Unsupervised Pre-Training for Vietnamese
Automatic Speech Recognition in the HYKIST Project

Le Duc Khai

Matrikelnummer: 3089345

Jülich, November 30, 2022

Eigenständigkeitserklärung

Ich erkläre hiermit, dass ich diese Bachelorarbeit selbstständig, ohne unzulässige Hilfe durch Dritte und ohne Benutzung anderer als der angegeben Hilfsmittel angefertigt habe. Insbesondere versichere ich, die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Mir ist bekannt, dass meine Arbeit zum Zwecke eines Plagiatsabgleichs mittels einer Plagiatserkennungssoftware auf ungekennzeichnete Übernahme von fremdem geistigem Eigentum überprüft werden kann.

Jülich,

_____

Le Duc Khai

Contents

1 Abstract

In today's interconnected globe, moving abroad is more and more prevalent, whether it's for employment, refugee resettlement, or other causes. Language difficulties between natives and immigrants present a common issue on a daily basis, especially in medical domain. This can make it difficult for patients and doctors to communicate during anamnesis or in the emergency room, which compromises patient care. The goal of the HYKIST Project is to develop a speech translation system to support patient-doctor communication with *ASR* and *MT*.

ASR systems have recently displayed astounding performance on particular tasks for which enough quantities of training data are available, such as LibriSpeech [53]. Building a good model is still difficult due to a variety of speaking styles, acoustic and recording settings, and a lack of in-domain training data. In this thesis, we describe our efforts to construct *ASR* systems for a conversational telephone speech recognition task in the medical domain for Vietnamese language to assist emergency room contact between doctors and patients across linguistic barriers. In order to enhance the system's performance, we investigate various training schedules and data combining strategies. We also examine how best to make use of the little data that is available. The use of publicly accessible models like *XLSR-53* [14] is compared to the use of customized pre-trained models, and both supervised and unsupervised approaches are utilized using *wav2vec 2.0* [6] as architecture.

2 Introduction

2.1 HYKIST Project

Migration to foreign countries is becoming more common in our globally connected world, whether for work, refugee movements, or other reasons. As a result, language barriers between locals and foreigners are a common daily issue. It is commonly known that, speaking with patients when they arrive at the hospital is crucial to their care. In medical care, a lack of or incorrect communication leads to underuse and misuse of medical services, lower quality of care, an increased rate of treatment errors, ineffective preventive measures for patients, and medical staff dissatisfaction. The doctors then inquire about the patient's problems as well as his or her medical history. However, there are currently 20.8 million immigrants in Germany, with up to 30% having only basic German language skills[1]. If doctors and patients do not speak the same language, information communication is severely constrained, which has a negative impact on the patients' care. In the event that no common language is available, doctors can contact Triaphon which provides translators to aid communication between the patient and the doctor. These bi-lingual interpreters then assist in communication between the patient and the doctor.

In the HYKIST scenario, the doctor talks German to the patient, who speaks only Arabic or Vietnamese. Meanwhile, German and Arabic, or German and Vietnamese, are the languages spoken by the interpreters. The interpreters are not professional translators, instead, they are volunteers who contribute their time to the translation. This is problematic because the interpreters may require time to look up unfamiliar words, such as medical termini, or they may make a mistake.

The ultimate goal of the HYKIST project is to facilitate doctor-patient communication in a growing number of languages with the help of *ASR* and *MT* in order to meet the robust medical domain requirements via following steps: The interpreter is summoned via

---

[1]https://www.apptek.com/news/germanys-federal-ministry-of-health-awards-hykist-project-to-apptek-to-equip-critical-care-with-artificial-intelligence-driven-automatic-speech-translation-technology

the hospital phone, which has an audio sampling rate of 8 kHz. We then create manual annotations with helps of our native-speaker volunteers. We investigate the use of additional outside-the-domain data for training as well as unsupervised methods because gathering project-specific data is an expensive and time-consuming operation.

ASR and *MT* technologies are linked with a dialogue system for initial anamnesis and integrated into an existing telecommunications platform for this purpose. First and foremost, the project collects dialogues in Arabic, Vietnamese, and German, which serve as the foundation for the development of algorithms and applications. During the project, the first technical tests for the accuracy and quality of the automated translations are already being performed. Following that, the overall system must be tested in a pilot test with clinical application partners for the area of emergency admissions and initial anamnesis in acute situations, as well as evaluated in a final clinical study for user acceptance.

The partners in the HYKIST Project are Triaphon[2], Fraunhofer Focus[3] and AppTek GmbH[4].

---

[2]https://triaphon.org/
[3]https://www.fokus.fraunhofer.de/en
[4]https://www.apptek.com

2.2 Motivation

Large amounts of labeled training data benefit neural networks. However, labeled data is much more difficult to obtain in many settings than unlabeled data: current speech recognition systems require thousands of hours of transcribed speech to achieve acceptable performance, which is not available for the vast majority of the nearly 7,000 languages spoken globally [42]. Learning solely from labeled examples is not comparable to human language acquisition: infants learn language by listening to adults around them - a process that necessitates the acquisition of good representations of speech. Therefore, semi-supervised learning aims to work like the natural language acquisition of human.

Unsupervised and semi-supervised methods have been shown to be successful in *ASR* in recent years. *wav2vec 2.0* [6], in particular, has demonstrated excellent performance. *wav2vec 2.0* is pre-trained using an unsupervised loss before being fine-tuned on labeled data. The goal of the paper is to offer a framework for self-supervised learning of representations from raw audio data. This framework opens the door for speech recognition models to be used in a low-resource language like Vietnamese in medical domain where previously much more transcribed audio data was required to provide acceptable accuracy. The model is then fine-tuned on labeled data in a hybrid framework [46] after pre-training on unlabeled speech.

In the HYKIST Project, we want to utilize the *wav2vec 2.0* model. One interesting aspect of *wav2vec 2.0* is that the unsupervised pre-training is well suited for exploiting unlabeled multilingual data so that supervised training on a target language gains benefit from multilingual speech representations. In [14], the authors focused on learning representations from unlabeled data that generalize across languages in a multilingual scenario. They built on *wav2vec 2.0* pretraining technique, in which a discrete vocabulary of *Latent Speech Representations* is learned alongside contextualized speech representations. We can utilize their public model *XLSR-53* because it was unsupervised pretrained on 8 languages from Multilingual LibriSpeech [57], 17 languages from the BABEL benchmark [18], which is conversational telephone data with Vietnamese language included, as well as 36 languages from Common-Voice [3], which is a corpus of read speech. With the exception of resource-rich languages, multilingual pretraining surpassed monolingual pretraining in most circumstances.

2.3 Related work

Having been an established and effective method for *ASR*, hybrid modeling has made steady progress in recent years and outperformed *End-to-End (E2E)* approach in most *ASR* situations [46]. Besides, the recent introduction of novel neural encoders has been reported to significantly improve the performance [76, 23, 73]. Other methods can also be used to achieve even greater improvements, like feature combination [71] or additional losses in the intermediate layers [68]. Furthermore, unsupervised approaches have grown in popularity due to their potential for high performance with little annotated data [48]. Semi-supervised learning was applied to an *ASR* task by [34, 62, 6] by running unsupervised pre-training on a large unlabeled dataset, followed by fine-tuning on a small annotated dataset. This technique can significantly reduce the amount of labeled data required to build *ASR* systems. The successes sparked additional research into improving the modeling approach [29, 64] and analyzing which individual components contribute most to the performance [55]. Besides, data used for pre-training and fine-tuning was deeply investigated as well, for example, in a domain-shift scenario [30] in English language or using multilingual data for the sake of improvements on monolingual benchmarks [14].

Because the contrastive loss is computed solely on the input speech audio and does not require labels, it is especially simple to use for monolingual or multilingual data. Therefore, a number of papers have begun to apply this loss for *ASR* research [14, 72, 77, 7]. Previously, supervised training with multilingual data could improve low resource languages by using a separate output layer for each language [69]. There has also been research specifically addressing medical domain tasks. However, a common problem for medical *ASR* faced by researchers is difficult acoustic conditions and a lack of transcribed medical audio data [17, 13, 33]. Another difficulty likely to be met is the medical terminology. In [60], a multilingual system for the medical domain is presented. Another method for dealing with the medical domain is to correct *ASR* errors at the output level [47].

To the best of our knowledge, unsupervised pretraining methods have mostly been investigated on well-known academic datasets, with no work done on applying them to difficult low-resource medical tasks. Furthermore, no previous work has been published that investigates the use of unsupervised pretraining methods for telephone speech directly on the 8kHz signal without resampling. Besides, the analysis of different pretraining data combination and regularization for a medical *ASR* system has never been presented.

3 Theory

3.1 Hybrid ASR framework

3.1.1 Bayes theorem

Given a sequence of acoustic observations x_1^T whose length is T, the most likely word sequence to be recognized is w_1^N. A variety of subword units, such as phonemes, and the acoustic representation of the audio signal are connected through acoustic models. In terms of probabilities, the relation w^* between the acoustic and word sequence is described as:

$$w^* = \arg\max_{w_1^N} p(w_1^N | x_1^T) \tag{3.1}$$

As stated in the introduction, conventional *ASR* systems typically consist of a number of modules, including dictionaries, language models, and acoustic models. By utilizing Bayes' Theorem to break out the posterior probability, it is possible to show the connections between them. For the maximization, the probability $p(x)$ can be ignored because it just acts as a normalization and has no bearing on the outcome.

$$p(w_1^N | x_1^T) = \frac{p(x_1^T | w_1^N) p(w_1^N)}{p(x_1^T)} \propto p(x_1^T | w_1^N) p(w_1^N) \tag{3.2}$$

$$w^* = \arg\max_{w_1^N} \underbrace{p(x_1^T | w_1^N)}_{\text{acoustic model}} \cdot \underbrace{p(w_1^N)}_{\text{language model}} \tag{3.3}$$

3.1.2 Audio features

The classification model uses features, which are representations taken from audio samples and used as input. There are many features, and they all show the spoken audio's frequency

information. Statistical models must learn some rather long-term dependencies within the input data due to the high resolution in the time-domain, which is often quite challenging and computationally expensive. As a result, we leverage acoustic features to simplify the signal while preserving the most crucial statistics.

Mel-frequency cepstral coefficient (MFCC): The windowing of the signal, application of the *Discrete Fourier Transform (DFT)*, calculation of the magnitude's log, warping of the frequencies on a Mel scale, and application of the inverse *Discrete Cosine Transform (DCT)* are the main steps in the *MFCC* feature extraction technique. Below is a short explanation [58] of each stage in the *MFCC* feature extraction process.

1. Pre-emphasis: Filtering that highlights the higher frequencies is referred to as pre-emphasis. Its function is to balance the spectrum of spoken sounds, which roll off sharply at high frequencies.

2. Frame blocking and windowing: Speech analysis over a short enough time span is required for stable acoustic features. The analysis must therefore always be performed on short segments where the speech signal is believed to be stationary.

3. *DFT* spectrum: Each windowed frame is converted into magnitude spectrum by applying *DFT*

4. Mel spectrum: The Fourier transformed signal is run through the Mel-filter bank, a collection of band-pass filters, to compute the Mel spectrum. A Mel is a unit of measurement based on the perceived frequency by human ears.

5. *Discrete Cosine Transform (DCT)*: Because the vocal tract is smooth, there is a tendency for adjacent bands' energy levels to correlate. When the converted Mel frequency coefficients are applied to the *DCT*, a set of cepstral coefficients are generated.

6. Dynamic MFCC features: Since the cepstral coefficients only include data from a single frame, they are frequently referred to as static features. By computing the first and second derivatives of the cepstral coefficients, additional information on the temporal dynamics of the signal is gained.

Gammatone features: The Gammatone filter [1], which is intended to mimic the human auditory filter, is the foundation for Gammatone features. They were initially presented for large vocabulary *ASR* in [61]. A filterbank of Gammatone filters with center frequencies sampled from the Greenwood function [22] is applied after pre-emphasizing the speech signal. Below is a summary of each stage in the Gammatone feature extraction process:

1. Typically, a Hanning window of 25 ms width with 10 ms shifts is used to perform the temporal integration of the absolute values of the filter outputs.

2. A spectral integration with a 9-channel window and a 4-channel shift followed.

3. (10th root or log) compression was performed, followed by cepstral decorrelation resulting in 16 cepstral coefficients.

4. Following the use of the 10th root compression, a discrete cosine transform (DCT)-based cepstral decorrelation and normalizing methods are used.

Extracted features from raw waveform: The features from raw waveform encoder are extracted by *CNN* feature encoder. First, the feature encoder's raw waveform input is normalized to zero mean and unit variance. The feature encoder contains seven blocks and the temporal convolutions in each block have 512 channels with strides (5,2,2,2,2,2,2) and kernel widths (10,3,3,3,3,2,2). Besides, layer normalization [5], and the *GELU* activation function [26] are also applied. This results in an encoder output frequency of 49 hz with a stride of about 20ms between each sample, and a receptive field of 400 input samples or 25ms of audio. The convolutional layer modeling relative positional embeddings has kernel size 128 and 16 groups.

3.1.3 Acoustic modeling

When modeling the probability $p(x_1^T|w_1^N)$, the length of time sequence T and of word sequence N are often not the same because N is usually much smaller than T. The alignment between the acoustic observations x_1^T and labels w_1^N is unknown and commonly even unclear. The *Hidden Markov Model (HMM)* is a statistical model that introduces a latent alignment by states s_1^T and subsequently modeling the probability of x_1^T for a given alignment to w_1^N [8]. The probability $p(x_1^T|w_1^N)$ is then calculated by adding all possible alignments between the acoustic observation and the labels. Assuming conditional independence of observations when states are given and that states only depend on their predecessor, this sum results in the equation below:

$$p(x_1^T|w_1^N) = \sum_{[s_1^T]} \prod_{t=1}^{T} p(x_t, s_t|s_{t-1}, w_1^N) = \sum_{[s_1^T]} \prod_{t=1}^{T} \underbrace{p(s_t|s_{t-1}, w_1^N)}_{\text{transition prob.}} \cdot \underbrace{p(x_t|s_t, s_{t-1}, w_1^N)}_{\text{emission prob.}} \quad (3.4)$$

A widely accepted simplification is to make the assumption that the last state for the emission probability is independent such that:

$$p(x_t|s_t, s_{t-1}, w_1^N) = p(x_t|s_t, w_1^N) \tag{3.5}$$

The transition model calculates the probabilities of moving from one state to the next. The emission probability models the probability of an acoustic observation based on the current and previous states. When the probability is simplified, it only depends on the current state. The transition model can have several topologies, but the *0-1-2* topology is the most commonly used. The topology is state-independent and has different transition probabilities: staying in the current state, jumping to the next state, or jumping to the second next state. By jumping faster or slower in time, the jump and stay property allows the alignment of labels and acoustic observations to adjust. The emission model calculates the probability of an acoustic observation in the current and previous states.

Context-Dependent Phone: Because a language's vocabulary is typically very large, modeling words directly in the classification is impractical. Phonemes, on the other hand, are frequently used for subword modeling. For better learning, the acoustic articulation of a phoneme is determined by its surroundings, for example the beginning, the middle and the ending part. As a result, multiple phonemes are combined to create triphone or allophone labels.

Classification and Regression Tree (CART) [10]: However, because of the cubic number of phonemes, these are a large class of labels. The possible triphones are greater than the number of observed triphones. Therefore, some share the same *GMM* model. *CART* is a decision tree used to cluster triphones that can share the same *GMM* model. To reduce the number of labels, allophones are clustered using a *CART*, and the subsequent clusters are used as labels.

Baum–Welch algorithm: In practical training of *HMM*, inferring the parameters of the *HMM* is not simple and cannot be done manually. An automated data-driven approach based on the *Expectation–maximization (EM)* algorithm is used instead, with a dataset of acoustic observations with transcriptions. Because the best alignment between acoustic observations and transcriptions is not always available, the *EM* algorithm is initially leveraged with a sub-optimal linear alignment. The observation model and alignment are then iteratively optimized using the steps below:

1. Maximization: Estimate the model parameters using the previously obtained alignment by maximizing the log-likelihood function.

2. Expectation: Using the parameters from step 1, estimate a new alignment.

3. Get back to step 1 until the model fully converges.

GMM/HMM: The *HMM* can be used to model the transition between phones and the corresponding observable. A widely used approach is modelling the emission probabilities for each label with a parametrized *GMM*, resulting the *GMM*/*HMM* method. The *GMM* is a weighted sum over K normal distributions

$$p(x_t|s_t, s_{t-1}, w_1^N) = \sum_{i=1}^{K} c_i \cdot \mathcal{N}(x_t|\mu_i, \sigma_i^2), \tag{3.6}$$

resulting in a multimodal emission probability with parameters μ_i, σ_i and mixture weights c_i for $i \in 1, K$. The mixture weights are non-negative and sum up to unity. Using the simplification in Equation 3.5 the state s_{t-1} can be additionally dropped.

DNN/HMM: Another approach that has been popular is modelling the posterior probability $p(a_{s_t}|x_1^T)$ discriminatively. Usually *Deep Neural Network (DNN)* is leveraged for this purpose, resulting in the *DNN*/*HMM* approach. The purpose of *GMM*/*HMM* system is to generate alignments for the training of *DNN*/*HMM* system [46]. The emission probability in the *HMM* can afterwards be calculated by applying Bayes rule such that:

$$p(x_1^T|a_{s_t}) = \frac{p(a_{s_t}|x_1^T)p(x_1^T)}{p(a_{s_t})}. \tag{3.7}$$

The probability $p(a_{s_t})$ can be estimated as the relative frequency of a_{s_t}. In order to simplify the Bayes decision rule, the probability $p(x_1^T)$ is constant and therefore can be removed.

3.1.4 Language modeling

In a hybrid system, we use the 4-gram count based *Language model (LM)*, using Kneser-Ney Smoothing algorithm [37]. The *LM*s employed all use full-words in the first-pass decoding [9]. In other words, lattice rescoring is not performed in the second-pass decoding.

In order to deal with multiple monolingual text corpora, the first step is to create an *LM* for each monolingual text corpus. Following that, we use a weighting process to combine the *LM*s into a single *LM*, yielding one *LM* for Vietnamese language.

3.1.5 Decoding

In order to recognize the speech given the acoustic observations, the *AM* and *LM* need to be combined following the Bayes decision rule, resulting in:

$$w_1^N = \arg \max_{N, w_1^N} p\Big(\prod_{n=1}^{N} p(w_n | w_{n-m}^{n-1}) \cdot \sum_{[s_1^T]} \prod_{t=1}^{T} p(x_t, s_t | s_{t-1}, w_1^N)\Big) \tag{3.8}$$

With dynamic programming, this maximization can be solved by Viterbi algorithm which recursively computes the maximum path in $O(k^2 T)$ where k and T are vocabulary size and sequence length respectively. The Viterbi approximation can be applied as

$$w_1^N = \arg \max_{N, w_1^N} p\Big(\prod_{n=1}^{N} p(w_n | w_{n-m}^{n-1}) \cdot \max_{[s_1^T]} \prod_{t=1}^{T} p(x_t, s_t | s_{t-1}, w_1^N)\Big), \tag{3.9}$$

so that the optimization reduces to a best-path problem in the alignment graph of all possible predicted words to the acoustic observations. Besides, beam search (*AM* and *LM* pruning) is used in the searching process which only focuses on the most promising predicted words at each time step [51].

3.1.6 Recognition Performance

The *Word-error-rate (WER)* is a widely used indicator of how well an *ASR* system is performing. The percentage of words that were incorrectly predicted is shown by this number. The *ASR* system performs better with a lower value; a *WER* of 0 equals a perfect result. *WER* can be calculated as:

$$\text{WER} = \frac{\text{Substitutions} + \text{Insertions} + \text{Deletions}}{\text{Reference words}} \tag{3.10}$$

3.2 Neural network

A neural network is a set of algorithms that attempts to recognize underlying relationships in a set of data using a process that mimics how the human brain works. Neural network contains layers of interconnected nodes. Each node is known as a perceptron.

3.2.1 Multilayer perceptron

By adding one or more hidden layers, we can get around the drawbacks of linear models. Stacking a lot of fully connected layers on top of one another is the simplest approach to accomplish this. Up until we produce outputs, each layer feeds into the layer above it. The first layers serve as our representation, and the top layer serves as our linear predictor. This design is frequently referred to as a *Multilayer Perceptron (MLP)*.
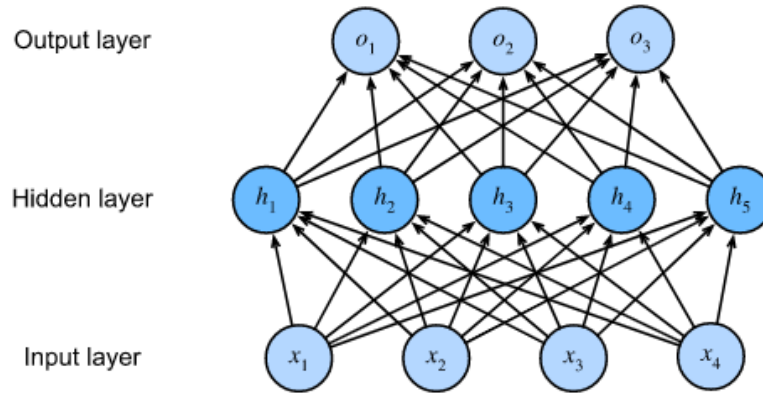

Figure 3.1: An MLP with a hidden layer of 5 hidden units [75]

This *MLP* has 4 inputs, 3 outputs, and 5 hidden units in its hidden layer. Because the input layer does not require any computations, producing outputs with this network necessitates implementing computations for both the hidden and output layers; thus, the number of layers in this *MLP* is 2. It should be noted that both layers are fully connected. Every input influences every neuron in the hidden layer, and every neuron in the output layer influences every neuron in the hidden layer.

We denote by the matrix $X \in R^{n \times d}$ a minibatch of n examples where each example has d inputs (features). For a one-hidden-layer *MLP* whose hidden layer has h hidden units, we denote by $H \in R^{n \times h}$ the outputs of the hidden layer, which are hidden representations. Since the hidden and output layers are both fully connected, we have hidden-layer weights

$W^{(1)} \in R^{d \times h}$ and biases $b^{(1)} \in R^{1 \times h}$ and output-layer weights $W^{(2)} \in R^{h \times q}$ and biases $b^{(2)} \in R^{1 \times q}$. This allows us to calculate the outputs of the one-hidden-layer MLP as follows:

$$H = XW^{(1)} + b^{(1)}$$
$$O = HW^{(2)} + b^{(2)}$$

(3.11)

To fully realize the potential of multilayer architectures, one more key component is required: a nonlinear activation function to be applied to each hidden unit after the affine transformation. For instance, a popular choice is the ReLU (Rectified Linear Unit) activation function [49] $\sigma(x) = \max(0, x)$ operating on its arguments element-wise. The outputs of activation functions are called activations. In general, with activation functions in place, our *MLP* cannot be collapsed into a linear model.

$$H = \sigma(XW^{(1)} + b^{(1)})$$
$$O = HW^{(2)} + b^{(2)}$$

(3.12)

3.2.2 Training a neural network

Epoch: one iteration where the model sees the whole training set to update its weights.

Mini-batch gradient descent: during the training phase, updating weights is usually not based on the whole training set at once due to computation complexities or one data point due to noise issues. Instead, the update step is done on mini-batches, where the number of data points in a batch is a hyperparameter (batch size) that we can tune.

Loss function: In order to quantify how a given model performs, the loss function L is usually used to evaluate to what extent the actual outputs y are correctly predicted by the model outputs z.

Cross-entropy loss: In the context of binary classification in neural networks, the cross-entropy loss $L(z, y)$ is commonly used and is defined as follows:

$$L(z, y) = -[y \log(z) + (1 - y) \log(1 - z)]$$

(3.13)

Forward propagation: The calculation and storage of intermediate variables (including outputs) for a neural network from the input layer to the output layer is referred to as forward propagation (or forward pass).

Backpropagation: The method of calculating the gradient of neural network parameters is known as backpropagation. In short, the method traverses the network in reverse order, from the output to the input layer, using calculus' chain rule. While calculating the gradient with respect to some parameters, the algorithm stores any intermediate variables (partial derivatives).

Updating weights: In a neural network, weights are updated as follows:

Step 1: Take a batch of training data and perform forward propagation (feedforward) to compute the loss.

Step 2: Backpropagate the loss to get the gradient of the loss with respect to each weight

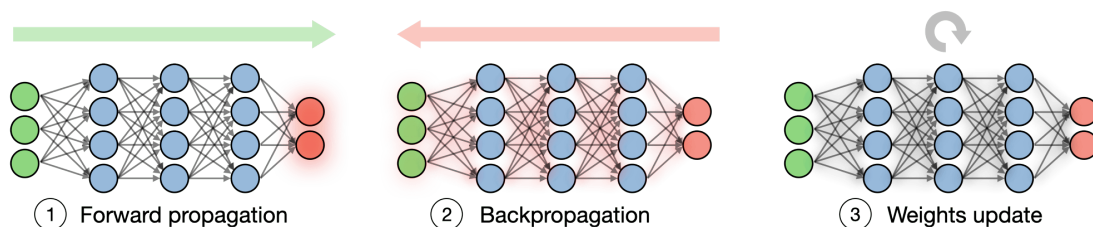Step 3: Use the gradients to update the weights of the network.

Figure 3.2: Updating weights in a neural network [2]

3.2.3 Parameter tuning

Weights initialization:

Xavier initialization [21]: Rather than simply randomizing the weights, Xavier initialization allows for initial weights that take into account characteristics that are unique to the architecture. Weights and inputs are centered at zero, while biases are initialized as zeros.

Transfer learning: It is frequently useful to leverage pre-trained weights from massive datasets that took days/weeks to train and apply them to our use case. Figure 3.3 shows some options for leveraging data, depending on how much we have:
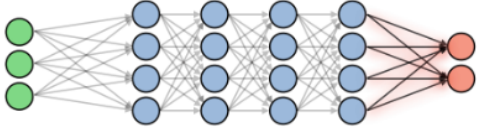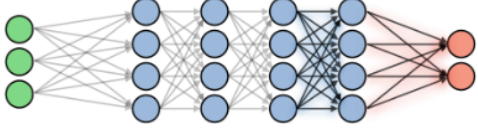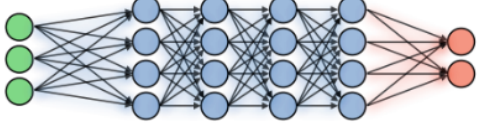
Optimizing convergence:

Training size	Illustration	Explanation
Small		Freezes all layers, trains weights on softmax
Medium		Freezes most layers, trains weights on last layers and softmax
Large		Trains weights on layers and softmax by initializing weights on pre-trained ones

Figure 3.3: Transfer learning strategy [2]

Learning rate: indicates how quickly the weights are updated. It can be fixed or changed adaptively. The most popular method at the moment is Adam [36], which is a method that adapts the learning rate.

Adaptive learning rates: Allowing the learning rate to vary when training a model can help to reduce training time while also improving the numerical optimal solution. While the Adam optimizer is the most commonly used technique, the following in figure 3.4 are also useful:

Method	Explanation	Update of w	Update of b
Momentum	• Dampens oscillations • Improvement to SGD • 2 parameters to tune	$w - \alpha v_{dw}$	$b - \alpha v_{db}$
RMSprop	• Root Mean Square propagation • Speeds up learning algorithm by controlling oscillations	$w - \alpha \dfrac{dw}{\sqrt{s_{dw}}}$	$b \longleftarrow b - \alpha \dfrac{db}{\sqrt{s_{db}}}$
Adam	• Adaptive Moment estimation • Most popular method • 4 parameters to tune	$w - \alpha \dfrac{v_{dw}}{\sqrt{s_{dw}} + \epsilon}$	$b \longleftarrow b - \alpha \dfrac{v_{db}}{\sqrt{s_{db}} + \epsilon}$

Figure 3.4: Adaptive learning rates methods [2]

Regularization:

Dropout [65]: to avoid overfitting the training data by removing neurons with probability $p > 0$. It forces the model to avoid relying too heavily on specific sets of features.

Weight regularization: Regularization techniques are typically used on the model weights to ensure that the weights are not too large and that the model is not overfitting the training set.

Early stopping: to halt training as soon as the validation loss reaches a plateau or begins to rise.

SpecAugment [54]: Rather than augmenting the input audio waveform, SpecAugment applies an augmentation policy directly to the audio spectrogram (i.e., an image representation of the waveform). The spectrogram is altered by warping it in time, masking blocks of consecutive frequency channels, and masking blocks of utterances in time. These augmentations are chosen to help the network to be robust against deformations in the time direction, partial loss of frequency information and partial loss of small segments of speech of the input.

3.2.4 Convolutional Neural Network

Architecture of a traditional *Convolutional Neural Network (CNN)* is generally composed of the following layers:

Convolution layer (CONV): This layer employs filters that perform convolution operations while scanning the input I in terms of its dimensions. The filter size F and stride S are two of its hyperparameters. The resulting output O is referred to as a feature map or an activation map.

Pooling layer (POOL): a downsampling operation used after a convolution layer to achieve spatial invariance. Max and average pooling, in particular, are types of pooling that take the maximum and average value, respectively.

Fully connected layer (FC): works with a flattened input, with each input connected to all neurons. FC layers, when present, are typically found near the end of *CNN* architectures and can be used to optimize objectives such as class scores.

3.2.5 Recurrent Neural Network

Recurrent Neural Network (RNN) is a deep learning model that captures the dynamics of sequences through recurrent connections, which can be viewed as node cycles in a network (connections between nodes can create a cycle). *RNN*s are unrolled across time steps (or sequence steps) using the same underlying parameters at each step. While standard connections are used synchronously to propagate activations from one layer to the next at the same time step, recurrent connections are dynamic, passing information across adjacent time steps. As illustrated in Figure 3.5, *RNN*s are feedforward neural networks in which the parameters of each layer (both conventional and recurrent) are shared across time steps.
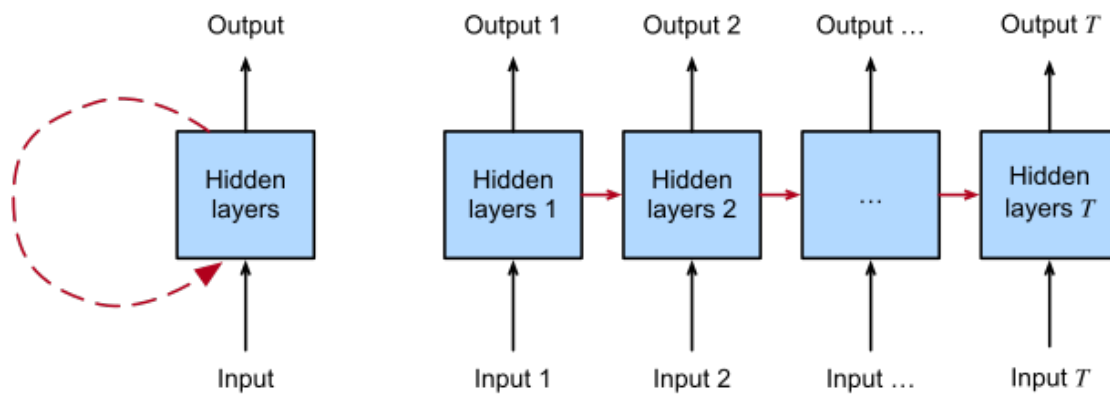

Figure 3.5: Recurrent connections are depicted on the left as cyclic edges. The RNN is unfolded over time steps on the right. Recurrent edges are computed synchronously, while conventional connections span adjacent time steps. [75]

3.2.6 Bidirectional Long Short-Term Memory

The most popular designs include mechanisms to mitigate *RNN*s' infamous numerical instability, as exemplified by vanishing and exploding gradients. We present the key concepts underlying the most successful *RNN* architectures for sequence, which are based on two papers published in 1997.

Long-Short Term Memory (LSTM) [27] is the first paper to introduce the memory cell, a unit of computation that replaces traditional nodes in a network's hidden layer. With these memory cells, networks can overcome training difficulties encountered by previous recurrent networks. To avoid the vanishing gradient problem, the memory cell keeps values in each memory cell's internal state cascading along a recurrent edge with weight 1 across many

successive time steps. A set of multiplicative gates assists the network in determining which inputs to allow into the memory state and when the memory state's content should influence the model's output. Given memory cell c_t, input gate i_t, forget gate f_t, output gate o_t associated with weight matrices W_j, U_j and weight vector b_j where $j \in \{i, f, o, c\}$, *LSTM* is described as:

$$
\begin{aligned}
i_t &= \mathrm{sigmoid}_g(W_i x_t + U_i h_{t-1} + b_i) \\
f_t &= \mathrm{sigmoid}_g(W_f x_t + U_f h_{t-1} + b_f) \\
o_t &= \mathrm{sigmoid}_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \mathrm{sigmoid}_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \odot \mathrm{sigmoid}_h(c_t)
\end{aligned}
\tag{3.14}
$$

The second paper, Bidirectional *Recurrent Neural Network (RNN)* [63], describes an architecture that uses information from both the future (subsequent time steps) and the past (preceding time steps) to determine the output at any point in the sequence. This is in contrast to previous networks, in which only previous input could influence output. Bidirectional *RNN*s have become a mainstay in audio sequence labeling tasks, among many others. Fortunately, the two innovations are not mutually exclusive and have been successfully combined for phoneme classification and handwriting recognition.

3.2.7 Transformer

The Transformer employs the encoder-decoder architecture, as shown in the left and right halves of Figure 3.6, with stacked self-attention and point-wise, fully connected layers for both the encoder and decoder.

The encoder is built up from N identical layers. Each layer is divided into two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, fully connected feed-forward network that is positionally connected. Following layer normalization [5], a residual connection [24] is used around each of the two sub-layers.

Attention: A query and a set of key-value pairs are mapped to an output by an attention function, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, with the weight assigned to each value determined by the query's compatibility function with the corresponding key.
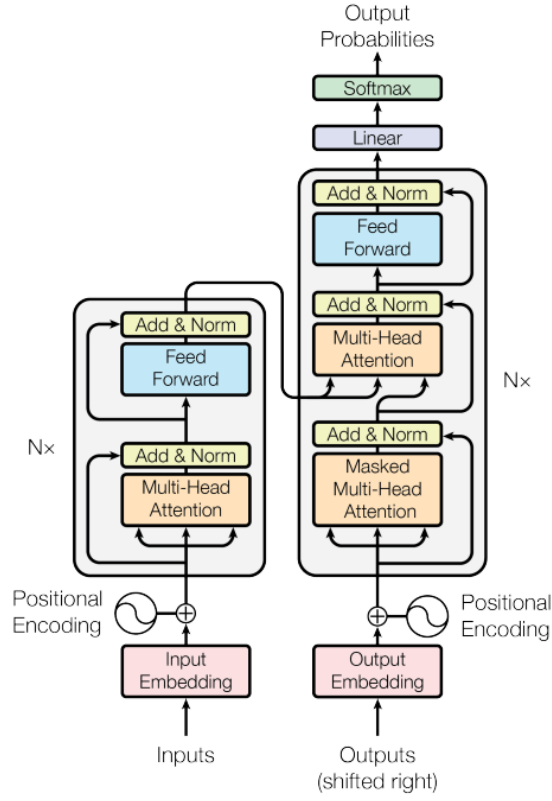
Figure 3.6: The Transformer model architecture [70]

Scaled Dot-Product Attention: The input consists of queries and keys of dimension d_k, and values of dimension d_v. The query's dot products are computed with all keys, divided by $\sqrt{d_k}$, and a softmax function is applied to get the weights on the values. In practice, we compute the attention function on a set of queries at the same time, which we pack into a matrix Q. The keys and values are also packed into matrices K and V. We compute the output matrix as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.15}$$

Multi-Head Attention: Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we perform the attention function in parallel on each of the projected versions of queries, keys, and values, yielding d_v-dimensional output values. These are concatenated and projected again, yielding the final values:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{3.16}$$

where: $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

and the projections are parameter matrices $W_i^Q \in R^{d_{\text{model}} \times d_k}$, $W_i^K \in R^{d_{\text{model}} \times d_k}$, $W_i^V \in R^{d_{\text{model}} \times d_v}$ and $W^O \in R^{h d_v \times d_{\text{model}}}$,

h is the number of attention heads.

3.3 Semi-supervised learning

Semi-supervised learning is a method of machine learning in which a small amount of labeled data is combined with a large amount of unlabeled data during training. Semi-supervised learning is intermediate between unsupervised (no labeled training data) and supervised learning (with only labeled training data). It is an example of weak supervision.

When combined with a small amount of labeled data, unlabeled data can significantly improve learning accuracy. Acquiring labeled data for a learning problem frequently necessitates the use of a skilled human agent (e.g., to transcribe an audio segment in *ASR* tasks). The cost of labeling may thus make large, fully labeled training sets unfeasible, whereas acquiring unlabeled data is relatively inexpensive. Semi-supervised learning can be extremely useful in such situations.

3.3.1 Wav2vec 2.0

Due to self-supervised training, *wav2vec 2.0* is one of the current *SOTA* models for *ASR*. This is a relatively novel concept in this sector. We can pre-train a model on unlabeled data, which is always more accessible, using this method of training. The model can then be fine-tuned for a specific purpose using a specific dataset.

The model consists of a multi-layer convolutional feature encoder $f : X \rightarrow Z$ that receives raw audio X as input and produces *Latent Speech Representations* $z_1, ..., z_T$ for T time steps. They are then supplied into a *Transformer* $g : Z \rightarrow C$, which generates representations $c_1, ..., c_T$ that capture data from the full sequence. In the self-supervised objective, the output of the feature encoder is discretized to q_t using a quantization module $Z \rightarrow Q$ to represent the objectives (Figure 3.7). The approach constructs context representations over continuous speech representations, and self-attention captures dependencies throughout the whole sequence of latent representations.

Feature encoder: The encoder is made up of many blocks that include temporal convolution, layer normalization [5], and the *GELU* activation function [26]. The encoder's raw waveform input is normalized to zero mean and unit variance. The number of time-steps T that are input to the *Transformer* is determined by the encoder's total stride.

Contextualized representations with Transformers: The feature encoder's output is sent into a context network that uses the *Transformer* architecture [70]. We utilize a convolutional layer that acts as a relative positional embedding instead of fixed positional embeddings that
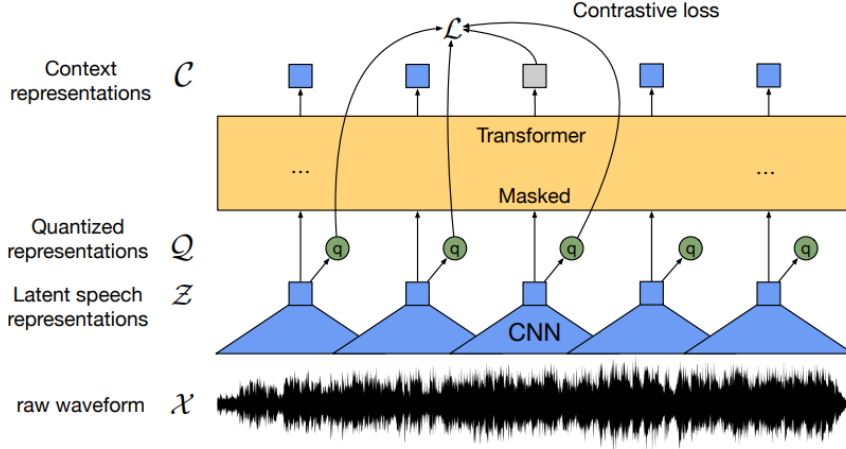
Figure 3.7: Illustration of our framework which jointly learns contextualized speech representations and an inventory of discretized speech units.

encode absolute positional information. We implement layer normalization after adding the convolution output followed by a *GELU* to the inputs.

Contrastive learning: Contrastive learning is a notion that involves the input being altered in two ways. The model is then trained to recognize whether two input transformations are still the same item. The *Transformer* layers are the first method of transformation in *wav2vec 2.0*; the second is quantization. In more technical terms, we would like to get such a context representation c_t for a masked latent representation z_t in order to guess the proper quantized representation q_t among alternative quantized representations.

Quantization module: Quantization is a process of converting values from a continuous space into a finite set of values in a discrete space [67]. A language's number of phonemes is limited. Furthermore, the number of posible phoneme pairs is limited. It means that the same *Latent Speech Representations* can correctly represent both of them. Furthermore, because the quantity is limited, we can design a codebook that contains all potential phoneme combinations. The quantization process then involves selecting the appropriate code word from the codebook. However,the total number of conceivable sounds is enormous. To make it easier to learn and use, we use product quantization [32] to discretize the output of the feature encoder z to a finite set of speech representations for self-supervised training. This choice yielded positive results, which acquired discrete units first and then contextualized representations. Concatenating quantized representations from several codebooks is what product quantization is all about. We take one item from each codebook and concatenate
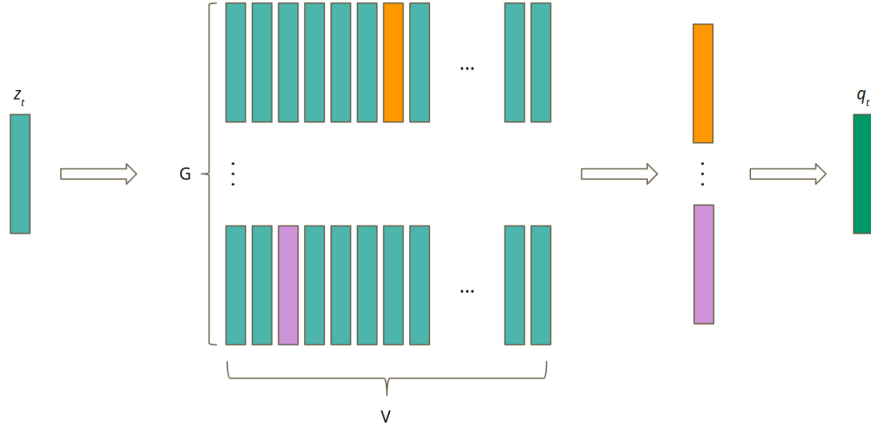
Figure 3.8: Quantization process: For each codebook, the best entry is extracted and concatenated with each other (from orange to purple entry)

the resulting vectors $e_1, ..., e_G$ (Figure 3.8), then perform a linear transformation $R^d \rightarrow R^f$ to get $q \in R^f$, given G codebooks or groups with V entries $e \in R^{V \times d/G}$.

3.3.2 Cross-lingual speech representation

Cross-lingual learning seeks to create models that use data from other languages to improve performance. By pretraining *Transformer* blocks with multilingual masked language models, unsupervised cross-lingual representation learning has shown great success [40, 35]. The authors in [14] studied cross-lingual speech representations by extending *wav2vec 2.0* [6] to the cross-lingual setting. Their method teaches a single set of quantized latent speech representations that are shared by all languages. They pre-trained *XLSR-53* on 56k hours of speech data from 53 languages (including Vietnamese language), then evaluated it on 5 languages from the BABEL benchmark (conversational telephone data) [18] and 10 languages from CommonVoice [3] - a corpus of read speech.

3.3.3 In-domain Match Level and Diversity Level

In this part, to better and easier analyze the effect of pre-training data on the performance of cross-lingual and domain-shift experiments, we introduce 2 new concepts, namely **"In-domain Match Level"** and **"Diversity Level"**.

In-domain Match Level: Given 3 datasets A, B and C, where A is the target telephone dataset used for recognition, B is also recorded by the telephone but its conversation is different from A's and C is the audio book recordings. The dataset B is more overlapped with the A than the C because both A and B are telephone recordings, so the **In-domain Match Level** of B is higher than the one of C. In general, the **In-domain Match Level** is determined by the similarity between recording conditions, naturalness and conversational topics.

"Diversity Level": Given another dataset D, which is recorded by more speakers with more diverse accents than B and C, then the **Diversity Level** of D is the highest compared to the rest. To some extent, the **Diversity Level** of the multilingual dataset is higher than the monolingual one because the first is able to represent more learnable phonemes which are likely to be helpful to target language in semi-supervised learning.

4 Experiments

4.1 Data

The first difficulty faced during the research in the HYKIST project is the lack of medical telephone speech dataset. Having a small medical dataset - HYKIST, we therefore use HYKIST only for the recognition and use in-house non-medical telephone speech dataset for training. This poses a challenge to reach a high-performance ASR because of the mismatch in training and recognition datasets. In addition, real-life dataset like HYKIST is difficult to be accurately transcribed by ASR models because of background noises, variation of speaking speed, unfamiliar pronunciation of medical terms...

4.1.1 HYKIST data

Our HYKIST project partner Triaphon recorded conversations between three people: a patient, a doctor, and an interpreter. The patient communicates in the non-German language - Arabic or Vietnamese - while the doctor communicates in German. The interpreter is fluent in both languages and assists the patient and doctor in communicating. In HYKIST, we have unique accents, foreign-born accents, from both interpreter and patient sides. This directly makes HYKIST more difficult for machines and humans to transcribe, leading understandable bad recognition performance. We received the audio recordings and had our transcribers perform speech transcription within the recordings. We divide the audio data into two sets: dev and test, with no speaker overlap between the two.

The data statistics for the dev and test sets for each individual language can be seen in Table 4.1. We only have a limited amount of data because we create it ourselves. Furthermore, the number of speakers is limited, resulting in a low level of diversity in the testing data. This may result in over-optimization of the evaluation data. To address the impact of the data issues, we obtained additional training data from our industry partner Apptek and other sources.

Language	Dataset	Usage	# Spks	Hours	Domain	In-domain match	Diversity level
Arabic	In-house	pretr.	3379	786	Tel., Conv.	Medium	Medium
German	In-house	pretr.	1723	177	Tel., Conv.	Medium	Medium
Vietnamese	In-house	pretr., finetu.	2240	219	Tel., Conv.	Medium	Medium
	HYKIST	adapt	1	1	Tel., Conv., Med.	High	Low
		dev	3	3			
		test	2	2			
	YouTube	pretr.	-	1.204	Read books	Low	
Multi	In-house*	pretr.	7342	1.182	Tel., Conv.	Medium	High
	XLSR-53	pretr.	-	56.000	Various	Low	

Table 4.1: Data statistics for acoustic data. *The multilingual in-house training dataset is the combination of the Arabic, German and Vietnamese ones listed above. Domain: Telephone (Tel.), Conversational (Conv.), Medical (Med.).

4.1.2 In-house data

AppTek, an industry partner, supplied us with annotated 8kHz conversational telephone speech data. The audio data was collected during telephone conversations between customers and various call centers. Table 4.1 displays the data statistics for the training sets for each of the three languages. We can see that the amount of training data available varies between languages.

We also have speakers with accents and/or dialects for the Arabic and Vietnamese data. For the Arabic data, we have four different datasets with distinct dialects: Syrian, Lebanese, Gulf, and Egyptian. Besides, our Vietnamese dataset has dominantly 2 accents, Northern and Central Vietnamese, and a very small fraction of Southern Vietnamese accent. The speakers with accents in the Vietnamese data are combined into a single dataset.

4.1.3 YouTube

We collected Vietnamese audio data from *YouTube (YT)* under Fair Use Policies[1] in addition to our annotated datasets. The domain in question is purely read speech, such as podcasts, audiobooks, radio stories, or something similar. Pre-processing was done manually by removing non-speech parts such as music and noise, leaving only speech. The audio files were then divided into 10-30 second segments. Table 4.1 displays the data statistics for the

---

[1]https://support.google.com/youtube/answer/9783148

web scraped data. During data collection, we headed to the balance of accents and genders. Therefore, the dataset is divided into Northern and Southern accents, yielding four subsets: Northern Female (518h), Northern Male (213h), Southern Female (290h) and Southern Male (183h).

4.1.4 CommonVoice Vietnamese

We obtain the Vietnamese dataset from the massively-multilingual speech corpus [4]. We use the data version 9.0[2], which includes 17 hours of noisy read speech data recorded by the large number of volunteer speakers. The dataset is split into train/dev/test set. We evaluate our models by directly recognizing on dev and test sets.

4.1.5 VIVOS

VIVOS [44] is a clean Vietnamese read speech corpus consisting of 15 hour recordings. We obtain the dataset[3] split into train/test sets. We evaluate our models by directly recognizing on test set. The test set includes 19 speakers and 48 minutes of duration in total.

4.1.6 Monolingual text data

Apptek, our project partner, provided monolingual text data for all three languages. Text from various sources is included in the data. The number of running words for each language is shown in Table 4.2.

4.1.7 Domain

As shown in Table 4.1 the data spans several domains. The HYKIST project's target domain is medical conversational telephone speech. The training data does not cover this specific domain. This domain mismatch in our data is highlighted. By listening to the audios and comparing them to our target domain, we can determine the in-domain match and diversity level.

---

[2]https://commonvoice.mozilla.org/en/datasets
[3]https://ailab.hcmus.edu.vn/vivos

4.2 Lexicon and language model

# words	vocab	dev		test	
in train	size	OOV	PPL	OOV	PPL
500M	11k	0.1%	67	0.2%	69

Table 4.2: 4-gram LM for Vietnamese.

4.2.1 Lexicon

The Babel project[4] provided us the initial lexicon for the Vietnamese language. The training lexicon is then created by extending the initial lexica with the toolkit Sequitur Grapheme-To-Phoneme[5] [11]. We supplement the lexicon with medical terms provided by our project partner Triaphon in order to decode the HYKIST data. The final recognition lexica for Vietnamese are 11k in size as shown in Table 4.2.

4.2.2 Language model

Language models (LMs) used are 4-grams and use entire words. We create our *LMs* using the training pipeline from the SRILM toolkit [66]. The first step is to create a *LM* for each monolingual text corpus separately. Then, using a weighting procedure, we merge all *LMs* into a single *LM*, producing one *LM* for Vietnamese language. Using the development text, interpolation weights can be determined by giving highest weight to the source language models that have the lowest perplexity on the specified development set.

Table 4.2 demonstrates how the *LM* performs. Vietnamese *LM* achieves a *Perplexity (PPL)* of 67 and a *Out-of-vocabulary (OOV)* rate of 0.1% on dev set.

---

[4]https://www.iarpa.gov/research-programs/babel
[5]https://github.com/sequitur-g2p/sequitur-g2p

4.3 Acoustic model

In this part, our experimental setups for acoustic models are described. We use the toolkit RE-TURNN[6] [15] for supervised training experiments and Fairseq[7] [52] for unsupervised *wav2vec 2.0* training. The recognition is done by using RASR[8] [59]. We convert the Fairseq models to RETURNN models with an automatic conversion toolkit[9]. We will release all training and decoding configurations online[10][11].

4.3.1 Supervised-only models

The training schedule for Vietnamese language's basic systems are similar and simply differ in the specifics. Assuming all models, we generate alignments obtained through the use of a *GMM*/*HMM* procedure will be utilized as labels for neural network training. In a supervised setting using *fCE*, all models are trained from scratch. The labels used in the *AM* modeling are context-dependent phonemes, more specific triphones. With 4501 *CART* labels in the end, we use a *CART* to tie the states. We employ the 40-dimensional Gammatone features as the *AM*'s input [61].

There is no pre-training, so the fine-tuning begins with a random initialization. All the fine-tunings from scratch takes 33 epochs. We use two distinct neural *AM* architectures: *Transformer* [70], and *Bidirectional Long-Short Term Memory (BLSTM)* [28].

BLSTM: We strictly adhere to the training recipe in [46] for the *BLSTM* model. The *BLSTM* uses 5 layers and 512 per-direction units. The following hyperparameters are used for fine-tuning: The initial learning rate is set at 0.0005, followed by a hold phase, and finally an exponential decay with decay factor of 0.8 in order to control the learning rate based on CE development set scores. In addition, we use Adam optimizer with Nesterov momentum (Nadam) [16]. Furthermore, a dropout of 10% is applied to all modules and batch shuffling is turned off. A batch size of 40000 frames is employed. The SpecAugment [54] algorithm is used for entire model training with masking of 50% in the time dimension and 10% in the feature dimension. This leads to the *BLSTM* size of 25M parameters.

---

[6]https://github.com/rwth-i6/returnn
[7]https://github.com/facebookresearch/fairseq
[8]https://github.com/rwth-i6/rasr
[9]https://github.com/rwth-i6/pytorch-to-returnn-converter
[10]https://github.com/rwth-i6/returnn-experiments
[11]https://github.com/rwth-i6/i6-experiments

Transformer: Our *Transformer* training schedule was obtained from [73, 74]. *Transformer* has 12 blocks. The attention dimension of each Multi-Head Self-Attention module is 768 with 12 attention heads. The dimension of the feed-forward module is 1536 with *ReLU* working as an activation function. The following hyperparameters are used for fine-tuning: The initial learning rate is set at 10^{-5} and a linear warm-up phase to 10^{-4} is used, followed by a hold phase, and finally an exponential decay with decay factor of 0.9 until the minimum learning of 10^{-7} is reached. In addition, we use Adam optimizer with Nesterov momentum (Nadam) [16]. Furthermore, a dropout of 10% is applied to all layers of encoder network and we use batch size of 8000 frames. Batches are constructed with shuffled data. The SpecAugment [54] algorithm is used for entire model training with masking of 50% in the time dimension and 10% in the feature dimension. This leads to the *Transformer* size of 90M parameters.

4.3.2 Models using unsupervised pre-training

XLSR-53: We look into using a publically accessible model, *XLSR-53* [14], in addition to pre-training our own models on our specific data. We utilize the checkpoint that was not fine-tuned to any language[12]. This was pre-trained on 56k hours of speech data from 53 different languages for 19 epochs. Additionally, we explore with initializing the *wav2vec 2.0* pre-training on our custom data using the *XLSR-53* model, followed by corresponding fine-tuning. Note that 16kHz data were used to train the *XLSR-53*. We shorten the stride of one *CNN* layer in the feature extractor to half because we work with 8kHz telephone conversation. In this method, we receive features at the desired frame rate while reducing the down-sampling factor from the waveform to the feature frames by a factor of 2.

Pretraining cases: For each pretrained model we divide into the following cases.

1. Instead of a custom pre-training with our available datasets, *XLSR-53* is applied directly for the fine-tuning.

2. Pre-training on our available datasets from scratch.

3. The parameters are initialized with the *XLSR-53* checkpoint and the pre-training is done with our available datasets. We call this type of pre-training continued pretraining.

---

[12]https://github.com/facebookresearch/fairseq/tree/main/examples/wav2vec

Wav2vec 2.0 architectures: We use the topologies from *wav2vec 2.0* for the experiments with unsupervised pre-training [6] and customize our own topologies into: *Base*, *Large* and *Large*$_{1-8}$. All architectures work with the raw audio waveform and have a feature extractor that uses 7 *CNN* layers. However, the *Large* architecture has an encoder stack made up of 24 *Transformer* layers with dimension of the feed-forward module being 1024 and the number of attention heads being 16. *Base* only has 12 *Transformer* layers with dimension of the feed-forward module being 768 and the number of attention heads being 12. *wav2vec 2.0* Large model trained on multilingual data makes up the *XLSR-53* model [14]. The 24 *Transformer* layers employed in the Large architecture place a heavy burden on the GPU's memory. Training times are dramatically increased when GPU memory is traded for a smaller batch size. We suggest discontinuing the *wav2vec 2.0* Large network after the 8th *Transformer* block and referring to the model as *Large*$_{1-8}$in order to mitigate. We discovered that an optimal trade-off between a large enough batch size and a model size that still fits into memory is 8 layers. The cut-off reduces the model size of the full architecture *Large* from 317M parameters to 115M of *Large*$_{1-8}$and is therefore much closer to 95M parameters of *Base* architecture. In addition to the difference between architectures,

Pretraining: During pretraining we employ the proposed hyperparameters in the *XLSR-53* paper [14] but apply the learning rate of *wav2vec 2.0* for the monolingual pre-trainings. The pre-trainings are done for 300 epochs if there is nothing mentioned otherwise. A linear warm-up is used during the first 30 epochs until the learning rate reaches 0.0005 and then a linear decay starts. The mini-batch size in the existing Fairseq implementation is specified in samples of the waveform. For both *Base* and *Large*$_{1-8}$ we use a dropout of 10% in the feature extractor, 5% in the encoder and 10% in the latent representations between the feature extractor and encoder. We do not apply dropout to pre-trainings with the *Large* architecture. a *NN* is pre-trained on unlabeled data using the contrastive loss and diversity loss as described in [6] using the *wav2vec 2.0* framework.

Finetuning: To finetune the acoustic model, we use the training system described in [46] to create a baseline *GMM/HMM* model for Vietnamese language. This model is used to generate alignments of the speech data with the *CART* labels for the *DNN* system. The hybrid model's *NN* is trained on these alignments in a supervised manner using the *Frame-wise Cross-entropy (fCE)* loss. An application of a two-stage training configuration is made when using unsupervised pre-training. After pretraining, the *NN* is then fine-tuned by adding a softmax output layer, initializing with a checkpoint from pre-training, training with the *fCE* loss on labeled data, and using the same alignment as in the fully supervised scenario. The following hyperparameters are used for fine-tuning: The initial learning rate is set to 10^{-5}

and uses a linear warm-up phase to 10^{-4} followed by a hold phase and afterwards ends with exponential decay of 0.9. The *wav2vec 2.0* SpecAugment variant introduced in [6] is used with the masking done by choosing independent random starting points in the time/feature dimension and the subsequent 10/64 steps are masked. We employ a mini-batch size of 1875 frames with length of 10ms, leading to the audio of 18.75 seconds. Furthermore, a gradient noise of 10% is used and we also apply a dropout of 5% to all layers of both feature extractor and Transformer encoder network.

In addition to dropout [65], we also investigate the performance of other regularization techniques like the intermediate loss, L2 and On-off Regularization described in 4.3.4, 4.3.5 and 4.3.6

4.3.3 Data augmentation

In this thesis, apart from the use of SpecAugment [54] stated above, we also use other data augmentation techniques in the pretraining stage.

The augmentation was exclusively done using the speed perturbation {90%, 110%, 115%} [38], random pitch perturbation {-350:-250; 250:350} [12] and reverberation perturbation [56]. We did not go further to analyze if these augmentation options are the most optimal.

4.3.4 Intermediate loss

Our intermediate loss setups are based on [68, 73]. Besides, we have 2 variants of intermediate loss, namely *Intermediate Cross-Entropy Loss (ICE Loss)*, which uses *Cross-entropy (CE)* loss and *Intermediate Focal Loss (IF Loss)*, which replaces*Cross-entropy (CE)* loss with focal loss [43].

ICE Loss: We conducted multiple experiments with intermediate loss scales ranging in {0.1, 0.2, 0.3, 0.4, 0.5} and dropout [65] values ranging in {0.05, 0.1}. We saw that the combination of loss scale 0.3 and dropout value 0.1 yielded the best results for all pretrained models and architectures, so we take this as default for all next experiments.

IF Loss: We experimented with 3 ways of integrating focal loss into the vanilla intermediate loss setup: only in the network *CE* output layer, only in the intermediate loss layer and in both of them. We found that putting the focal loss in both 2 positions yielded better result. To find a good focal loss value, we conducted experiments with multiple values in {1.5, 2.0, 2.5, 3.0}. The higher the value is, the more on labels the network is forced to

"focus". We saw that the 2 values {1.5, 2.0} did not make difference in results, while for higher focal values {2.5, 3.0}, the model gained more benefits on in-house training set but hurt the performance on out-domain recognition test sets. We highly recommend the use of focal value 2.0 so that the model generalizes on all different test sets.

4.3.5 L2 regularization

To find good values of L2 regularization [39], we used grid-search technique. We tested the value ranging in {0.01, 0.005, 0.001, 0.0005, 0.0001} to see the resulting *Word-error-rate (WER)*. Each pretraining model and architecture has its own unique L2 value to work best. We put L2 regularization at all linear layers in the network.

4.3.6 On-off Regularization

To further improve the accuracy performance of *IF Loss*, we introduce a new regularization technique called "On-off Regularization technique". We turn off all regularizations (Dropout, SpecAugment and *IF Loss*) in the first stage of training (3-10 first epochs). We call this stage "Off Regularization". We then reset the learning rate and turn all regularizations back on in the second stage of training, which we call "On Regularization". The second stage of training ends when the model is fully converged.

5 Experimental results

5.1 Supervised baselines

AM	WER [%]	
	Hykist dev	Hykist test
GMM	62.2	59.7
BLSTM	32.9	38.4
Transformer	31.0	35.1

Table 5.1: *WERs* [%] for supervised-only baselines on Vietnamese HYKIST data [45]. Models are trained on the monolingual in-house data. The labels are context-dependent phonemes (*CART* state tying) with their size being 4501.

The baseline for Vietnamese is trained using the relevant in-house 8kHz monolingual telephone speech data. The performance of the baseline *ASR* systems is displayed in Table 5.1. The intrinsic difficulty of the language and the data causes the systems to function differently. We believe there are various causes for this. Due to the natural flow of speakers, the Vietnamese transcriptions are hard to reach high quality. Additionally, Vietnamese also incorporates accented speech which is even difficult for native speakers to fully understand. Furthermore, the accent mismatch between Vietnamese fine-tuning and recognition data is also a major factor to the degradation of performance. Our Vietnamese in-house dataset has dominantly 2 native accents, Northern and Central Vietnamese, and a very small fraction of Southern Vietnamese native accent, while HYKIST, because of being a simulation dataset, has unique accents - foreign-born accents - from both interpreter and patient sides.

On the HYKIST data, switching from a *GMM*/*HMM* framework to a hybrid *HMM* framework with a *RNN-BLSTM* results in a reduction of *WER* from 62.2% and 59.7% to 32.9% and 38.4% on dev and test set respectively. Besides, the *WER*s continue decreasing to 31.0% and 35.1% by replacing *BLSTM* with *Transformer* encoder.

5.2 Unsupervised Pre-training

5.2.1 Monolingual pre-training

Table 5.2 shows the outcomes from models pretrained on monolingual data. The number of pre-training epochs is decided upon using the best downstream *WER* on Vietnamese.

Pre-training		Fine-tuning	WER [%]	
Data (hours)	Epochs	Epochs	Hykist dev	Hykist test
None	None	33	32.1	36.6
Viet. in-house (219h)	100	26	31.4	33.4
Aug. Viet. in-house (1168h)	300		31.0	32.3
Viet. YT (1168h)			29.8	35.2
Viet. in-house + YT (1168h)			25.3	27.2

Table 5.2: *WERs* [%] for models pretrained on monolingual data. All fine-tunings use the *Large*$_{1-8}$ architecture and are trained until full convergence on Vietnamese in-house data and the recognition is done on HYKIST. All pre-trainings have been done with random initialization. Pre-training data "None" in the 3rd row means fine-tuning from scratch with *wav2vec 2.0 Large*$_{1-8}$ architecture.

Even though no additional data is included for pre-training here, pre-training on the monolingual in-house data for Vietnamese reveals a reduction of *WER*s from 32.1% and 36.6% to 31.4% and 33.4% on dev and test set respectively. This proves that on *wav2vec 2.0* architecture, the unsupervised pretraining helps the *WER* performance.

Next, when we pretrain with the augmented in-house data, we achieve a small improvement to 31.0% and 32.3% on dev and test set respectively. This shows that data augmentation for pretraining is helpful.

We then examine the impact of pre-training on the *YouTube (YT)* data for Vietnamese, which results improvements to 29.8% and 35.2%. Although *YT* data is much more than the in-house data (1168h compared to 219h), both results seem to similar in terms of the average result on dev and test set. This proves that having more data is not always helpful, because of 2 reasons. The first reason is that the domain of the in-house data is closer to that of HYKIST (both of them are telephone domain), while the domain mismatch between

YT and HYKIST is larger (read speech compared to telephone speech). Another reason is that *YT* data has less speakers, leading to worse generalization while pretraining.

The greatest significant improvement is achieved by combining the in-house and *YT* data leading to a reduction of *WER*s to 25.3% and 27.2%. This is the best result produced using solely monolingual data. Because we substitute 200 hours of *YT* with in-house data, the amount of pre-training data used here is comparable to that of only *YT* pre-training. This result proves that a diversity of domains and speakers in the pretraining stage is necessary for better performance on test sets.

5.2.2 Multilingual pre-training

Pre-training			WER [%]	
Init	Data (hours)	Epochs	Hykist dev	Hykist test
random	Viet. in-house + YT (1168h)	300	25.3	27.2
	Multilingual in-house (1168h)		26.8	28.7
$XLSR$-$53_{1\text{-}8}$	None	None	27.6	31.9

Table 5.3: *WERs* [%] for models using unsupervised pretraining on multilingual data compared to pretraining on monolingual data. All fine-tunings use the *Large*$_{1\text{-}8}$ architecture and are trained until full convergence on Vietnamese in-house data and the recognition is done on HYKIST. The 2nd model is pretrained on our multilingual in-house dataset, and the 3rd uses *XLSR*-$53_{1\text{-}8}$ to directly finetune on Vietnamese in-house data.

We then examine models that have already been multilingually pre-trained in Table 5.3. For Vietnamese dev/test, combining the Arabic, German, and Vietnamese in-house data to create a custom multilingual pre-training significantly outperforms the non-pretraining baseline, at *WER*s of 26.8% and 28.7% on dev and test set respectively. However, the monolingual combination of in-house and *YT* data is still better for Vietnamese, at 25.3% and 27.2% on dev and test set respectively. These results reject [14]'s conclusion where multilingual pretraining is proved to outperform monolingual pretraining.

Strong increases of *WER*s can also be seen by fine-tuning only utilizing the *XLSR*-$53_{1\text{-}8}$ checkpoint. With the exception of the Vietnamese test set, where it is up to 11% worse, it performs only relatively worse than the custom pre-training on the multilingual in-house data, at *WER*s of 27.6% and 31.9%. This may be due to the absence of 8kHz data in the

pre-training of *XLSR-53*. Nevertheless, adopting it in a fast and simple manner can result in considerable benefits.

5.2.3 *XLSR-53* as pre-training initialization

Pre-training			WER [%]	
Architecture	Data (hours)	Epochs	Hykist dev	Hykist test
Large$_{1-8}$	None	None	27.6	31.9
	Viet. in-house	25	27.6	29.5
Large	(219h)	100	26.2	29.0
Large$_{1-8}$	Viet. YT (1168h)	100	24.3	28.1
	Viet. in-house + YT (1168h)		24.5	27.2
	Multilingual in-house (1168h)	50	23.9	27.4

Table 5.4: *WERs* [%] for models using unsupervised pre-training with the public *XLSR-53* model as initialization (3rd row is for direct finetuning and the rest are initialization for pretraining on specific data). All fine-tunings use the *Large*$_{1-8}$ architecture and are trained until full convergence on Vietnamese in-house data and the recognition is done on HYKIST. The 1st model is the direct finetuning on Vietnamese in-house data, and the remaining models use *XLSR-53* as initialization for pretrainings (full model *Large* or cut-off model *Large*$_{1-8}$).

As an alternative, we might use *XLSR-53*$_{1-8}$ as an initialization for a customized pre-training, as shown in Table 5.4. On the in-house Vietnamese data, the *WER*s reduce from 31.4% and 33.4% (Table 5.2) to 27.6% and 29.5% on dev and test set respectively, compared to 27.6% and 31.9% of direct finetuning with *XLSR-53*$_{1-8}$. This proves that continued pretraining using *XLSR-53* model outperforms the pretraining using random initialization and the direct finetuning using *XLSR-53*.

A *Large* model initialized with *XLSR-53* is also pre-trained on the monolingual in-house data before being reduced to a smaller size for fine-tuning. This performs better than pre-training with the smaller *Large*$_{1-8}$ (26.2% and 29.0% compared to 27.6% and 29.5% on dev and test set respectively), but at the expense of increased pre-training's resource usage. Therefore, if the resource usage is neglected, the *Large* model should be chosen for better *WER*.

For the pretraining on the *YT* data using *XLSR-53* as initialization, the *WER*s reduce from 29.8% and 35.2% (Table 5.2) to 24.3% and 28.1% on dev and test set respectively. The

benefits of integrating *XLSR-53* into the multilingual data are substantially lower, with *WER*s being reduced from 26.8% and 28.7% (Table 5.3) to 23.9% and 27.4%. On the domain-diverse dataset (the combination of monolingual in-house and *YT* data), the benefits of continued pretraining are also reduced, with *WER*s being reduced from 25.3% and 27.2% (Table 5.2) to 24.5% and 27.2% on dev and test set respectively. This shows that the continued pretraining is beneficial for both the monolingual and the multilingual scenario. However, the continued pretraining on less diverse data benefits more from the diverse and multilingual data.

5.2.4 Comparison to supervised baselines

AM	Init	Pre-training	WER [%]	
		Data (hours)	Hykist dev	Hykist test
Transformer	random	None	31.0	35.1
wav2vec 2.0	random	None	32.1	36.6
		Viet. in-house (219h)	31.4	33.4
		Viet. YT (1168h)	29.8	35.2
	$XLSR\text{-}53_{1\text{-}8}$	Viet. in-house + YT (1168h)	24.5	27.2
		Multilingual in-house (1168h)	23.9	27.4

Table 5.5: *WERs* [%] for models using unsupervised pre-training and supervised-only training. All fine-tunings use the $Large_{1\text{-}8}$ architecture and are trained until full convergence on Vietnamese in-house data and the recognition is done on HYKIST. The 1st model is the supervised-only training using Transformer. The 2nd and 3rd models are pretrained on specific data using random initializaton. The 4th and 5th models are continued pretraining methods (using $XLSR\text{-}53_{1\text{-}8}$ as initialization).

As shown in Table 5.5, we can see that fine-tuning using *wav2vec 2.0* $Large_{1\text{-}8}$ from scratch is worse when we compare with the findings from the supervised-only baseline (32.1% and 36.6% vs. 31.0% and 35.1% on dev and test set respectively). With monolingual pre-training on the identical data, there is still no apparent advantage (31.4% and 33.4%). When we increase the pretraining data to 5 times with a less diverse data (*YT* data), the performance also does not clearly outperform the supervised-only baseline (29.8% and 35.2%). This proves that the *wav2vec 2.0* unsupervised pretraining does not always outperform the *Transformer* supervised-only approach, especially when the pretrained data is not diverse enough.

However, we are able to significantly outperform the supervised baselines when applying continued pretraining. In comparison to the best supervised-only baseline, the best results for continued pretraining show a reduction of *WERs* to 24.5 % and 27.2% on monolingual data and to 23.9% and 27.4% on multilingual data. Therefore, we can conclude that continued pretraining should be used to gain the most benefits in terms of accuracy.

5.3 Encoder and initialization comparison

5.3.1 Encoder comparison

Architecture	Pretraining Data (hours)	WER [%]	
		Hykist dev	Hykist test
Base	None	35.8	39.9
Large$_{1\text{-}8}$		35.0	40.7
Base	Viet. in-house	30.2	33.3
Large$_{1\text{-}8}$	(219h)	31.5	33.4
Base	Multilingual in-house	26.2	28.8
Large$_{1\text{-}8}$	(1168h)	26.8	28.7

Table 5.6: *WERs* [%] for architecture *Base* and *Large*$_{1\text{-}8}$ using different pretraining schedules: no pretraining, pretraining on in-house data and on multilingual data. All fine-tunings are done until full convergence on Vietnamese in-house data and the recognition is done on HYKIST.

We compare the performance of 2 types of encoder: *Base* and *Large*$_{1\text{-}8}$. As shown in Table 5.6, we receive mix results for various pretraining schedules: no pretraining, pretraining on in-house data and pretraining on multilingual data. It is mentioned by [31] in language modeling that the *Base* architecture works better than the *Large*. However, in acoustic modeling in *ASR*, our results prove against this statement. Considering the amount of parameters between *Base* and *Large*$_{1\text{-}8}$, 97M vs. 118M, we recommend the use of *Base* in order to keep the performance competitive to *Large*$_{1\text{-}8}$ while reducing the number of trainable parameters.

5.3.2 Initialization comparison

In the case of super short pretraining (1 epoch pretraining on only 0.01h of data), the results outperform those of raw waveform from scratch for both *Base* and *Large*$_{1\text{-}8}$ architecture as shown in Table 5.7. The reason for the improvement comes from the difference of initialization schemes. The parameters from the pretrained model are first initialized by Fairseq [52] using Kaiming Initialization [25], and then fed into RETURNN [15], while the parameters for raw waveform training are initialized directly by RETURNN using Glorot (also known as Xavier) Initialization [20]. We therefore recommend the use of Kaiming Initialization for *wav2vec 2.0* architecture.

Architecture	Init. scheme	Pretraining		WER [%]	
		Data (hours)	Epochs	Hykist dev	Hykist test
Base	Kaiming Init.	Viet. in-house	1	30.6	35.2
Large$_{1-8}$	(Fairseq)	(0.01h)		31.8	35.7
Base	Glorot Init.	None	None	35.8	39.9
Large$_{1-8}$	(RETURNN)			35.0	40.7

Table 5.7: *WERs* [%] for architecture *Base* and *Large*$_{1-8}$ using 2 different initialization schemes: Kaiming Initialization (in Fairseq framework [52]) and Glorot Initialization (in RETURNN framework [15]). All fine-tunings are done until full convergence on Vietnamese in-house data and the recognition is done on HYKIST.

5.4 Effectiveness of intermediate loss

5.4.1 Effectiveness of Intermediate Cross-Entropy Loss

Improvement on HYKIST data: In Table 5.8, when using in-house telephone dataset to train and transcribe the HYKIST dataset with the help of *ICE Loss*, we report the total improvement in performance for from scratch experiment where the *WERs* decrease from 35.6% and 40.7% to 33.8% and 38.1% on dev and test set respectively. For *YT* experiment, the *WERs* decrease from 29.8% and 35.2% to 27.3% and 31.5%. We also report a small improvement for the combination of Vietnamese in-house data and *YT* data, from 25.3% and 27.2% to 25.1% and 27.1%.

Pre-training data	With ICE	WER [%]	
		Hykist dev	Hykist test
None	No	35.6	40.7
	Yes	33.8	38.1
Viet. YT (1168h)	No	29.8	35.2
	Yes	27.3	31.5
Viet. in-house + YT (1168h)	No	25.3	27.2
	Yes	25.1	27.1

Table 5.8: Improvements of *WERs* [%] on HYKIST data between pretraining schedules when applying *ICE Loss*. All models are finetuned until full convergence on Vietnamese in-house data. Only 1 intermediate layer is applied in the middle *Transformer* block, e.g. position 4 for $Large_{1-8}$ and 6 for *Base* architecture.

Degradation on HYKIST data: As shown in Table 5.9, for the directly finetuning experiment with *XLSR-53* preloaded, the performance is hurt totally (both *WERs* on dev and test sets increase). Besides, both continued pretrainings on Vietnamese in-house and on *YT* data experience the partial improvements (only *WERs* on test sets are slightly increased but *WERs* on dev sets decrease). The rest pretraining schedules in Table 5.9 also experience partial improvements.

Improvement on CommonVoice and VIVOS data: In the situation of more out-of-domain recognition shown in Table 5.10, which means using the model finetuned on our in-house spontaneous telephone speech dataset to do the recognition on read speech datasets like CommonVoice and VIVOS, we report the total improvements in performance for $Large_{1-8}$ in-

Arch.	Init.	Pre-training data	With ICE	WER [%]	
				Hykist dev	Hykist test
Large$_{1-8}$	*XLSR-53*	None	No	27.9	32.3
			Yes	28.4	33.3
	None	Viet. in-house (219h)	No	30.4	33.4
			Yes	29.1	33.7
	XLSR-53		No	25.5	29.1
			Yes	25.2	29.2
Base	None		No	30.2	33.3
			Yes	29.7	33.4
Large$_{1-8}$		Multiling. in-house (1168h)	No	26.8	28.7
			Yes	25.5	29.4
	XLSR-53	Viet. YT (1168h)	No	24.3	28.1
			Yes	23.7	28.2

Table 5.9: Degradations of *WERs* [%] on HYKIST data between pretraining schedules when applying *ICE Loss*. All models are finetuned until full convergence on Vietnamese in-house data. Only 1 intermediate layer is applied in the middle *Transformer* block, e.g. position 4 for *Large*$_{1-8}$ and 6 for *Base* architecture.

house pretraining, from scratch and *YT* experiments. Notable is from scratch training where *WER*s reduce from 20.8%, 44.7%, 34.9% to 18.6%, 42.1%, 33.1%; and *YT* pretraining where *WER*s reduce from 16.4%, 34.4%, 28.7% to 15.6%, 32.2%, 27.6% on CommonVoice dev/test and VIVOS test set respectively. Together with the improvements on HYKIST reported in Table 5.8, we conclude that using *ICE Loss* for from scratch training and for pretraining on *YT* data improves the recognitions on both telephone and read speech domain.

Pre-training data	With ICE	WER [%]		
		CV dev	CV test	Vivos
None	No	20.8	44.7	34.9
	Yes	18.6	42.1	33.1
Viet. YT (1168h)	No	16.4	34.4	28.7
	Yes	15.6	32.2	27.6
Viet. in-house (219h)	No	16.4	35.6	31.3
	Yes	16.1	34.8	30.4

Table 5.10: Improvements of *WERs* [%] on CommonVoice and VIVOS between pretraining schedules when applying *ICE Loss*. All models are finetuned on Vietnamese in-house data. Only 1 intermediate layer is applied in the middle *Transformer* block, e.g. position 4 for *Large*$_{1-8}$ and 6 for *Base* architecture.

Degradation on CommonVoice and VIVOS data: As shown in the Table 5.11, we experience the total degradations for 2 cases: continued pretraining on Vietnamese in-house data and pretraining on the combination of in-house and *YT* data; where *WER*s for all read speech test sets increase. The rest cases experience partial degradations.

Arch.	Init.	Pre-training data	With ICE	WER [%]		
				CV dev	CV test	Vivos
$Large_{1\text{-}8}$	*XLSR-53*	None	No	14.8	32.5	30.3
			Yes	15.8	33.9	30.0
		Viet. in-house (219h)	No	11.5	29.4	27.2
			Yes	12.3	29.8	27.7
Base			No	16.6	35.4	30.9
			Yes	15.4	34.2	31.3
$Large_{1\text{-}8}$	None	Multiling. in-house (1168h)	No	15.2	29.7	29.5
			Yes	14.8	30.5	28.8
		Viet. in-house + YT (1168h)	No	12.9	26.5	21.0
			Yes	13.6	28.2	21.9
	XLSR-53	Viet. YT (1168h)	No	11.8	28.4	25.6
			Yes	12.3	28.3	25.0

Table 5.11: Degradations of *WER*s [%] on CommonVoice and VIVOS between pretraining schedules when applying *ICE Loss*. All models are finetuned on Vietnamese in-house data. Only 1 intermediate layer is applied in the middle *Transformer* block, e.g. position 4 for $Large_{1\text{-}8}$ and 6 for *Base* architecture.

5.4.2 Effectiveness of Intermediate Focal Loss

Effectiveness on HYKIST data:

As shown in Table 5.12 and Table 5.13 below, when using *IF Loss*, we see the *WER*s on HYKIST improved compared to the baselines for various pretraining schedules (7/9 experiments experience total improvements), compared to only 3/9 experiments experiencing total improvements using *ICE Loss* (*ICE Loss* results are shown in Table 5.8 and Table 5.9 above). In addition, we report all *WER*s of *IF Loss* experiments to be lower than those of *ICE Loss* experiments, except the one on HYKIST test set of from scratch training. We therefore conclude that, when finetuning and recognizing on the same telephone domain, *IF Loss* works better than *ICE Loss*.

Compared to our strongest continued pretraining baseline, the application of *IF Loss* on the combination of Vietnamese in-house and *YT* data (24.5% and 27.1%) outperforms the results of continued pretraining on the combination of Vietnamese in-house and *YT* data (24.5% and 27.2% on dev and test set respectively as shown in Table 5.4). However, we believe that the *IF Loss* can further reduce *WER*s for this continued pretraining schedule, as it does with continued pretraining on *YT* data.

Among all total improvements reported in Table 5.12, notable is the *WER*s reduction of *YT* experiment from 29.8% and 35.2% to 26.1% and 30.8% on dev and test set respectively, whose relative *WERR* is around 12.5% in average. For a more diverse pretrained data (Vietnamese in-house data), we report the *WER*s reduction from 30.4% and 33.4% to 28.6% and 33.0%, whose relative *WERR* is around 3.6% in average. For even more diverse pretrained data (Vietnamese in-house + *YT* data), we report the *WER*s reduction from 25.3% and 27.2% to 24.5% and 27.1%, whose relative *WERR* is around 1.8% in average. We therefore conclude that the effectiveness of *IF Loss* decreases when the pretrained data becomes more diverse.

Arch.	Init.	Pre-training data	With IF	WER [%]	
				Hykist dev	Hykist test
$Large_{1\text{-}8}$	None	None	No	35.6	40.7
			Yes	33.0	38.8
		Viet. in-house (219h)	No	30.4	33.4
			Yes	28.6	33.0
	XLSR-53		No	25.5	29.1
			Yes	24.7	29.1
Base			No	30.2	33.3
			Yes	29.0	33.0
$Large_{1\text{-}8}$	None	Viet. YT (1168h)	No	29.8	35.2
			Yes	26.1	30.8
		Viet. in-house + YT (1168h)	No	25.3	27.2
			Yes	24.5	27.1
	XLSR-53	Viet. YT (1168h)	No	24.3	28.1
			Yes	23.4	28.1

Table 5.12: Improvements of *WER*s [%] on HYKIST data between pretraining schedules when applying *IF Loss*. All models are finetuned until full convergence on Vietnamese in-house data. Only 1 intermediate layer is applied in the middle *Transformer* block, e.g. position 4 for $Large_{1\text{-}8}$ and 6 for *Base* architecture.

As shown in Table 5.13, only for the case of directly finetuning with *XLSR-53*, using *IF Loss*

makes the *WERs* on HYKIST increased compared to the baselines. However, the degradation is rather small, from 27.9% and 32.3% to 28.0% and 32.8% on dev and test set respectively. Besides, a partial degradation of performance is reported in the multilingual in-house data experiment, where the average *WER* of dev and test set (25.2% and 29.3%) is even lower than the baseline (26.8% and 28.7%). Hence, in a rapid deployment of an *ASR* system, we recommend the direct use of *IF Loss* in training without the need of one more training as a baseline for performance comparison.

Init.	Pre-training data	With IF	WER [%]	
			Hykist dev	Hykist test
XLSR-53	None	No	27.9	32.3
		Yes	28.0	32.8
None	Multiling. in-house (1168h)	No	26.8	28.7
		Yes	25.2	29.3

Table 5.13: Degradations of *WERs* [%] on HYKIST data between pretraining schedules when applying *IF Loss*. All models are finetuned until full convergence on Vietnamese in-house data. Only 1 intermediate layer is applied in the middle *Transformer* block, e.g. position 4 for *Large*$_{1\text{-}8}$ and 6 for *Base* architecture.

Effectiveness on CommonVoice and VIVOS data:

In the larger domain-shift recognition, we still receive the significant reduction of *WERs* in multiple experiments as shown in Table 5.14. The notable reduction of *WERs* compared to baselines is again on *YT* data, whose *WERs*s decrease from 16.4%, 34.4% and 28.7% to 14.5%, 30.9% and 26.9% respectively for 3 read speech sets, that makes *WERR* about 9.3% in average. The *ICE Loss* in Table 5.10 makes 3 experiments totally improved, while the *IF Loss* makes 4. Furthermore, the *WER*s for *IF Loss* on 3 read speech datasets are as competitive as *ICE Loss*. In addition, when finetuning and recognizing on the same telephone domain, *IF Loss* works better than *ICE Loss* as proved above. We therefore conclude that *IF Loss* works better than *ICE Loss* in all domains.

However, in the larger domain-shift recognition, we still meet degradations of performance in experiments pretrained on diverse data, as shown in Table 5.15. We therefore recommend the use of *IF Loss* only for less diverse pretrained data if the domain of finetuning and recognition data are too different.

Arch.	Pre-training data	With IF	WER [%]		
			CV dev	CV test	Vivos
$Large_{1\text{-}8}$	Viet. in-house (219h)	No	16.4	35.6	31.3
		Yes	15.8	34.5	29.6
	None	No	20.8	44.7	34.9
		Yes	19.7	43.1	33.9
$Base$	Viet. in-house (219h)	No	16.6	35.4	30.9
		Yes	15.9	34.4	30.5
$Large_{1\text{-}8}$	Viet. YT (1168h)	No	16.4	34.4	28.7
		Yes	14.5	30.9	26.9

Table 5.14: Improvements of *WERs* [%] on CommonVoice and VIVOS between pretraining schedules when applying *IF Loss*. All models are finetuned on Vietnamese in-house data. Only 1 intermediate layer is applied in the middle *Transformer* block, e.g. position 4 for $Large_{1\text{-}8}$ and 6 for *Base* architecture.

Init.	Pre-training data	With IF	WER [%]		
			CV dev	CV test	Vivos
$XLSR\text{-}53$	None	No	14.8	32.5	30.3
		Yes	15.4	33.6	30.0
	Viet. in-house (219h)	No	11.5	29.4	27.2
		Yes	13.0	29.8	27.6
None	Multiling. in-house (1168h)	No	15.2	29.7	29.5
		Yes	14.5	30.6	28.1
	Viet. in-house + YT (1168h)	No	12.9	26.5	21.0
		Yes	12.7	28.6	22.1
$XLSR\text{-}53$	Viet. YT (1168h)	No	11.8	28.4	25.6
		Yes	13.2	29.1	24.5

Table 5.15: Degradations of *WERs* [%] on CommonVoice and VIVOS between pretraining schedules when applying *IF Loss*. All models are finetuned on Vietnamese in-house data. Only 1 intermediate layer is applied in the middle *Transformer* block, e.g. position 4 for $Large_{1\text{-}8}$ and 6 for *Base* architecture.

5.5 Intermediate loss analysis

5.5.1 Studies on Intermediate Focal Loss design

Viet. in-house $Large_{1\text{-}8}$					
Layer	Hykist dev	Hykist test	CV dev	CV test	Vivos
None	30.4	33.4	16.4	35.6	31.3
2	29.4	34.0	**15.5**	35.7	30.0
4	**28.6**	**33.0**	15.8	**34.5**	**29.6**
6	29.1	**33.0**	16.6	34.1	29.9
2,6	29.1	34.1	15.6	35.5	30.7
3,5	29.1	33.3	16.4	35.0	30.1
Viet. in-house *Base*					
None	30.2	33.3	16.6	35.4	30.9
3	**28.7**	33.4	17.0	35.5	**30.1**
6	29.0	33.0	**15.9**	**34.4**	30.5
9	29.5	**32.6**	14.8	34.4	30.5
4,8	29.3	33.5	16.2	35.0	30.3

Table 5.16: *WERs* [%] comparison of *IF Loss* on different layers between 2 architecture sizes: *Base* and $Large_{1\text{-}8}$. All models are finetuned until full convergence on Vietnamese in-house data and recognized on HYKIST, CommonVoice and VIVOS dataset. Layer "None" means the baseline (no application of *IF Loss*).

In Table 5.16, we study variants of putting *IF Loss* at different layers. For $Large_{1\text{-}8}$ model, we observe the performance degradation when moving the single *IF Loss* to different layers, while this gives mix results for the *Base* model. [41] also reports the same behavior when using Intermediate CTC Loss on 12-layer, 24-layer and 48-layer models in a supervised-only scenario. When applying 2 intermediate layers, we meet the degradation of performance for both *Base* and $Large_{1\text{-}8}$ models. From the experimental results, we therefore conclude that: Single *IF Loss* in the middle network layer yields the best result among variants.

5.5.2 On-off Regularization technique

To better exploit the *IF Loss*, we introduce the **"On-off Regularization technique"**. We experiment this technique for raw waveform from scratch training. Experimental results in Table 5.17 show that, if we train without any regularization techniques ("Off Regularization"

Off reg.				On reg.		
Epochs	Hykist dev	Hykist test	Continue fine-tuning $\longrightarrow$	Epochs	Hykist dev	Hykist test
0	-	-		33	33.0	38.8
3	45.6	48.6		33	**32.5**	**38.4**
7	43.4	45.9		33	34.3	39.5
10	44.6	47.1		33	34.1	39.5

Table 5.17: *WERs* [%] comparison of On-off Regularization technique over epochs for raw waveform from scratch training. "Off Regularization" stage means training without regularization techniques like Dropout, SpecAugment and *IF Loss*. After training for some first epochs, the learning rate is reset and the model is preloaded in the "On Regularization" stage (all regularization techniques are turned on). All models are then continued being finetuned until full convergence on Vietnamese in-house data and recognized on HYKIST dataset. The 3rd row (0 epoch for "Off Regularization") is the baseline.

stage) for the first 3 epochs and then reset the learning rate and continue training with all regularizations turned on ("On Regularization" stage), we achieve the *WER*s reduction from 33.0% and 38.8% to 32.5% and 38.4% on dev and test set respectively compared to the baseline.

In the future work, we plan to apply the "On-off Regularization" technique to other pretraining schedules.

5.5.3 Combination of L2 regularization and Intermediate Focal Loss

Arch.	Init.	Pre-training data	Reg.	WER [%]	
				Hykist dev	Hykist test
Large$_{1-8}$	None	Viet. in-house (219h)	With IF	28.6	33.0
			With IF + L2	28.6	32.9
		None	With IF	33.0	38.8
			With IF + L2	31.4	36.4
Base		Viet. in-house (219h)	With IF	29.0	33.0
			With IF + L2	28.8	32.3
Large$_{1-8}$	*XLSR-53*		With IF	24.7	29.1
			With IF + L2	24.3	29.2

Table 5.18: *WERs* [%] comparison of the L2 Regularization combination with *IF Loss*. All models are finetuned until full convergence on Vietnamese in-house data and recognized on HYKIST dataset.

Due to time constraint and project requirement, we only tune L2 regularization values in favor of HYKIST data performance. By using grid-search technique for right L2 value selection, we are able to reduce *WERs* of multiple pretraining schedules as shown in Table 5.18. Notable results are seen on raw waveform from scratch training, where *WERs* reduce from 33.0% and 38.8% (only *IF Loss*) to 31.4% and 36.4% (combination with L2 regularization) on dev and test set respectively, that makes *WERR* 5.5% in average.

We stick with default parameters for *ICE Loss* and *IF Loss* because *WER*s do not fluctuate significantly when choosing other parameters. However, the right parameters for L2 regularization are chosen based on grid-search strategy and different parameters make the *WER*s vary greatly. Therefore, we recommend the use of L2 regularization should be the last regularization effort in the entire regularization pipeline due to its higher sentitivity to *WER*s compared to *ICE Loss* and *IF Loss*.

6 Conclusion

6.1 Overall results

In this thesis, we describe our efforts to develop HYKIST-related *ASR* systems for conversational telephone speech in the medical field for Vietnamese language.

Firstly, we use various acoustic encoder topologies to present supervised-only baselines while deploying the hybrid *HMM* framework.

Secondly, we use unsupervised *wav2vec 2.0* pretraining to improve system performance and analyze the effects of pretraining data on performance. The experimental findings demonstrate that this is especially effective when diverse pretraining data is used, e.g. data on multiple domains, multi-speaker data, augmented data... Also, multilingual pretraining does not always outperform monolingual pretraining. It is also shown that cost-effective model development is possible by utilizing the *XLSR-53* model, which is freely available. We then compare with the baselines and show that the *wav2vec 2.0* unsupervised pretraining does not always outperform the *Transformer* supervised-only approach, especially when the pretrained data is not diverse enough.

Thirdly, our best method to further improve the accuracy is using continued pretraining approach, where we pretrain multiple 8kHz datasets using parameters initialized by the 16kHz multilingual *XLSR-53* model. We show that continued pretraining is beneficial for both the monolingual and the multilingual scenario. However, the continued pretraining on less diverse data benefits more than the diverse data.

Fourthly, we compare the performance of *wav2vec 2.0* encoders and recommend the *Base* architecture instead of *Large*$_{1-8}$ for the sake of both accuracy and inference performance. We also recommend the use of Kaiming Initialization for better accuracy of *wav2vec 2.0* architecture, instead of Xavier Initialization.

Finally, we apply and analyze the use of intermediate loss - *Intermediate Cross-Entropy Loss (ICE Loss)* and *Intermediate Focal Loss (IF Loss)* - to make *wav2vec 2.0* more robust for all

recognition domains. We prove that *IF Loss* works better than *ICE Loss* in all data domains. In addition, for the small out-of-domain recognition *IF Loss* works well but for the large out-of-domain recognition it should only be applied on less diverse pretrained data. In order to further improvement of accuracy, we integrate *IF Loss* with On-off Regularization and L2 Regularization.

6.2 Future work

During the work of this thesis, we have discovered some promising directions which are planned for future work. First, section 5.2 shows that the system performance benefits from the unsupervised pretraining on diverse data but pretraining on the in-domain data, medical speech data in other words, is not compared yet. Second, we show that the data augmentation in pretraining stage is effective. However, such data augmentation for finetuning is not investigated yet. Third, due to time constraint, the effectiveness of On-off Regularization for different pretraining schedules is not studied. This leads to the question if sequence discriminative training [19], which also uses learning rate reset, works well with *wav2vec 2.0*. Finally, Wav2vec 2.0 - Conformer [50] has been popular lately. However, its effectiveness on Vietnamese has not been investigated yet.

A Bibliography

[1] A. M. H. J. Aertsen, J. H. J. Olders, and P. I. M. Johannesma. *Spectro-temporal receptive fields of auditory neurons in the grassfrog*. en. 1981. DOI: 10.1007/bf00342772. URL: http://dx.doi.org/10.1007/BF00342772.

[2] Afshine Amidi and Shervine Amidi. *Deep Learning Cheatsheet*. 2018.

[3] Rosana Ardila et al. "Common Voice: A Massively-Multilingual Speech Corpus". English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 4218–4222. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.520.

[4] Rosana Ardila et al. "Common Voice: A Massively-Multilingual Speech Corpus". In: *Proceedings of the 12th Language Resources and Evaluation Conference*. 2020, pp. 4218–4222.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. DOI: 10.48550/ARXIV.1607.06450. URL: https://arxiv.org/abs/1607.06450.

[6] Alexei Baevski et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12449–12460.

[7] Junwen Bai et al. "Joint unsupervised and supervised training for multilingual ASR". In: IEEE. 2022, pp. 6402–6406.

[8] Leonard E. Baum and John A. Eagon. "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology". In: *Bulletin of the American Mathematical Society* 73 (1967), pp. 360–363.

[9] Eugen Beck et al. "Lstm language models for lvcsr in first-pass decoding and lattice-rescoring". In: *arXiv preprint arXiv:1907.01030* (2019).

[10] K. Beulen and H. Ney. "Automatic Question Generation For Decision Tree Based State Tying". In: *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*. 1998, pp. 805–809.

[11]   Maximilian Bisani and Hermann Ney. "Joint-sequence models for grapheme-to-phoneme conversion". In: *Speech Communication* 50.5 (2008), pp. 434–451. ISSN: 0167-6393. DOI: `https://doi.org/10.1016/j.specom.2008.01.002`. URL: `https://www.sciencedirect.com/science/article/pii/S0167639308000046`.

[12]   Guoguo Chen et al. "Data Augmentation For Children's Speech Recognition–The" Ethiopian" System For The SLT 2021 Children Speech Recognition Challenge". In: *arXiv preprint arXiv:2011.04547* (2020).

[13]   Chung-Cheng Chiu et al. "Speech Recognition for Medical Conversations". In: *Proc. Interspeech 2018*. 2018, pp. 2972–2976. DOI: `10.21437/Interspeech.2018-40`.

[14]   Alexis Conneau et al. "Unsupervised cross-lingual representation learning for speech recognition". In: *arXiv preprint arXiv:2006.13979* (2020).

[15]   Patrick Doetsch et al. "RETURNN: the RWTH extensible training framework for universal recurrent neural networks". In: New Orleans, LA, USA, 2017.

[16]   Timothy Dozat. "Incorporating nesterov momentum into adam". In: (2016).

[17]   Erik Edwards et al. "Medical Speech Recognition: Reaching Parity with Humans". In: *Speech and Computer*. Ed. by Alexey Karpov, Rodmonga Potapova, and Iosif Mporas. Cham: Springer International Publishing, 2017, pp. 512–524. ISBN: 978-3-319-66429-3.

[18]   Mark J. F. Gales et al. "Speech recognition and keyword spotting for low-resource languages: Babel project research at CUED." In: *SLTU*. ISCA, 2014, pp. 16–23. URL: `http://dblp.uni-trier.de/db/conf/sltu/sltu2014.html#GalesKRR14`.

[19]   Matthew Gibson and Thomas Hain. "Hypothesis spaces for minimum Bayes risk training in large vocabulary speech recognition." In: Citeseer.

[20]   Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[21]   Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. URL: `https://proceedings.mlr.press/v9/glorot10a.html`.

[22] Donald D. Greenwood. *A cochlear frequency-position function for several species—29 years later*. en. June 1990. DOI: 10.1121/1.399052. URL: http://dx.doi.org/10. 1121/1.399052.

[23] Anmol Gulati et al. "Conformer: Convolution-augmented transformer for speech recognition". In: *arXiv preprint arXiv:2005.08100* (2020).

[24] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[25] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[26] Dan Hendrycks and Kevin Gimpel. "Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units". In: *CoRR* abs/1606.08415 (2016). arXiv: 1606. 08415. URL: http://arxiv.org/abs/1606.08415.

[27] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco. 1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[28] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[29] Wei-Ning Hsu et al. "HuBERT: How much can a bad teacher benefit ASR pre-training?" In: IEEE. 2021, pp. 6533–6537.

[30] Wei-Ning Hsu et al. "Robust wav2vec 2.0: Analyzing Domain Shift in Self-Supervised Pre-Training". In: *Interspeech*. 2021.

[31] Kazuki Irie et al. "Language Modeling with Deep Transformers". In: *Proc. Interspeech 2019* (2019), pp. 3905–3909.

[32] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. "Product Quantization for Nearest Neighbor Search". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2011), pp. 117–128.

[33] Snigdhaswin Kar et al. "Systematic Evaluation and Enhancement of Speech Recognition in Operational Medical Environments". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021. 9553607.

[34] Kazuya Kawakami et al. "Learning robust and multilingual speech representations". In: *arXiv preprint arXiv:2001.11128* (2020).

[35] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of NAACL-HLT*. 2019, pp. 4171–4186.

[36] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR (Poster)*. 2015.

[37] R. Kneser and H. Ney. "Improved backing-off for M-gram language modeling". In: *1995 International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. 1995, 181–184 vol.1. DOI: 10.1109/ICASSP.1995.479394.

[38] Tom Ko et al. "Audio augmentation for speech recognition". In: *INTERSPEECH*. 2015.

[39] Anders Krogh and John Hertz. "A Simple Weight Decay Can Improve Generalization". In: *Advances in Neural Information Processing Systems*. Ed. by J. Moody, S. Hanson, and R.P. Lippmann. Vol. 4. Morgan-Kaufmann, 1991. URL: https://proceedings.neurips.cc/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf.

[40] Guillaume Lample and Alexis Conneau. "Cross-lingual language model pretraining". In: *arXiv preprint arXiv:1901.07291* (2019).

[41] Jaesong Lee and Shinji Watanabe. "Intermediate loss regularization for ctc-based speech recognition". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6224–6228.

[42] M. Paul Lewis, ed. *Ethnologue: Languages of the World*. Sixteenth. Dallas, TX, USA: SIL International, 2009.

[43] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[44] Hieu-Thi Luong and Hai-Quan Vu. "A non-expert Kaldi recipe for Vietnamese speech recognition system". In: *Proceedings of the Third International Workshop on World-wide Language Service Infrastructure and Second Workshop on Open Infrastructures and Analysis Frameworks for Human Language Technologies (WLSI/OIAF4HLT2016)*. 2016, pp. 51–55.

[45] Christoph Lüscher et al. "Development of Hybrid ASR Systems for Low Resource Medical Domain Conversational Telephone Speech". Baseline systems for Arabic, German, and Vietnamese for HYKIST project. Submitted to ICASSP 2023. Oct. 2022. URL: http://https://arxiv.org/abs/2210.13397.

[46] Christoph Lüscher et al. "RWTH ASR Systems for LibriSpeech: Hybrid vs Attention". In: Graz, Austria, Sept. 2019.

[47] Anirudh Mani, Shruti Palaskar, and Sandeep Konam. "Towards Understanding ASR Error Correction for Medical Conversations". In: *NLPMC*. 2020.

[48] Abdelrahman Mohamed et al. "Self-Supervised Speech Representation Learning: A Review". In: *arXiv preprint arXiv:2205.10643* (2022).

[49] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[50] Edwin G. Ng et al. "Pushing the Limits of Non-Autoregressive Speech Recognition". In: *Proc. Interspeech 2021*. 2021, pp. 3725–3729. DOI: `10.21437/Interspeech.2021-337`.

[51] Stefan Ortmanns, Hermann Ney, and Xavier Aubert. "A word graph algorithm for large vocabulary continuous speech recognition". In: *Computer Speech & Language* 11.1 (1997), pp. 43–72.

[52] Myle Ott et al. "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 2019, pp. 48–53.

[53] Vassil Panayotov et al. "Librispeech: an asr corpus based on public domain audio books". In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 5206–5210.

[54] Daniel S Park et al. "Specaugment: A simple data augmentation method for automatic speech recognition". In: *arXiv preprint arXiv:1904.08779* (2019).

[55] Ankita Pasad, Ju-Chieh Chou, and Karen Livescu. "Layer-wise analysis of a self-supervised speech representation model". In: IEEE. 2021, pp. 914–921.

[56] Vijayaditya Peddinti et al. "JHU ASpIRE system: Robust LVCSR with TDNNS, iVector adaptation and RNN-LMS". In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (2015), pp. 539–546.

[57] Vineel Pratap et al. "MLS: A Large-Scale Multilingual Dataset for Speech Research". In: *INTERSPEECH*. 2020.

[58] K. Sreenivasa Rao and Manjunath K E. *Speech Recognition Using Articulatory and Excitation Source Features*. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-49220-9. URL: http://dx.doi.org/10.1007/978-3-319-49220-9.

[59] David Rybach et al. "RASR - The RWTH Aachen University Open Source Speech Recognition Toolkit". In: Waikoloa, HI, USA, Dec. 2011.

[60] Sakriani Sakti et al. "Towards Multilingual Conversations in the Medical Domain: Development of Multilingual Medical Data and A Network-based ASR System". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 2639–2643. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/709_Paper.pdf.

[61] Ralf Schlüter et al. "Gammatone Features and Feature Combination for Large Vocabulary Speech Recognition". In: Honolulu, HI, USA, Apr. 2007, pp. 649–652.

[62] Steffen Schneider et al. "wav2vec: Unsupervised Pre-Training for Speech Recognition". In: Graz, Austria, Sept. 2019, pp. 3465–3469.

[63] M. Schuster and K.K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. DOI: 10.1109/78.650093.

[64] Anuroop Sriram, Michael Auli, and Alexei Baevski. "Wav2Vec-Aug: Improved self-supervised training with limited data". In: *arXiv preprint arXiv:2206.13654* (2022).

[65] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[66] Andreas Stolcke. "SRILM – An extensible language modeling toolkit". In: *IN PROCEEDINGS OF THE 7TH INTERNATIONAL CONFERENCE ON SPOKEN LANGUAGE PROCESSING (ICSLP 2002)*. 2002, pp. 901–904.

[67] Łukasz Sus. *Wav2Vec 2.0: A Framework for Self-Supervised Learning of Speech Representations - Model for speech recognition explained*. Ed. by towardsdatascience.com. June 2021. URL: https://towardsdatascience.com/wav2vec-2-0-a-framework-for-self-supervised-learning-of-speech-representations-7d3728688cae.

[68] Andros Tjandra et al. "Deja-vu: Double feature presentation and iterated loss in deep transformer networks". In: IEEE. 2020, pp. 6899–6903.

[69] Zoltán Tüske et al. "Data Augmentation, Feature Combination, and Multilingual Neural Networks to Improve ASR and KWS Performance for Low-resource Languages". In: Singapore, Sept. 2014, pp. 1420–1424.

[70] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[71] Peter Vieting et al. "On Architectures and Training for Raw Waveform Feature Extraction in ASR". In: IEEE. 2021, pp. 267–274.

[72] Chengyi Wang et al. "Unispeech: Unified speech representation learning with labeled and unlabeled data". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10937–10947.

[73] Mohammad Zeineldeen et al. "Conformer-based hybrid ASR system for Switchboard dataset". In: IEEE. 2022, pp. 7437–7441.

[74] Mohammad Zeineldeen et al. *Improving the Training Recipe for a Robust Conformer-based Hybrid Model*. 2022. DOI: 10.48550/ARXIV.2206.12955. URL: https://arxiv.org/abs/2206.12955.

[75] Aston Zhang et al. "Dive into deep learning". In: *arXiv preprint arXiv:2106.11342* (2021).

[76] Frank Zhang et al. "Fast, Simpler and More Accurate Hybrid ASR Systems Using Wordpieces". In: *arXiv preprint arXiv:2005.09150* (2020).

[77] Zi-Qiang Zhang et al. "XLST: Cross-lingual self-training to learn multilingual representation for low resource speech recognition". In: *arXiv preprint arXiv:2103.08207* (2021).

B List of Abbreviations and Glossaries

AM Acoustic model 15, 33

ASR Automatic Speech Recognition: an interdisciplinary subfield of computer science and computational linguistics that develops methodologies and technologies that enable the recognition and translation of spoken language into text by computers 5–11, 15, 25, 38, 44, 50, 55

BLSTM Bidirectional Long-Short Term Memory 33, 38

CART Classification and Regression Tree 13, 33, 35, 38, 68

CE Cross-entropy 36

CNN Convolutional Neural Network 12, 20, 34, 35

DCT Discrete Cosine Transform 11

DFT Discrete Fourier Transform 11

DNN Deep Neural Network 14, 35

E2E End-to-End 9

EM Expectation–maximization 13

fCE Frame-wise Cross-entropy 33, 35

GELU Gaussian Error Linear Units: an activation function 12, 25, 26

GMM Gaussian Mixture Model 13, 14, 33, 35, 38

HMM Hidden Markov Model 12–14, 33, 35, 38, 55

ICE Loss Intermediate Cross-Entropy Loss 36, 46–48, 50, 54–56, 69

IF Loss Intermediate Focal Loss 36, 37, 48–56, 69, 70

Latent Speech Representations Latent (or hidden) variables from empirical measurements, e.g. speech input 8, 25, 26

LM Language model 14, 15, 32

LSTM Long-Short Term Memory 21, 22

MFCC Mel Frequency Cepstral Coefficients 11

MLP Multilayer Perceptron 16, 17

MT Machine Translation: a sub-field of computational linguistics that investigates the use of software to translate text or speech from one language to another 5–7

NN Neural Network 35

OOV Out-of-vocabulary 32

PPL Perplexity 32

ReLU Rectified Linear Units: an activation function 34

RNN Recurrent Neural Network 21, 22, 38

SOTA State-of-the-art 25

Transformer A state-of-the-art architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies efficiently 25–27, 33–35, 38, 42, 46–51, 55, 69, 70

wav2vec 2.0 A framework for self-supervised learning of speech representations 5, 8, 25–27, 33–36, 39, 42, 44, 55, 56, 68

WER Word-error-rate 15, 37–54, 68–70

WERR Word-error-rate reduction 49, 50, 54

XLSR-53 A large cross-lingual speech representation model pretrained in 53 languages 4, 5, 8, 27, 34, 35, 41, 46, 49, 55

YT YouTube 30, 39–42, 46–50

List of Figures

List of Tables