeating characters are merged first, then any ϵ is removed and

the remaining characters are the output. In figure 18 it can be seen that for an input of audio signal, the input could be fed into RNN or a CNN where the network gives $p_t(a|X)$ a distribution over the outputs $\{h, e, l, o, \epsilon\}$ for each input step, the probability of different sequences is then calculated [55]. The alignment of the most likely output can be calculated by getting the highest probable alignment as follows:

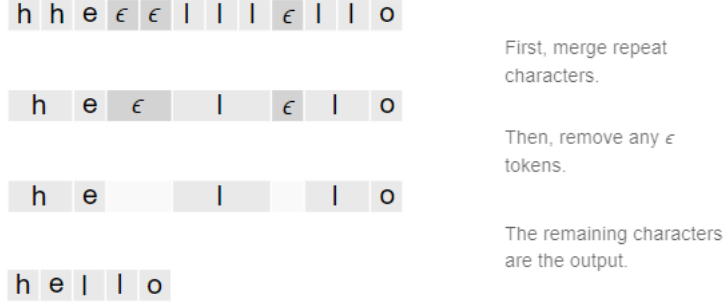$$\operatorname*{argmax}_{A} \prod_{t=1}^{T} p_t(a_t|X) \tag{30}$$

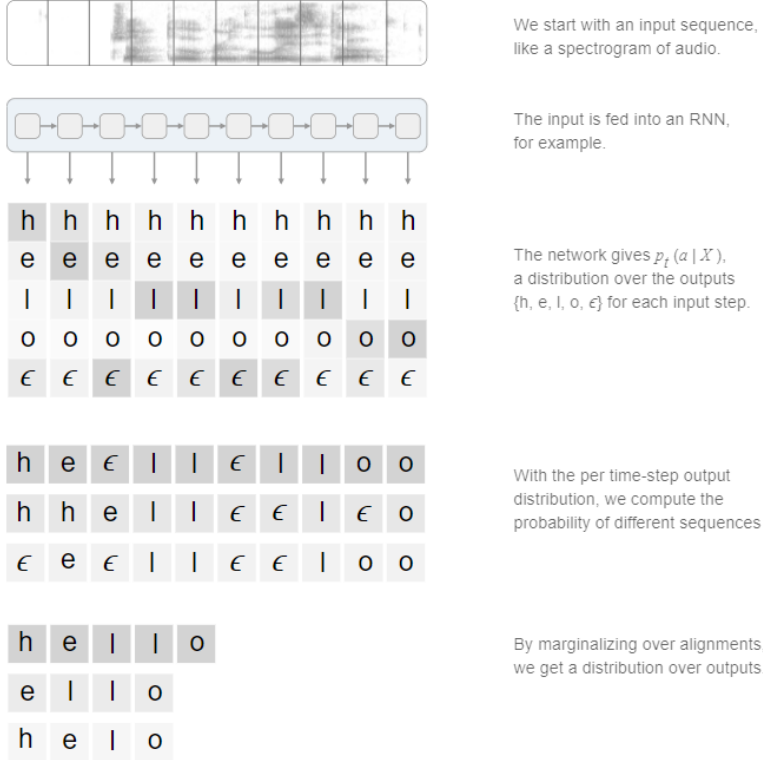Figure 17: ϵ Introduction into the word hello

Figure 18: How CTC assigns probabilities

2.9 ASR Models Evaluation

The error rate of the predicted transcript is calculated as a measurement of how accurate an ASR is. The purpose of this process is to compare the performance of

different models and techniques, but it can also go as deep as calculating the performance of these models with different hyperparameters settings. When evaluating an ASR we compare the output of the ASR system with the literal transcription of the input audio. The performance of ASRs depends on several factors including audio recording condition (e.g. background noise or the recording channel noise), spoken language variability (e.g. spontaneous speech, formal speech, or different accents or dialects), or the speaker variability (e.g. men and women, a speaker in different conditions such as illness, different emotions, and tiredness).

There are two key areas related to ASR errors, the first one is the reference-recognised alignment which consists of finding the best word alignment between the reference and the automatic transcription and the second one is the evaluation metrics measuring the performance of the ASR systems [56].

2.9.1 Reference-Recognised Word Sequences Alignment (RRWSA)

RRWSA refers to comparing the referenced and recognised words for the purpose of finding three types of errors. First, substitution error: where a word in the reference word sequence is predicted as a different word. Second, Deletion error: where a word could entirely be deleted in the predicted transcript. The third error type is the Insertion error, and in this error, a new word appears in the predicted transcript. This word sequence alignment is normally calculated using Viterbi Edit Distance [57], in which the weighted error score for the word sequence is minimized. Two distances are calculated (shown in 31 and 32), the first distance $(d_\phi^v(x^T, y^V))$ is the most likely transduction between the two strings that is calculated by taking the logarithm of the probability of the most likely edit sequence for the string pair $\langle x^T, y^V \rangle$ and it is calculated recursively [57], whereas $d_\phi^s(x^T, y^V)$ is called the stochastic edit distance, and it aggregates all transductions between the two strings by calculating the logarithm of the probability of the string pair $\langle x^T, y^V \rangle$ according to the transducer ϕ.

$$d_\phi^v(x^T, y^V) = -\log \max_{\{z^n : v(z^n) = \langle x^T, y^V \rangle\}} \{p(z^n | \phi)\} \tag{31}$$

$$d_\phi^s(x^T, y^V) = -\log\ p(x^T, y^V | \phi) \tag{32}$$

The **Viterbi edit distance** calculates the probability of the most likely edit sequence for a string pair (x^T, y^V)

2.9.2 Word Error Rate (WER) Evaluation Metric

Word Error Rate (WER) is the most popular metric for ASR evaluation, it measures the percentage of incorrect words (Substitutions (S), Insertions (I), Deletions (D)) regarding the total number of words processed [56].

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{H + S + D} \tag{33}$$

where S is the total number of words that were substituted, in the predicted transcript, with other words, D is the total number of words that were deleted in the predicted

transcript but they are in the input transcript, I is the total number of newly inserted words into the predicted written words, N is the total number of input words, and H is the total number of correct words.

3 Data

The training data for the Language Model (LM) is limited, and many require Natural Language Processing (NLP) tools such as morph analyzer, diacritizer, and text normalizers to be developed [58]. The data in this thesis is downloaded from Mozilla Common Voice Corpus (6.1), the size of the dataset is 2 GB with 49 validated hours by the community (the community validates by either matching the voice with the written text or speaking the text itself), 77 hours in total, 672 number of voices, and mp3 audio format.

3.1 How Mozilla Common Voice Community validates

Mozilla depends on contributors who record their voice clips by reading a piece of text and record their voice, this voice is then queued for validation. The clip is valid if another user listens to it and marks it as valid, however, to have the clip in the Common Voice dataset, it must be validated by at least two users. If a clip gets a negative vote, it will be returned to the queue, and if it gets another negative vote, then it will be moved to the clip graveyard where all invalid clips exist. [4]

3.2 Data Preprocessing

3.2.1 Preprocessing Audio Files

Since the dataset was initially mp3 audio files, all the files were converted to wav files, the framework considers the following conditions:

- The sample rate is to be changed to 16000
- The Pulse-code modulation (PMC) is to also be changed to 16 bit
- The silence should be removed from all audio files
- Each audio file should contain only one person speaking

3.2.2 Preprocessing Transcripts

The framework requires the transcript to be in the following format:

1. One sample per line
   - The Arabic language is written from right to left (rtl language), hence, the sentences were reversed to meet the criteria for the transcript file.
2. Upper case
   - The Arabic language does not have an upper and lower case as English, therefore, this condition is ignored.
3. All numbers should be transformed into verbal form.
   - All numbers found in the dataset were transformed to verbal form.
4. All special characters (eg. punctuation) should be removed. The final text should contain words only.
   - All punctuation marks have been removed from the sentences according to what is required by the framework.

5. Words in a sentence must be separated by whitespace character

   - All words in sentences were separated by whitespaces.

3.2.3 Diacretics

Diacritical marks play a crucial role in meeting the criteria of usability of typographic text, such as homogeneity, clarity, and legibility [59]. Diacritics are placed above, below, or through a letter, and they can change the semantic of a word completely, see Figure 19.

Diacritical Marks التَّشْكِيْل				
Tanween with *Shaddah*	*Tanween* تَنْوِيْن	Short vowels with *Shaddah* شَدّة	Short vowels	
⸝	⸝	⸝	´	*fatHah* فَتْحَة
⸜	⸛	⸜	⸗	*kasrah* كَسْرَة
⸞	⸞	⸞	⸟	*DHammah* ضَمَّة
		⸠	˚	*sukuun* سُكُون

© Ibnulyemen • 2018

Figure 19: Diacritical marks in Arabic (https://blogs.transparent.com/arabic/basic-arabic-diacritical-marks/)

All diacritics were removed from the sentences as they posed a problem when training the model with them, the model read diacritics as individual letters which affected the accuracy since it was required to remove all punctuation marks from the sentences because the model learns the meaning from the context and does not depend on individual words.

4   Methodology

Self-supervised learning is used in this research with its two phases of pre-training and fine-tuning, where transfer learning is the main technique. The implementation is done on a cloud-hosted server with Tesla K80 Nvidia GPU, using a wrapper version of wav2vec 2.0 framework [2] that is used for the purpose of training and testing the model. After the data preprocessing phase of Mozilla Common Voice Arabic dataset (6.1), two approaches were carried out. The first one was to pre-train a monolingual model on the training dataset and fine-tune it on the validation dataset, this model is then tested on the test dataset. The second approach was to use a pre-trained XLSR model and fine-tune it on training and validation datasets, then the fine-tuned model is evaluated on the test dataset. The first approach was inefficient because of the time and resources it took for completing the pre-training phase, the model used in this approach had a long pre-training time, it continued for more than 21 days and the pre-training was ongoing until it was stopped with 58% accuracy. It was then decided to choose a pre-trained model that could be fine-tuned on Arabic language (the second approach), the chosen model for this purpose is a pre-trained Wav2Vec2-XLSR-53 that is pre-trained on 53 languages including Arabic.

4.1   Data Preparation

The wrapper used for model training was used for the English language, and it requires the transcript file to include the path to the wav files and the transcript corresponding to the was file in one line. Since English is a left-to-right (ltr) language and Arabic is a right-to-left (rtl) language, the Arabic sentences are reversed to fit in the wrapper requirements. Figure 20 illustrates the representation of English in the English transcript file, whereas Figure 21 demonstrates the reversed Arabic sentence in the Arabic transcript file.

```
/path/to/1.wav AND IT WAS A MATTER OF COURSE THAT IN THE MIDDLE AGES WHEN THE CRAFTSMEN
/path/to/2.wav AND WAS IN FACT THE KIND OF LETTER USED IN THE MANY SPLENDID MISSALS PSALTERS PRODUCEC
/path/to/3.wav JOHN OF SPIRES AND HIS BROTHER VINDELIN FOLLOWED BY NICHOLAS JENSON BEGAN TO PRINT IN
/path/to/4.wav BEING THIN TOUGH AND OPAQUE
```

Figure 20: a snippet of the English transcript file

```
/home/ubuntu/yousif/data/test/common_voice_ar_19216700.wav قلم الديك
/home/ubuntu/yousif/data/test/common_voice_ar_23558552.wav أمس يوم من أبعد الأرض هذه على مسافة هناك ليست
/home/ubuntu/yousif/data/test/common_voice_ar_23675091.wav المشكلة تكبر إنك
/home/ubuntu/yousif/data/test/common_voice_ar_23974261.wav بك يلتفن أن يرغب
/home/ubuntu/yousif/data/test/common_voice_ar_23972887.wav حتى لماذا يعرفون لا إنهم
/home/ubuntu/yousif/data/test/common_voice_ar_22759417.wav تحب وقت أي مساعدتك سيسعدني
/home/ubuntu/yousif/data/test/common_voice_ar_23703284.wav المفقودة الأمتعة من بالكامل مكونة زحل حلقات أن هي إلى علمية نظرية أحب
```

Figure 21: a snippet of the Arabic transcript file

4.2   Pre-training Phase

During pre-training, representations of speech audio are learnt by solving a contrastive task (see section 2.7.1). The pre-training phase requires having a set of distractors from which to identify the correct quantized latent audio representation. This phase took more than 21 days (504 hours), it was then stopped, after 662 epochs for the purpose of fine-tuning.

$$\frac{p(z_{i+k}|z_i...z_{i-r})}{p(z_{i+k})} \tag{34}$$

During the pre-training phase, the model predicts future samples from a given signal context, this poses a challenge when modeling the data distribution $p(x)$ due to the difficulty associated with calculating feature samples for high-frequency speech signals. To avoid this problem, raw speech samples x are encoded into a feature representation z at a lower temporal frequency, and then equation 34 shows the density modeling ratio.

4.3 Fine-tuning Phase

After the pre-training phase, the model is fine-tuned on the labeled dataset by adding an output layer on top of the Transformer network for prediction. The model is optimized with Adam optimizer [60] with a batch size of 3.2-minute samples.

4.4 Pre-trained Cross-language Model

A second methodology is used in this research, where multilingual pre-trained models are explored. The pre-trained model used is XLSR-Wav2Vec2, the successor to wav2vec, this model learns basic speech units used to tackle a self-supervised task for several languages, and is pre-trained and released by Facebook AI Research in September 2020. The pre-trained model is trained the same way *wav2vec2.0* is, but on 53 different languages. This model can benefit from the availability of some languages and use them for less available languages. XLSR-Wav2Vec2 is fine-tuned using CTC for calculating the loss by summing up all scores of all possible alignments between input and output.

XLSR-Wav2Vec2 model has many hyperparameters to tune, nonetheless, a small number of these parameters, that we fine-tune during the fine-tuning phase, is mentioned here:

save_steps is the number of update steps before two checkpoint saves.
eval_steps is when evaluation is done (and logged) by calculating the WER.
num_train_epochs is the number of full iteration on our data the neural network is going to perform.
per_device_train_batch_size is the batch size per GPU core.
gradient_accumulation_steps is the number of updates steps to accumulate the gradients for, before performing a backward/update pass.
save_total_limit is the number of checkpoints to save.

5 Result

5.1 The result of the monolingual model

Figure 22 shows training and validation accuracies (which indicates how accurate the model predicts the true masked vector) during pre-training on Mozilla Common Voice training dataset. As it can be seen in the chart, the training accuracy was in a very slow increase trend, whereas the validation accuracy never went above 0.601 except for the first 50 warmup epochs. The validation accuracy started to slightly decrease after around 450 epochs (on the fifteenth day of pre-training). This accuracy refers to how good our contrastive learning is, in other words, how accurate is our prediction of the true masked vectors.
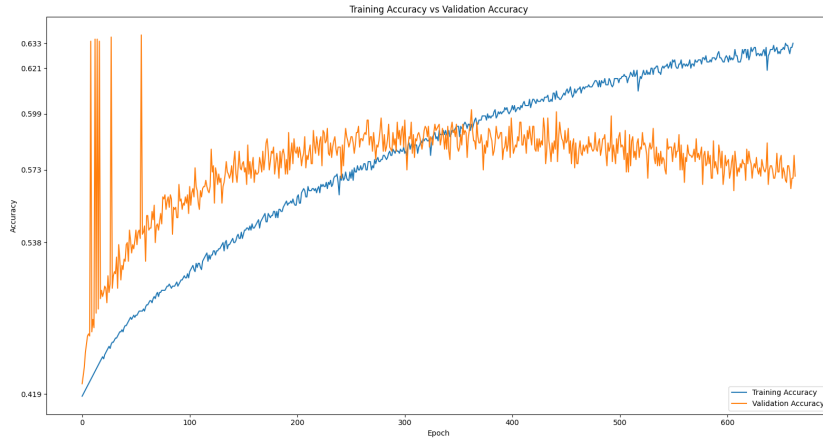

Figure 22: Training Accuracy During Pre-training

The pre-training phase took more than 21.5 days when it was stopped for the purpose of fine-tuning, there was no early-sopping and the change in validation accuracy was as low as 0.002. Since the pre-training phase on the monolingual dataset was computationally expensive and had low accuracy, it was decided to choose a cross-lingual pre-trained model. Therefore, the fine-tuning for this model was also stopped before it was finished.

5.2 The result of the pre-trained XLSR model

Figure 23 illustrates how WER decreases as the number of epochs and steps increases, it also shows how the training and validation losses decrease, as well as the number of steps.

Figure 24 shows the word error rate with 250 warmup steps, *3e-4* learning rate, batch size of 8, gradient accumulation of 8 steps, and we evaluate the model every 400 steps.

Figure 25 shows the predicted sentence for one of the audio files from the test dataset. As it can be seen in figure 25 the predicted sentence and the source sentence (Reference) are almost the same except for two words. One of the two words is altered by *Insertion* and the second word is changed by both *Deletion* and *Insertion* in terms of WER estimation.

Figure 26 shows different models with WER that are compared with each other based on WER, as we can see our model placed third with WER of 36.69 % [61]. The first

[18120/18120 7:17:26, Epoch 30/30]

Step	Training Loss	Validation Loss	Wer	Runtime	Samples Per Second
12000	0.992300	0.430847	0.399565	645.960700	11.799000
12500	0.235200	0.434993	0.389307	655.163400	11.634000
13000	0.222300	0.433335	0.393477	661.358800	11.525000
13500	0.208600	0.427543	0.385981	640.659700	11.897000
14000	0.202800	0.435244	0.386032	651.793900	11.694000
14500	0.194900	0.428143	0.380481	675.525800	11.283000
15000	0.186000	0.447510	0.383269	661.993100	11.514000
15500	0.182800	0.441939	0.374981	653.909600	11.656000
16000	0.177600	0.438888	0.376618	675.407800	11.285000
16500	0.175000	0.439787	0.374162	674.119100	11.307000
17000	0.166900	0.427809	0.373011	665.235000	11.458000
17500	0.167500	0.432440	0.371706	673.437200	11.318000
18000	0.163600	0.436680	0.370683	692.292300	11.010000

The training is done

Figure 23: Training Loss, Validation Loss, and WER of Training + Validation Datasets

Special tokens have been added in the vocabulary, make sure the associated word embedding are fine-tuned or trained.

100%   7622/7622 [01:44<00:00, 72.75ex/s]

100%   7622/7622 [03:18<00:00, 38.38ex/s]

100%   953/953 [07:33<00:00, 2.10ba/s]

WER: 36.531082

Figure 24: WER Result for Test Dataset

Prediction: ['ليست نارك مكسافة على هذه الأرض أبعد من يوم أمس' , 'ألديك قلم']
Reference: ['ليست هناك مسافة على هذه الأرض أبعد من يوم أمس' , '؟ ألديك قلم.']

Figure 25: Sample Prediction with 0.369 WER

model is derived from the second model by fine-tuning the model hyperparameters.

View	Test WER					

RANK	MODEL	TEST WER ↓	VALIDATION WER	TEST CER	DETAILS	YEAR
1	Sinai Voice Arabic Speech Recognition Model	23.8%			≋	2021
2	elgeish XLSR Wav2Vec2 Large 53	26.55%	23.39		≋	2021
3	Mohammed XLSR Wav2Vec2 Large 53	36.69%	36.69		≋	2021
4	Mohammed XLSR Wav2Vec2 Large 53	36.699%	36.699		≋	2021
5	XLSR Wav2Vec2 Arabic by Jonatas Grosman	40.52%		18.37%	≋	2021
6	XLSR Wav2Vec2 Arabic by Othmane Rifki	46.77%			≋	2021
7	XLSR Wav2Vec2 Arabic by Othmane Rifki	46.77%			≋	2021
8	Hasni XLSR Wav2Vec2 Large 53	52.18%			≋	2021

Figure 26: A comparison between different fine-tuned models

6   Discussion

Self-supervised learning provides a great way of making use of unlabeled datasets. The data that is used for this research is unlabeled data for the pre-training phase and labeled data for fine-tuning. SSL had been there, before the introduction of wav2vec framework, to automatically generate supervisory signals from unlabeled data to solve a task (a contrastive one in our research). But it was not until wav2vec was introduced in September 2019 by Facebook AI Research, that we have seen a significant improvement in SSL application in speech recognition. wav2vec is a CNN that takes raw audio as input and produces general representations that could be used as input to an ASR. It accomplishes that by solving a contrastive task that requires differentiating between the true future label from negative ones. As oppose to phoneme classification, the learned representations are applied to improve ASR systems.

Performing a pre-training task on a single language model with a single input language is nowhere near as accurate as pre-training ASR models on multiple languages and then fine-tune it on our target language (Arabic). This was proven by Facebook AI Research [52], and was also seen clearly when we pre-trained our model on only Arabic input, which resulted in 58% accuracy during pre-training, and that was a good reason for sopping the fine-tuning phase on labeled data as the contrastive loss is high. This could be attributed to: firstly, the huge amount of training data for cross-lingual model one could acquire as opposed to finding a monolingual model dataset. Secondly, as the humans' sound-generating system is designed in the same way, it does make sense to use other languages, with the target language that we want to fine-tune our model on, to pre-train a model then fine-tune that model on that target language alone.

Moreover, the process of having pre-trained models for each language for the purpose of fine-tuning these models on languages they were pre-trained on is time-consuming and resource-intensive. It could be seen from our pre-training phase that more than 21 days were taken to pre-train our model (21.5 days) without convergence. Despite the Nvidia Tesla G80 GPU that was used for pre-training.

Fine-tuning our model on the same dataset (Mozilla Common Voice) with different diacritics removal techniques and model parameters has resulted in a different dictionary and a higher accuracy compared to other models fine-tuned on the same datasets but with different diacritics removal techniques. There are many diacritics in the Arabic language that need to be removed before in the preprocessing step before doing any model training, these diacritics could be difficult to number, and that is the reason we saw different models scoring low WER despite them having excellent hyperparameter tuning. For instance, one of the models uploaded on huggingface.co is reported to have had WER of 46.77 % [62], however, when checked with our diacritics removal script on the dataset the model was tested on, the model's accuracy increased, with a WER of 45.5.

Diacritics can be confusing and using them might lead to low accuracy during model fine-tuning because they are neither letters nor spaces, when they were included in the fine-tuning phase they were considered by the model as language letters in the language dictionary. This dictionary is stored by the framework for comparing the correct characters and assigning codes to them. Therefore, the removal of all diacritics from the transcript files resulted in higher accuracy and low WER.

The model initially scored 56.22 % WER (Figure 27) before hyperparameters tuning. After removing all diacritics and tuning hyperparameters, our model achieved a WER of 36.69 %. It was then reported to be compared with other models [63]. This accuracy placed our model on the top three Arabic language models used in huggingface speech recognition [61], after a model that scored 26.55 for WER, and another model that was tuned with different hyperparameters, thus, decreased the WER to 23.8 %.

Figure 27: Initial WER before hyperparameters fine-tuning

The model that scored the best accuracy used *transliteration*, which is the process of changing a word from the alphabet of its original language (source language) to another language (target language). By performing transliteration on Arabic (changing all its letters to their corresponding English letters), avoidance of diacritics problem is for certain as we no longer have diacritics. However, having manually removed diacritics from Arabic has also resulted in high accuracy compared to other models.

This model could have been improved and other models could have been developed with a different number of network layers, different vector quantization techniques, or different language models. But due to the limited time allocated for this thesis, it was not possible to use other models.

7   Conclusion

In this project, an Automatic Speech Recognition (ASR) system was implemented based on a small amount of data for Arabic language. Self-supervised learning (SSL) was the main means used for conducting this research. A pre-training phase was, first, carried out on 42 hours of unlabeled Arabic language dataset from *Mozilla Common Voice* for the purpose of fine-tuning this pre-trained model on labeled Arabic data, the pre-training took more than 21 days without finishing. Due to this, attention was instead directed to using a pre-trained model available online. This study, secondly, took advantage of an unsupervised cross-lingual speech representation (XLSR) that had been pre-trained on multiple languages, including Arabic, and then this pre-trained model was fine-tuned on Arabic language. The unsupervised XLSR was pre-trained on 53 different languages and built on *wav2vec 2.0* which solves a contrastive task for masked latent speech representations. This model was proven to be significantly more efficient than the monolingual pre-training.

The Word Error Rate (WER) of the fine-tuned *XLSR* with *wav2vec 2.0* on Arabic language was initially 54.00 %. After fine-tuning the model parameters, we obtained a WER of 36.69 % which placed this model third in *huggingface* company's Arabic language models. It is now being used and has been downloaded 32 times from the official website of *huggingface* (https://huggingface.co/mohammed/ar).

The first research question is about the scarcity of transcribed data, which is addressed by open-source projects such as *Mozilla Common Voice* that enables communities to participate in creating speech corpus for different languages. The speech data used in this study was obtained from *Mozilla Common Voice*. Additionally, finding a huge amount of labeled data has become no longer a concern given *self-supervised learning* (SSL). SSL, which is our second question of this research, has proven successful when dealing with unlabeled Arabic speech data. Cross-lingual speech representation (XLSR) was used in this research to utilize unlabeled data from multiple languages in the pre-training phase and then fine-tune our model on labeled Arabic language data. Using pre-trained models for Automatic Speech Recognition, and fine-tuning them on labeled Arabic speech data, was more efficient and time-saving than pre-training a monolingual model. The third question takes the model accuracy given the amount of labeled data used, and we proved that by using as few as 42 hours of labeled data we could achieve a WER of 36.69 %.

8   Future Work

Data augmentation is one of the techniques that was not used in this research and could be used to decrease WER. It is implemented by cropping audio sequences of a predefined length when creating batches. This method could increase the performance if all cropped audio (sequences) have the same length. However, if the maximum length for different audio sequences is different, then it might result in a worse accuracy [51]. Increasing the amount of unlabeled data for pre-training or labeled data fine-tuning would also increase the model accuracy. *wav2vec Unsupervised* (*wav2vec-U*) can be used to train speech recognition models without labeled data, it leverages self-supervised speech representations to segment unlabeled audio and learn a mapping from these representations to phonemes via adversarial training [64].

9 References

[1]   U. Shrawankar and V. Thakare, "Adverse conditions and ASR techniques for robust speech user interface." 2013, [Online]. Available: http://arxiv.org/abs/1303.5515.

[2]   A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," *arXiv e-prints*, p. arXiv:2006.11477, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.11477.

[3]   A. Hannun *et al.*, "Deep speech: Scaling up end-to-end speech recognition." 2014, [Online]. Available: http://arxiv.org/abs/1412.5567.

[4]   "Mozilla common voice." https://commonvoice.mozilla.org/en/about.

[5]   L. R. Rabiner and B. H. Juang, "Hidden markov models for speech recognition — strengths and limitations," in *Speech recognition and understanding*, 1992, pp. 3–29.

[6]   P. Pujol, S. Pol, C. Nadeu, A. Hagen, and H. Bourlard, "Comparison and combination of features in a hybrid HMM/MLP and a HMM/GMM speech recognition system," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 1, pp. 14–22, 2005, doi: 10.1109/TSA.2004.834466.

[7]   L. Li *et al.*, "Hybrid deep neural network–hidden markov model (DNN-HMM) based speech emotion recognition," in *2013 humaine association conference on affective computing and intelligent interaction*, 2013, pp. 312–317, doi: 10.1109/ACII.2013.58.

[8]   S. Xue, O. Abdel-Hamid, H. Jiang, and L. Dai, "Direct adaptation of hybrid DNN/HMM model for fast speaker adaptation in LVCSR based on speaker code," in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2014, pp. 6339–6343, doi: 10.1109/ICASSP.2014.6854824.

[9]   G. F. S. Eberhard David M. and C. D. Fennig, "Ethnologue: Languages of the world. Twenty-third edition. Dallas, texas: SIL international," *Online version: http://www.ethnologue.com*, 2020.

[10]   A. Baevski, S. Schneider, and M. Auli, "Vq-wav2vec: Self-supervised learning of discrete speech representations." 2020, [Online]. Available: http://arxiv.org/abs/1910.05453.

[11]   A. Skodo, "Sweden: By turns welcoming and restrictive in its immigration policy," *Online version: https://www.migrationpolicy.org/article/sweden-turns-welcoming-and-restrictive-its-immigration-policy#:*, 2018.

[12]   J. Garofolo *et al.*, "TIMIT acoustic-phonetic continuous speech corpus," *Linguistic Data Consortium*, Nov. 1992.

[13]   O. Enshassi, "Adaptation of acoustic and language model for improving arabic automatic speech recognition," PhD thesis, 2016.

[14]   E. Chang, J.-L. Zhou, S. Di, C. Huang, and K.-F. Lee, "Large vocabulary mandarin speech recognition with different approaches in modeling tones." Jan. 2000, pp. 983–986.

[15]   J. Ivanecký, V. Fischer, and S. Kunzmann, "French–german bilingual acoustic modeling for embedded voice driven applications," Aug. 2005, pp. 744–744, doi: 10.1007/11551874_30.

[16]   W. Byrne *et al.*, "Towards language independent acoustic modeling," in *2000 IEEE international conference on acoustics, speech, and signal processing. Proceedings (cat. no.00CH37100)*, 2000, vol. 2, pp. II1029–II1032 vol.2, doi: 10.1109/ICASSP.2000.859138.

[17]   U. Shrawankar and V. M. Thakare, "Techniques for feature extraction in speech recognition system : A comparative study." 2013, [Online]. Available: http://arxiv.org/abs/1305.1145.

[18]   H. F. Pardede, V. Zilvan, D. Krisnandi, A. Heryana, and R. B. S. Kusumo, "Generalized filter-bank features for robust speech recognition against reverberation," in *2019 international conference on computer, control, informatics and its applications (IC3INA)*, 2019, pp. 19–24, doi: 10.1109/IC3INA48034.2019.8949593.

[19]   J. W. Cooley, P. A. W. Lewis, and P. D. Welch, "The fast fourier transform and its applications," *IEEE Transactions on Education*, vol. 12, no. 1, pp. 27–34, 1969, doi: 10.1109/TE.1969.4320436.

[20]   G. V. Haines and A. G. Jones, "Logarithmic Fourier transformation," *Geophysical Journal International*, vol. 92, no. 1, pp. 171–178, Jan. 1988, doi: 10.1111/j.1365-246X.1988.tb01131.x.

[21]   M. J. F. Gales, S. Watanabe, and E. Fosler-Lussier, "Structured discriminative models for speech recognition: An overview," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 70–81, 2012, doi: 10.1109/MSP.2012.2207140.

[22]   J. M. Baker *et al.*, "Developments and directions in speech recognition and understanding, part 1 [DSP education]," *IEEE Signal Processing Magazine*, vol. 26, no. 3, pp. 75–80, 2009, doi: 10.1109/MSP.2009.932166.

[23]   "Pronunciation model." https://maelfabien.github.io/machinelearning/speech_reco/#pronunciation.

[24]   G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012, doi: 10.1109/MSP.2012.2205597.

[25]   Jong-Hwan Lee, Ho-Young Jung, Te-Won Lee, and Soo-Young Lee, "Speech feature extraction using independent component analysis," in *2000 IEEE international conference on acoustics, speech, and signal processing. Proceedings (cat. no.00CH37100)*, 2000, vol. 3, pp. 1631–1634 vol.3, doi: 10.1109/ICASSP.2000.862023.

[26]   H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition." 2015, [Online]. Available: http://arxiv.org/abs/1507.06947.

[27] A. V. Nefian, L. Liang, X. Pi, L. Xiaoxiang, C. Mao, and K. Murphy, "A coupled HMM for audio-visual speech recognition," in *2002 IEEE international conference on acoustics, speech, and signal processing*, 2002, vol. 2, pp. II-2013-II-2016, doi: 10.1109/ICASSP.2002.5745027.

[28] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin." 2015, [Online]. Available: http://arxiv.org/abs/1512.02595.

[29] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, 2014, doi: 10.1109/TASLP.2014.2339736.

[30] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/s13244-018-0639-9.

[31] "Kernel size and strides." https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9.

[32] Q. Lemaire and A. Holzapfel, "Temporal convolutional networks for speech and music detection in radio broadcast," 2019.

[33] X. Tian *et al.*, "Deep LSTM for large vocabulary continuous speech recognition." 2017, [Online]. Available: http://arxiv.org/abs/1703.07090.

[34] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks." 2013, [Online]. Available: http://arxiv.org/abs/1303.5778.

[35] A. Vaswani *et al.*, "Attention is all you need." 2017, [Online]. Available: http://arxiv.org/abs/1706.03762.

[36] L. Lu, C. Liu, J. Li, and Y. Gong, "Exploring transformers for large-scale speech recognition." 2020, [Online]. Available: http://arxiv.org/abs/2005.09684.

[37] R. Horev, "BERT explained: State of the art language model for NLP." 2018, [Online]. Available: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[38] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context." 2019, [Online]. Available: http://arxiv.org/abs/1901.02860.

[39] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty," *arXiv e-prints*, p. arXiv:1906.12340, Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.12340.

[40] M. E. Peters *et al.*, "Deep contextualized word representations." 2018, [Online]. Available: http://arxiv.org/abs/1802.05365.

[41] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[42] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding." 2019, [Online]. Available: http://arxiv.org/abs/1810.04805.

[43] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning." 2020, [Online]. Available: http://arxiv.org/abs/1911.05722.

[44] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations." 2020, [Online]. Available: http://arxiv.org/abs/2002.05709.

[45] D. Jiang *et al.*, "Improving transformer-based speech recognition using unsupervised pre-training." 2019, [Online]. Available: http://arxiv.org/abs/1910.09932.

[46] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011, doi: 10.1109/TPAMI.2010.57.

[47] S. Dieleman, A. van den Oord, and K. Simonyan, "The challenge of realistic music generation: Modelling raw audio at scale." 2018, [Online]. Available: http://arxiv.org/abs/1806.10474.

[48] A. F. Agarap, "Deep learning using rectified linear units (ReLU)." 2019, [Online]. Available: http://arxiv.org/abs/1803.08375.

[49] V. Joshi, R. Zhao, R. R. Mehta, K. Kumar, and J. Li, "Transfer learning approaches for streaming end-to-end speech recognition system." 2020, [Online]. Available: http://arxiv.org/abs/2008.05086.

[50] L. N. M. and S. K. Kopparapu, "Choice of mel filter bank in computing MFCC of a resampled speech." 2014, [Online]. Available: http://arxiv.org/abs/1410.6903.

[51] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition." 2019, [Online]. Available: http://arxiv.org/abs/1904.05862.

[52] G. Lample and A. Conneau, "Cross-lingual language model pretraining." 2019, [Online]. Available: http://arxiv.org/abs/1901.07291.

[53] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition." 2020, [Online]. Available: http://arxiv.org/abs/2006.13979.

[54] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks," in *ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning*, Jan. 2006, vol. 2006, pp. 369–376, doi: 10.1145/1143844.1143891.

[55] "CTC." https://distill.pub/2017/ctc/.

[56]   R. Errattahi, A. El Hannani, and H. Ouahmane, "Automatic speech recognition errors detection and correction: A review," *Procedia Computer Science*, vol. 128, pp. 32–37, 2018, doi: https://doi.org/10.1016/j.procs.2018.03.005.

[57]   E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 522–532, 1998, doi: 10.1109/34.682181.

[58]   S. M. Abdou and A. M. Moussa, "Arabic speech recognition: Challenges and state of the art," in *Computational linguistics, speech and image processing for arabic language*, WORLD SCIENTIFIC, 2018, pp. 1–27.

[59]   M. Hssini and A. Lazrek, "Design of arabic diacritical marks." 2011, [Online]. Available: http://arxiv.org/abs/1107.4734.

[60]   D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." 2017, [Online]. Available: http://arxiv.org/abs/1412.6980.

[61]   "Othrif model huggingface." https://paperswithcode.com/sota/speech-recognition-on-common-voice-arabic.

[62]   "Othrif model huggingface." https://huggingface.co/othrif/wav2vec2-large-xlsr-arabic.

[63]   "The model huggingface." https://huggingface.co/mohammed/ar.

[64]   A. Baevski, W.-N. Hsu, A. Conneau, and M. Auli, "Unsupervised speech recognition." 2021, [Online]. Available: http://arxiv.org/abs/2105.11084.