Automatic Speech Recognition for Small Data and Its Application on Cognitive Assessment

Liu Chen
M. S., Oregon Health & Science University, 2016

Presented to the
within the Oregon Health & Science University
School of Medicine
in partial fulfillment of
the requirements for the degree
Doctor of Philosophy
in
Computer Science & Engineering

November 2023

School of Medicine
Oregon Health & Science University

CERTIFICATE OF APPROVAL

This is to certify that the Ph. D. dissertation of

Liu Chen

has been approved.

Meysam Asgari, Thesis Advisor
Associate Professor

Hiroko Dodge
Professor

Xubo Song
Professor

Steven Bedrick
Associate Professor

Peter Heeman
Associate Professor

Acknowledgements

I would like to express my deepest appreciation to my advisor, Meysam Asgari, for his guidance during my Ph.D. degree. His support and encouragement have been invaluable to me.

I would like to thank my dissertation advisor committee — Hiroko Dodge, Xubo Song, Peter Heeman, and Steven Bedrick — for their invaluable patience and insightful feedback. I would also like to thank Peter Heeman for countless hours of helping me revise my dissertation. I would like to thank current and former faculties of Center for Spoken Language Understanding for their important input and motivation. Many thanks to Patricia Dickerson for her great administrative support.

I would like to thank Tuan Dinh and Zicheng Ren for all their support and help. I would like to thank Jessie Walder-Biesanz, Hellen Rogway, and Jodi Walder for their kindness and support since I came to America. I would like to thank my parents and my cousin for their unconditional support. I would like to thank all people who helped me see the world from a different perspective.

Contents

List of Tables

List of Figures

Abstract

Automatic Speech Recognition for Small Data and Its Application on Cognitive Assessment

Liu Chen

Doctor of Philosophy
within the Oregon Health & Science University
School of Medicine

November 2023
Thesis Advisor: Meysam Asgari

Automatic speech recognition (ASR) is an essential component for building automatic cognitive assessment systems designed to monitor older adults' cognitive status. While, in the ASR field, remarkable achievements have been reported on publicly available academic datasets, there are two under-explored problems that are important to building automatic cognitive assessment systems: ASRs' performance on aging voice and accuracy in transcribing keywords. Both problems are important to deliver high-quality transcriptions for assessment purposes.

In this dissertation, we focus on developing transfer learning techniques/methods to build ASRs that perform well on older adults with possible cognitive impairment. Firstly, we present a transfer learning technique to improve an open-source ASR's performance on older adults (80+ years old) with limited data (i.e., about 10 hours of audio recordings). We demonstrate the aging voice dramatically impacts an ASR's performance and adapting the ASR with older adults' recording data through fine-tuning can improve the performance. We propose a transfer learning technique that utilizes intermediate outputs

to increase the fine-tuning efficiency with limited training data. This technique achieves better performance than the standard fine-tuning.

Secondly, we refine the DNN architecture of Wav2Vec 2.0 in order to improve the training efficiency on a small training dataset (i.e., about 100 hours of recorded audio), and a small DNN architecture (i.e., having a small number of parameters). We refine the DNN architecture with the local attention mechanism and parameter sharing. As a preliminary study, we compare the training efficiency difference between our refined DNN architecture and the original architecture through comparing their models that are trained on a small academic dataset. Our model performs 16% better than the other.

Thirdly, we propose a transfer learning strategy that can effectively resolve the not-transcribe-filler problem while causing minor negative side effects to Whipser ASRs which are the state-of-the-art DNN-based ASRs. Evaluating models on English and Chinese testing datasets, we show that both large training datasets and scaling the model size of ASRs do not guarantee these models transcribe the keyword accurately. Analyzing the Whisper-large-v1, we show that the acoustic encoder cannot generate general hidden representations for fillers. Moreover, we show that tuning the encoder not only increases the transcription accuracy of fillers on in-domain and out-of-domain testing sets but also moderately impacts the model's native ability of multilingual transcription.

In addition to improving ASRs when there is training data available, we also encounter a project that does not have any training data available. We utilize ASR as a tool to extract handcrafted time-based features from the animal fluency test to improve the accuracy of cognitive assessment. Combining count-based features and our proposed time-based features achieves the best performance in distinguishing older people with mild cognitive impairment from those with normal cognition.

Chapter 1

Introduction

1.1 Motivation

Before the 2010s, the dominant ASR systems were based on representing speech signals using Gaussian Mixture Models (GMMs) that are based on Hidden Markov Models (HMMs) [112]. Such systems, which are called the GMM-HMM ASRs, are based on the fact that a speech signal can be considered as a piecewise stationary signal which can be modeled by a GMM [112] and HMM is used to model the temporal dependencies. In the early 2010s, research demonstrated that the benefit of replacing GMMs with DNNs in HMM-based ASR systems increased with the size of the training dataset [112]. The DNN-HMM ASR achieved a one-third error rate reduction on the Switchboard conversational transcription task over the state-of-the-art GMM-HMM ASRs [134].

In the middle 2010s, Hannun et al. [57] developed a carefully engineered DNN-based ASR, called DeepSpeech, that utilized DNNs to also model the temporal dependencies and this ASR outperformed various HMM-based ASRs including both GMM-HMM and DNN-HMM ASRs [57]. Amodei et al. [7] further improved this DNN-based ASR by focusing on three components: the DNN architecture, size of training data, and computation power, and named the improved DNN DeepSpeech2. Through applying distributed training across 16 Nvidia GPUs and increasing the training data to 12,000 hours, Amodei et al. demonstrated continuing improvements by increasing the training data and parameter size of the DNN.

Since then, in the ASR field, the benefits of increasing computational power, the size of the DNN, and training data have been constantly reported. In the 2020s, Baevski et

al. [17] proposed a training method that used unlabeled datasets to train a DNN and chose a DNN architecture that was designed to deploy distributed training across multiple GPUs. Through applying distributed training across 128 GPUs and scaling to 53,000 hours of unlabeled training data, Baevski's best ASR, which is named Wav2Vec 2.0 Large, achieved state-of-the-art performance on testing sets of Librispeech which is a widely used academic dataset in the ASR field. Various methods based on Wav2vec 2.0 were proposed and further improved the performance on the same testing sets. On the other hand, Radford et al. [122] put their focus on improving the robustness of DNN-based ASRs (i.e., reasonable performance across multiple testing sets). Radford scaled the labeled training data to 680,000 hours through combining multiple types of data including transcription datasets from 97 languages and translation datasets. Compared with the best Wav2vec 2.0 ASR, the best Whisper ASR achieved an average relative error reduction of 55.2% when evaluated on 12 academic speech recognition datasets.

Even with such remarkable achievements, there are a number of deficits of current systems. Two problems explored in this dissertation are: the ASR's performance on older adults with possible cognitive impairment and the performance on certain types of words (e.g., fillers, repeated words). Nowadays, researchers report their ASR's performance on academic testing sets whose audio recordings mostly belong to healthy working adults. Older adults' vocal characteristics are different from those adults. Age-related speech deterioration begins around 60 years old [104], resulting in significantly different voice characteristics in comparison to the younger generation [88]. Moreover, the plausible influence of impaired cognitive functioning on acoustic features of mild cognitive impairment (MCI) subjects [126] may serve as an additional source of acoustical mismatch. Feng et al. [40] showed that an ASR performance varies among different age groups: teenagers' speech is the best recognized and the performance on older adults (over 65 years old) is worse than the former. Moreover, they demonstrated that the articulation changes among older adults, especially those over 75 years old, negatively impact their ASR's performance [40]. Wang et al. [159] also reported a similar observation. In the same recording environment, an ASR performs noticeably better on recognizing adults' recordings than recordings from older adults with possible cognitive impairment.

A second under-explored problem is that state-of-the-art ASRs perform badly at transcribing disfluencies, especially filler, and the word error rate cannot correctly reveal an ASR's performance on transcribing those words. For example, let's assume that we have a testing set that has 1000 recordings/utterances where each utterance has 10 words and a 50% chance it has a filler. If an ASR does not transcribe any filler and makes no other errors, the error rate of this ASR on this testing set is 5%. We can consider the ASR performs well on this testing set because its error rate is low and it makes no errors that change any utterance's semantic meaning. But, since no fillers are transcribed, researchers cannot use the ASR's transcriptions to study anything related to fillers.

ASRs in automatic cognitive assessment systems: An ASR is essential to an automatic cognitive assessment system that monitors older adults' cognitive progress and can also contribute to improving assessment accuracy. While cognitive assessment is important to help older adults prepare themself for potential cognitive impairment (e.g., mild cognitive impairment, Alzheimer's disease) that may be treatable for a period of time, the manual assessment only covers parts of the older population. Lang et al. [84] showed that the undetected dementia population rate in the U.S. is 61%. Automatic cognitive assessment systems provide a convenient as well as less labor-intensive way for monitoring older adults' cognitive progress and can potentially cover more older adults.

The general pipeline of an automatic cognitive assessment system, which is presented in Figure 1.1, contains three major components. The first component is an ASR that transcribes an older adult's audio. The second component is a group of feature extractors, which take text/audio as input, for various cognitive tests (e.g., describe a given picture, recall animal names, recall a told story, etc). The third component is a classification module that takes the extracted features as input and makes the assessment.

1.2 Dissertation Problem and Statement

Both under-explored problems are essential to automatic cognitive assessment systems. First, if an ASR does not transcribe recordings accurately, feature extractors that were

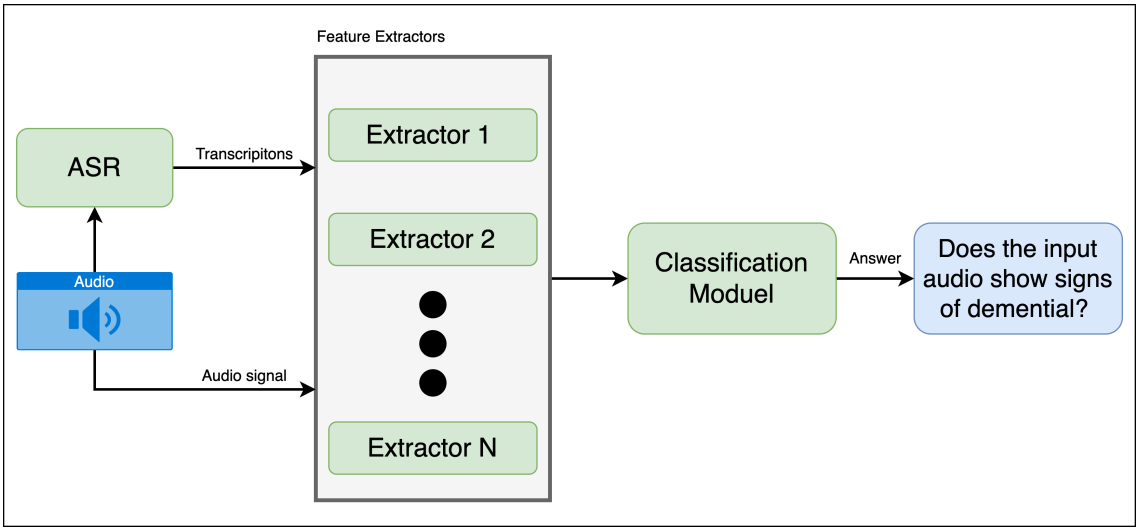

Figure 1.1: The general pipeline of an automatic cognitive assessment system. The audio recordings from various cognitive tests are first transcribed by an ASR. Then, the audio and transcribed text are passed to multiple feature extractors and each extractor extracts features for a specific cognitive test. A classification module that takes the extracted features as input and makes the assessment.

evaluated on manual transcriptions cannot extract meaningful information from an answer, resulting in reduced assessment accuracy. Second, various cognitive research has demonstrated that several types of keywords (e.g., fillers, repeated words) have achieved promising results in cognitive assessment. If an ASR mistranscribes these keywords, those well-studied features cannot contribute to improving the performance of assessment systems.

Data scarcity is the major problem that we have to face when building an ASR that performs well on older adults with possible cognitive impairment. While the mainstream idea of building DNN-based ASR is training a large DNN model with big data (i.e., 1000+ hours of transcribed recordings from thousands of speakers), the available training data from older adults is much smaller than the size of big data. Empirically, we can hardly gather 100 hours of recordings from older adults. Thus, it is essential to develop techniques for improving DNN-based ASR in order to fully utilize the available small data.

In this dissertation, we utilize transfer learning to improve ASRs for older adults and make them suitable for the automatic cognitive assessment system. Transfer learning has

been well-studied in resolving the data scarcity problem, thus we develop transfer learning techniques/methods to resolve both under-explored problems. For the first problem, we propose a transfer learning method that can efficiently adapt a pre-trained DNN model to the target domain (Chapter 4) and we also propose a DNN architecture to fully utilize small training data (Chapter 5). For the second problem, we propose a transfer learning strategy that causes minor negative side effects to a large pre-trained DNN model (Chapter 6).

1.3 Contribution Overview

In this dissertation, we make the following contributions.

We present an efficient transfer learning technique and develop DNN architecture for training the base model (i.e., the pre-trained model). In Chapter 4, based on the hypothesis that a pre-trained DNN model's intermediate outputs contain useful information related to a target domain, we present a technique to improve a DNN-based ASR's performance on older adults (75+ years old) with limited data (i.e., 10 hours of audio recordings). We first demonstrate the aging voice dramatically impacts an ASR performance. This performance difference supports the domain difference between the pre-trained DNN-based ASR model and the aging voice. Then, we show that the pre-trained DNN-based ASR's performance on aging voices can be improved through adapting this ASR with older adults' recording data through transfer learning. Third, we propose a transfer learning technique that improves the adapting efficiency through leveraging the intermediate outputs from the pre-trained model. Our technique performs better than the standard fine-tuning technique when there are only 10 hours of training data.

In Chapter 5, our goal is to build pre-trained/base ASR models for transfer learning purposes. We hypothesize that the domain difference between older adults with normal cognition and those with cognitive impairment is small. Collecting data from the former population is easier than gathering data from the latter. If we train a DNN-based ASR for the former population, we can easily adapt this model to the latter population due to the small domain difference. Empirically, we think that collecting 100 hours of transcribed recordings from older adults with normal cognition for training is achievable

and fully utilizing this training dataset is essential to build the pre-trained model for the transfer learning purpose. As a preliminary study, we refine Transformer [155], which is a well-studied DNN architecture, in order to improve the training efficiency on a small training dataset (i.e., 100 hours of recorded audio) and evaluate it on a small academic dataset. Though analyzing a publicly available Transformer-based ASR, we propose two modifications: applying local attention mechanism and parameter sharing across attention blocks. We evaluate each modification separately and show the improvement over the original DNN architecture through comparing the performances of trained DNNs. Then, we show that applying both modifications results in the most efficient DNN architecture in this chapter.

Ensuring transcription accuracy of keywords is essential but gains much less attention than it should be. Because various research has shown scaling training data and DNN model size increase the performance and robustness when evaluating on word error rate (WER), which is the mainstream evaluation matrix. But, low WER does not guarantee high transcription accuracy on keywords. In Chapter 6, we propose a transfer learning strategy that can efficiently improve the transcription accuracy of fillers while causing minor negative side effects to Whisper ASRs, which are autoregressive ASR models. We analyze multiple Whisper ASR models, in which their parameter sizes are different, and each model consists of an acoustic encoder and a linguistic decoder. We first demonstrate the importance of this problem through comparing these ASR models' performance on transcribing fillers, which are well-studied keywords in cognitive research, on both English and Chinese testing sets. We show that both large training datasets and enlarging model size do not guarantee good accuracy of transcribing fillers on either language and demonstrate that WER cannot reveal the problem accurately. Then, considering the largest ASR model as the research target, we analyze the causes of this transcribing problem. We show that the ASR model's acoustic encoder does not generate general representations for fillers. Based on the analyzing result, we propose an effective transfer learning strategy (i.e., only fine-tune the encoder). Training on an English dataset, the tuned model not only performs well on various in-domain and out-of-domain Engish testing sets but also preserves the base/pre-trained model's abilities of multilingual transcription (i.e., no

performance drop on a Chinese testing set).

In addition to improving ASRs when there is training data available, we also encounter a project that does not have any training data available. Thus, we utilized an HMM-based ASR as an aligner to extract timestamps of manual transcriptions, which is widely used in phonetic studies [19, 81, 142] and focus on improving the cognitive assessment. In Chapter 3, we utilize HMM-based ASR as a tool to extract handcrafted time-base features from the animal fluency (AF) test to improve the accuracy of cognitive assessment. We propose time-based features based on early cognitive studies on the AF test and utilize these features to distinguish older people with mild cognitive impairment (MCI) from those with normal cognition (NC).

Over the last 5 years, the remarkable progress of ASR has been achieved through improving training approaches in order to scale DNN, training data, and computational power. We conducted our research with the state-of-the-art approaches at that time. In Chapter 4, we use a pre-trained DeepSpeech2 [7] as the base model and develop our transfer learning technique that utilizes intermediate outputs during fine-tuning the model for the target domain. In Chapter 5, we conduct our research on training base models with a small dataset based on Wav2Vec 2.0 [17]. In Chapter 6, we analyze Whisper models [122] and propose a tuning strategy that makes minor side effects on the models' original capability. Since each chapter focuses on different aspects of transfer learning, we can apply our works in Chapter 4 and 5 to Whisper.

Chapter 2

Background

In this chapter, we present background related to our research. The background contains three major parts: general knowledge about DNN, DNN-based ASRs, and a brief overview of mild cognitive impairment.

First, we present general knowledge of DNN, which is slanted towards what is needed for DNN-based ASRs. In Section 2.1, we present the general pipeline of DNN, which serves as the overview for coming sections. In Section 2.2, we present seq-to-seq DNNs and two subcategories: autoregressive and non-autoregressive DNNs. In Section 2.3, we present DNN layers used in seq-to-seq ASR systems. Moreover, we present the attention mechanism in detail since it is one of the core components used in our research. In Section 2.5, we describe the training approaches for seq-to-seq DNNs, including supervised learning, contrastive learning and weakly-supervised learning. In Section 2.6, we present loss functions related to our research. Last, we present transfer learning in Section 2.7.

Second, we present approaches about automatic speech recognition (ASR), which is the task where machines (i.e., digital computers) accurately convert a speech signal into a text transcription [121]. More specifically, we present DNN-based seq-to-seq ASR systems. In Section 2.8, we present commonly used data representation methods in seq-to-seq ASR systems. In Section 2.9, we present three architectures that are used in our research in detail: DeepSpeech2 [7], Wav2Vec 2.0 [17] and Whisper [122].

Last, we present the background about mild cognitive impairment in Section 2.10.

2.1 Deep Neural Network

A Deep Neural Network (DNN) is a machine learning technique inspired by the human brain structure that provides computational systems with artificial intelligence. A DNN aims to approximate a groundtruth function f^* that people are interested in. For example, let $y = f(X)$, where $f()$ represents a complicated function that maps an input X to an output y. Both input and output can either be vectors or matrices. The task itself decides the shape of y. If it is a classification task, y is presented as a one-hot vector where only the element representing the corresponding category is one. ASR is a classification task where y is a stack of one-hot vectors. If it is a representation task that DNN learns to represent an X with a meaningful vector/matrix, then a y is a vector/matrix. The DNN defines a mapping $y^* = f^*(X;\theta)$ and the weights, θ, are used to learn to approximate f as good as possible from a lot of sample pairs (i.e., (X,y)) provided by a training dataset.

The DNN $f^*(\theta)$ can be multiple functions chained together. For example, let's assume there are four functions: $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ and $f^{(4)}$, and $f^*(X) = f^{(4)}(f^{(3)}(f^{(2)}(f^{(1)}(X))))$. In this example, $f^{(1)}$ is called the first layer of the network, $f^{(2)}$ is called the second layer, $f^{(3)}$ is called the third layer and $f^{(4)}$ is called the fourth layer. Each function has its weights $\theta^{(l)}$ where $l = 1, 2, 3, 4$. For this example, we use the linear layer (also known as the fully-connected layer), $f^{(l)}(input) = input\, W_l + b_l$, to linearly transforms an input matrix *input* to a new representation matrix through matrix multiplication. Both W_l and b_l are this layer's weights. In addition to linear layers, a DNN also incorporates non-linear functions. This is because a composition of linear functions is still a linear function and its expressive power is limited. Adding non-linear functions (commonly known as activation layers) to a DNN (i.e., $f^*()$ in this example) increases the expressive power [105].

In our example, let $f^{(3)}$ be an activation layer. The input X is first transformed by $f^{(1)}$ and $f^{(2)}$, and then, processed by the activation layer $f^{(3)}$. The output is transformed by the fourth linear layer, $f^{(4)}$, which is also known as the output layer, and this layer yields the final output y^*. It is worth noting that, while there is no constraint on adding activation layers anywhere in a DNN, empirically, researchers add an activation layer after every linear function. Moreover, nowadays, a DNN can have more than a hundred layers.

A four-layer DNN is presented in Figure 2.1 and the training and inference process. In the training process, many (X, y) pairs are used to train the DNN. A learning method guides the DNN to learn an optimal approximation through having a loss function to measure the difference between the ys and y^*s from Xs. The difference is called error in Figure 2.1 and is then used to update the $f^*()$ through a technique called backpropagation which is also known as the backward process [128]. The backward process first calculates gradients of all weights and then updates these weights with scaled gradients. The scaler is called the learning rate and the way of scaling gradients is called the optimizer. The training process is repeated multiple times until the updated DNN cannot reduce the error anymore or researchers decide to stop the process. The trained DNN is called a model and this term will be used throughout the dissertation. This model is an approximation of f among various possible approximations and is used to yield ys for unseen Xs. To increase the training stableness, multiple pairs are sampled in each training process and the average of errors over these pairs are used to update the DNN. The sampled pairs in a process are commonly known as batch or minibatch. Moreover, a DNN can be trained with multiple datasets of different tasks (i.e., various types of (X, y) pairs). This type of DNN usually has multiple output layers one for each task and this training style is called multi-task training. Neither types of datasets nor tasks impact the general training process. The inference process is similar to the training process, except there is no loss function and backward process.

2.2 Seq-to-seq DNN

In the previous section, we present the general idea of DNNs. In this section, we present the general concept of seq-to-seq DNNs, which are specific types of DNNs. The seq-to-seq DNN is a task-oriented concept. It refers to DNNs that map input sequences (e.g., a raw speech waveform, acoustic speech representation or words) into the corresponding output sequences (e.g., characters, words, or sub-word sequences). Based on the way of making predictions, seq-to-seq DNNs can be categorized into two categories: autoregressive (Section 2.2.1) and non-autoregressive DNN (Section 2.2.2). The major challenge of seq-to-seq

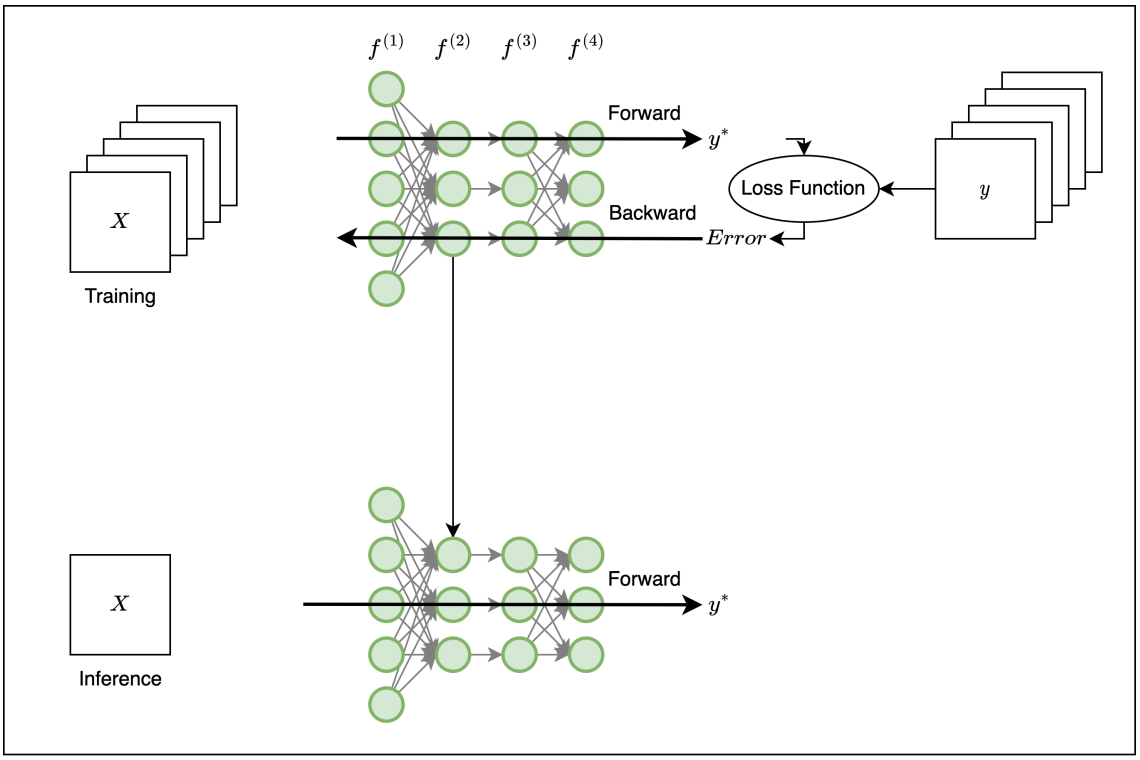

Figure 2.1: An example of DNN

DNN is that the length of the sequence pairs (i.e., the input and corresponding output sequences) is different. Both categories solve this challenge with different solutions based on their innate characteristics.

2.2.1 Autoregressive DNN

An autoregressive DNN makes next-output predictions based on the input sequence and predicted history. The DNN contains two components: an encoder and a decoder. The left graph in Figure 2.2 shows the general architecture. The encoder encodes the input sequence, (x_1, x_2, x_3, x_4), into hidden representations that contain essential information for the decoder and passes the representations to the decoder. Intuitively, the encoder can be considered as an independent DNN that transforms an input sequence into hidden representations. There is no constraint on the encoder's DNN architecture. Researchers can use a simple DNN (e.g., a DNN that only has one linear layer) as an encoder, while they can also use more complicated DNNs. Moreover, there is no constraint on the form of

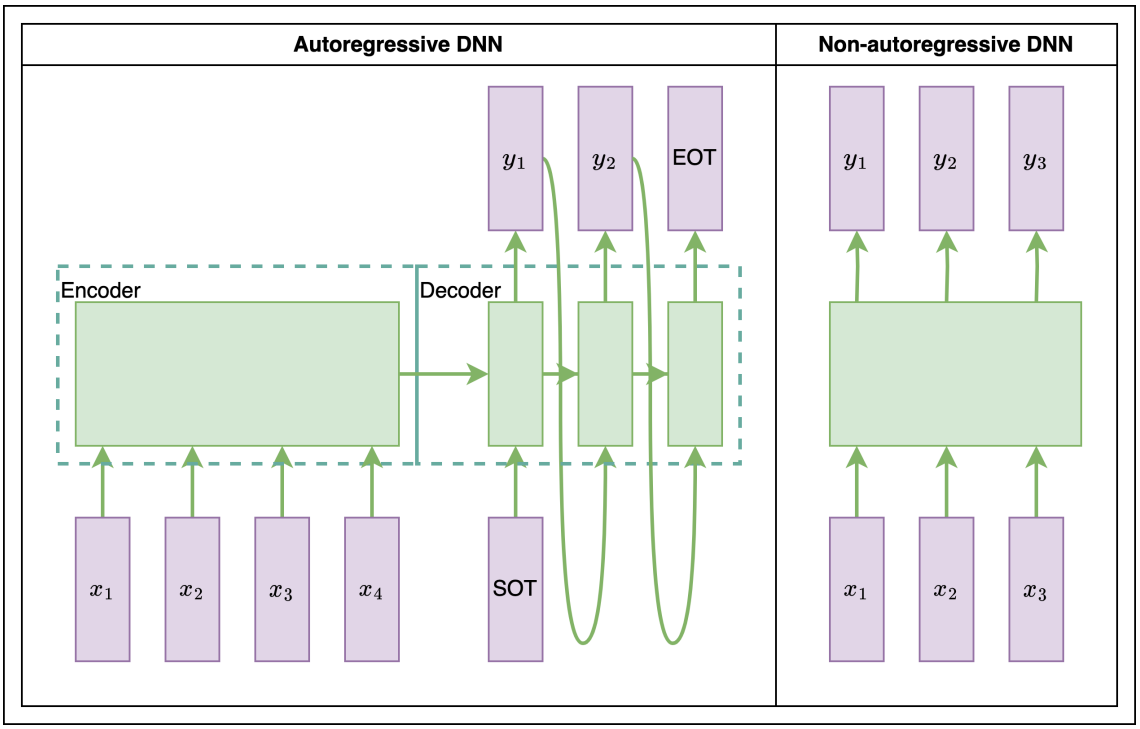

Figure 2.2: The general structure of autoregressive and non-autoregressive DNN. "SOT" and "EOT" refer to the dummy start and end symbols.

hidden representations. The decoder takes the representations and a dummy start symbol as inputs and predicts the next label. Each prediction depends on previous predictions and inputs. The prediction/inference process continues until a dummy end symbol is predicted. This means there is no requirement on the length of the output sequence. In other words, autoregressive DNN does not have any restriction on the input/output lengths. Similar to the encoder, there is no constraint on the decoder's DNN architecture.

2.2.2 Non-autoregressive DNN

A non-autoregressive DNN makes a new prediction only based on the input sequence. Unlike autoregressive DNN, the non-autoregressive DNN unifies the encoder and decoder into one component. The right graph in Figure 2.2 shows the general architecture. After DNN processes the input sequence, it directly outputs predictions. Moreover, no dummy symbol is needed. In other words, the length of input and output sequence must be the same and the sequence mismatch is a problem for non-autoregressive DNN. To resolve

the problem, researchers develop sequence expansion processes along with paired loss functions. These solutions rely on the characteristics of the task. Popular solutions for ASR will be presented in Section 2.6.

Both autoregressive and non-autoregressive DNNs have irreplaceable advantages. On the one hand, the inference speed of non-autoregressive DNNs is faster than autoregressive DNNs. Because, while autoregressive DNNs generate outputs iteratively, non-autoregressive DNNs generate all outputs simultaneously. On the other hand, autoregressive DNNs can resolve the length mismatch problem, while researchers have to design non-autoregressive DNNs for different situations (i.e., the input sequence is always shorter/longer than the output sequence). To our knowledge, no non-autoregressive DNN can handle unclear length differences between input and output sequences, and using autoregressive DNNs is the only option. In the ASR field, researchers generally agree that the input sequence is always longer than the output sequence. Thus, autoregressive and non-autoregressive DNNs can be used to build ASR systems.

2.3 Neural Net Layers

In this section, we present DNN layers used in the DNN architectures that will be presented in Section 2.9. We have introduced the linear layer in Section 2.1.

2.3.1 Convolution Layers

Convolution layers were proposed 30 years ago. It was initially proposed for image processing tasks to extract general characteristics from training images, such as detecting edges and shapes. Then, its use case has been expanded into ASR for extracting acoustic characteristics. In the rest of the section, we present both 2D convolution (Conv2d) and 1D convolution (Conv1d) layers, as they are used in the ASR field.

Conv2d layer was proposed by Lecun et al. [85] for handwriting recognition and nowadays is commonly used in image recognition and ASR. The layer applies filters, also known as kernels, to each small region of an input and produces an output value representing a feature in that region. Figure 2.3 shows how a kernel produces output from an input. In

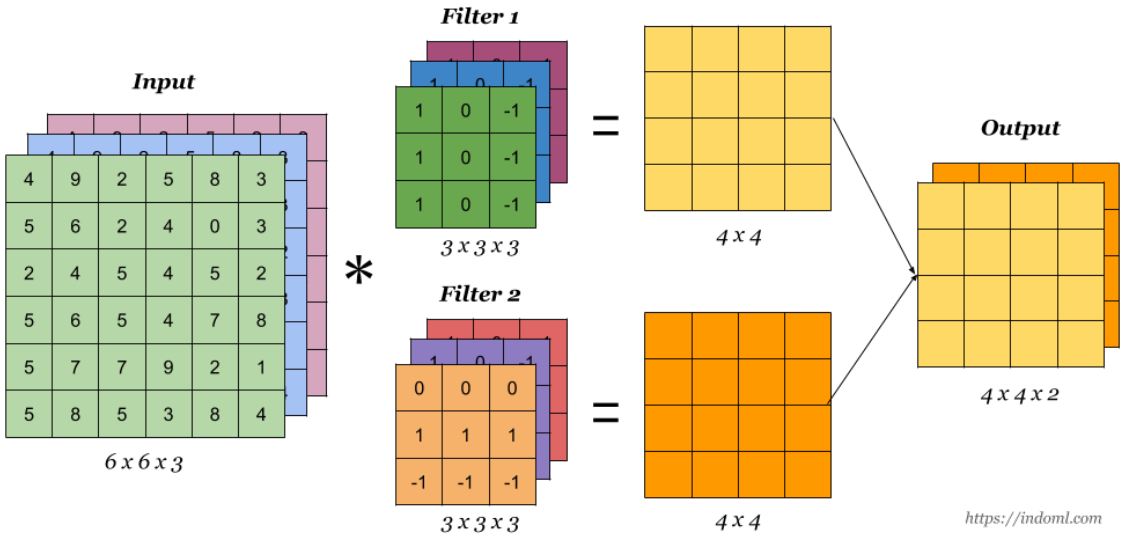

Figure 2.3: A example of 2d convolution operation with 3 input channels and 2 output channels. Picture courtesy: [3]

the example, the input size is $6 \times 6 \times 3$ (e.g., a colored image (i.e., 3 color representations) with a dimensionality of 6×6) and the kernel size is $3 \times 3 \times 3$. The last "3"s in both the input and the kernel are referred to the number of input channels. When a kernel scans across the input, an element-wise product between each element of the kernel and the input matrix is calculated at each location and summed to obtain the output value in the corresponding position of the output. In the example, there are 2 kernels, thus, the output size is $4 \times 4 \times 2$. The number of output channels is 2, the same as the number of kernels. Formally, the kernel size is $(num\_kernel, H_k, W_k, C_{\text{in}})$ where H_k and W_k are the height and width of the kernel. num_kernel and C_{in} are the number of input channels and output channels.

Conv1d layer is a variation of Conv2d and is widely studied [72, 70, 71, 13, 12, 4]. The significant difference between Conv2d and Conv1d is the shape of the kernel. Kernel of Conv1d is $(num\_kernel, W_k, C_{\text{in}})$ where W_k is the width of the kernel. C_{in} and num_kernel are the number of input channels in the input and number of kernels. Other than this difference, the math operation of Conv1d is the same as Conv2d. Thus, we refer readers to the previous paragraph for more details. Similar to the Conv2d layer, the two key

hyperparameters are kernel size and the number of kernels.

2.3.2 Gated Recurrent Unit

In this section, we first present the concept of the recurrent mechanism [66], which is the foundation of all recurrent neural networks (RNNs), and then, present two DNN layers based on the recurrent mechanism: the RNN layer and the Gated recurrent unit [32] (GRU) layer.

The recurrent mechanism [66] is designed to process a variable-length sequence input using hidden states whose activation at the current time depends on the previous time. As shown in Figure 2.4, there is one RNN unit, A, and, at each time step, the unit processes both input of the current step, x_i, and the hidden state from the previous step, s_{i-1}, and then, yields the hidden state of the current step, s_i. The output state is sent to the above

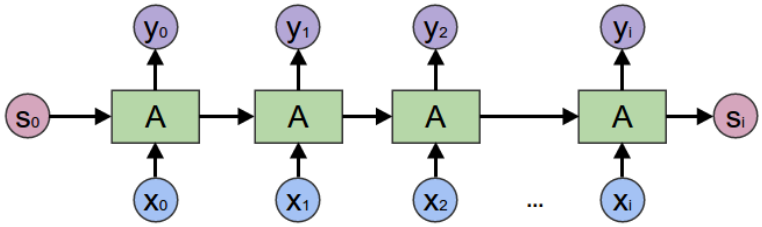

Figure 2.4: An example of the unfolded recurrent mechanism process. A single unit, A, iteratively processes the input x_i and s_{i-1}. Picture courtesy: [1]

layer and sent back to unit A again for yielding hidden state s_{i+1}.

While, in theory, the recurrent mechanism can process any sequence without length limitation, in practice, it cannot handle long sequences well. One practical solution, which was proposed by Schuster et al. [133], is having two recurrent mechanisms and processing a sequence simultaneously in both positive and negative time directions. Figure 2.5 presents how bidirectional recurrent mechanisms process on the same input sequence. One mechanism scans the input sequence from the start to the tail and the other scans the same input from the end to the start. Through concatenating both RNN mechanisms' hidden states which are yielded from the same input x_i, more information related to x_i

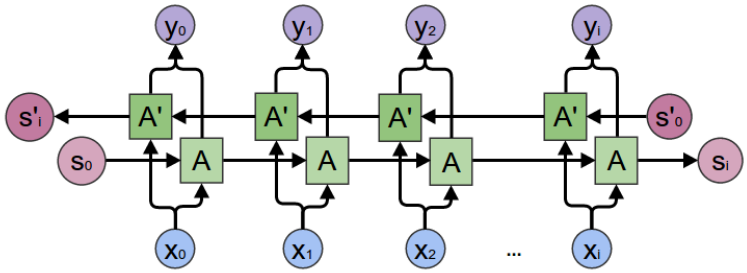

Figure 2.5: An example of the bidirectional recurrent mechanism process. Two RNN units, A and A′, process the same input sequence $x_0, \ldots, x_i$ from opposite directions. y_i is obtained through concatenating both units' outputs of x_i. Picture courtesy: [1]

is preserved in the concatenated state, y_i. Due to the popularity of this solution, putting the word "bidirectional" at the front of any RNN variation refers to using this solution with that RNN variation.

In addition to the bidirectional RNN mechanism, researchers have explored various RNN units (i.e., unit A in previously presented figures.) for modeling sequences. The simplest RNN unit is:

$$s_i = \sigma(W_{in}x_i + W_{sn}h_{(i-1)} + b_n) \tag{2.1}$$

The W_{in}, W_{sn} and b_n are learnable weights. σ refers to the Sigmoid activation function. The hidden state, s_i, contains information from previous hidden state, s_{i-1}, and current input x_i. The initial state, s_0, is a zero vector where all elements are zeros. It means that s_0 contains no information. The s_i is the output, y_i, and is also the hidden state s_i processed by unit A in the coming step (i.e., $i+1$). The output y_i is the same as s_i and s_i is also processed by unit A in the coming step (i.e., $i+1$). The recurrent mechanism that uses this unit is commonly called the RNN layer. The RNN layer is straightforward and is mathematically Turing Complete [69]. But, the RNN layer suffers from the vanishing gradient problem that leads to oscillating weights and unstable learning [60].

The GRU unit is one of the RNN unit variations that was proposed to ease the vanishing gradient problem. The layer that uses the GRU unit is called GRU layer. Following

is the mathematical expression of the GRU unit:

$$r_i = \sigma(W_{ir}x_i + W_{sr}s_{(i-1)} + b_r) \tag{2.2}$$

$$z_i = \sigma(W_{iz}x_i + W_{sz}s_{(i-1)} + b_z) \tag{2.3}$$

$$\tilde{s}_i = f(W_{in}x_i + r_i * (W_{sn}s_{(i-1)}) + b_n) \tag{2.4}$$

$$s_t = (1 - z_t) * \tilde{s}_i + z_i * s_{(i-1)} \tag{2.5}$$

The s_i is the hidden state and x_i is the input at ith step, respectively. $h_{(i-1)}$ is the hidden state of the previous step. $\tilde{s}_i$ is the candidate hidden state. r_i and z_i are the reset and update gates, respectively. σ is the Sigmoid function, and $*$ is the Hadamard product. $f()$ refers to the activation function. The hidden state, s_i, contains compressed information from previous inputs (i.e., $x_1, \ldots, x_{i-1}$). The initial state, s_0, is a zero vector where all elements are zeros. Compared with the RNN unit presented in Equation 2.1, where the new hidden state s_i is a non-linear function of s_{i-1}, the GRU unit explicitly modifies the hidden state s_i through an explicit addition with a candidate hidden state, $\tilde{s_i}$. This ensures the constant error flow in the backpropagation process. The reset gate r_i controls how much information from the previous hidden state s_{i-1} is irrelevant in the future and should be dropped. In other words, the candidate hidden state $\tilde{s}_i$ contains information that is useful to the future. If the reset gate r_i is close to 0, the candidate state $\tilde{s}_i$ would contain information mostly from the current input x_i and ignore most information from the previous hidden state s_{i-1}. The update gate z_i controls how much information from the previous hidden state s_{i-1} would be carry over to the current hidden state s_i. Both r_i and z_i control how the information flows through time. Amodei et al. [7] demonstrated empirically that using the GRU layer in their DNN increases the training speed and reduces the likelihood of training diverges. The DNN used in Chapter 4 uses bidirectional GRU layers.

2.3.3 Attention Mechanism

The attention mechanism layer has become popular in recent years. It has been explored for two use cases: 1) using the attention mechanism to manage memories without forgetting [52]; 2) using the attention mechanism to replace RNN for sequential modeling [155].

Our dissertation is involved in both directions. In Chapter 4, we use an attention mechanism for managing memory. In Chapter 5 and Chapter 6, we use DNNs that utilize attention mechanisms for sequential modeling. In the rest of the section, we present two variations of the attention mechanism one for each use case, respectively.

Luong et al [93] proposed using the attention mechanism to manage encoders' outputs (i.e., memories) in an autoregressive DNN for machine translation tasks. They showed the effectiveness of attention mechanisms on translation tasks (i.e., English to German and German to English). Let the encoder's hidden outputs be a list of vectors $(\bar{h}_1, \bar{h}_2, \ldots, \bar{h}_s)$ where $s = 1, 2, \ldots, S$ and the decoder's hidden outputs be $(h_1, h_2, \ldots, h_t)$ where $t = 1, 2, \ldots, T$. The context vector, which is represented as c_t, is the sum of the encoder's hidden outputs weighted by alignment scores for h_t and is defined by the equations below.:

$$score_{t,s} = \langle h_t, \bar{h}_s \rangle \tag{2.6}$$

$$\alpha_{t,s} = softmax(score_{t,*}) \tag{2.7}$$

$$= \frac{score_{t,s}}{\sum_{s'=1}^{s'=S} score_{t,s'}} \tag{2.8}$$

$$c_t = \sum_{s=1} \alpha_{t,s} * \bar{h}_s \tag{2.9}$$

The $softmax$ converts a vector of numbers into a vector of probabilities that sums up to 1 as expressed in equation 2.8. The function $score_{t,s}$ is referred to a content-based function for which Luong et al [93] consider three different alternatives:

$$score_{t,s} = h_t \bar{h}_s \tag{2.10}$$

$$score_{t,s} = h_t W_\alpha \bar{h}_s \tag{2.11}$$

$$score_{t,s} = W_{\alpha,2} tanh(W_{\alpha,1}[h_t; \bar{h}_s]) \tag{2.12}$$

where W_α, $W_{\alpha,1}$ and $W_{\alpha,2}$ are learnable parameters. This layer is called the global attention layer as it draws dependencies across all encoder's hidden outputs.

The attention mechanism is also explored as a sequence modeling method. Vaswani et al. [155] proposed the global multi-head attention (GMA) layer that provides significant parallelization and is designed to replace RNN to draw global dependencies between input and output. GMA layer is the key component of Transformers [155] which will be presented

in later sections. Let $X_q \in R^{T_q \times d_{model}}$, $X_k \in R^{T_k \times d_{model}}$ and $X_v \in R^{T_k \times d_{model}}$ be input sequences for query, key and value, respectively. T_q refers to the length of the query sequence and T_k refers to the length of the key/value sequence. Vaswani et al. [155] found that, instead of performing query, key and value to a single attention function, it is beneficial to linearly project the query, key and value H times with different learned weight matrices. These groups of projected query, key and value are fed in attention functions in parallel, resulting in H output sequences. These outputs are concatenated and projected by another weight matrix, resulting in the final output that is represented by O. In this dissertation, h refers to the hth attention head. The following shows the details of the GMA layer:

$$Q_h = X_q W_{qh} \tag{2.13}$$

$$K_h = X_k W_{kh} \tag{2.14}$$

$$V_h = X_v W_{vh} \tag{2.15}$$

$$alpha_h = softmax(Q_h K_h^T / \sqrt{d_k}) \tag{2.16}$$

$$Head_h = alpha_h * V_h \tag{2.17}$$

$$O = Concat([Head_1, \ldots, Head_h, \ldots, Head_H])W_o \tag{2.18}$$

The projections are parameter matrices $W_{qh} \in R^{d_{model} \times d_k}$, $W_{kh} \in R^{d_{model} \times d_k}$, $W_{vh} \in R^{d_{model} \times d_v}$ and $W_o \in R^{d_{model} \times d_v}$. The ith attention $alpha_i$ is obtained from linearly projected queries (Q_h) and keys(K_h) through Equation 2.16. The hth attention head generates outputs $Head_h$ through applying the Hadamard product between $alpha_h$ and V_h. Outputs from all attention heads are concatenated through the function $Concat()$ and once again projected through W_o, resulting in the final values. For example, if we have 8 attention heads and the hidden size of the model, d_{model}, is 1024, we define $d_k = d_v = d_{model}/8 = 128$.

Vaswani et al. [155] further distinguished two use cases of the GMA layer with different names: global multi-head cross-attention (GMCA) layer and global multi-head self-attention (GMSA) layer. GMCA layer is used for autoregressive DNNs and manages memory from the encoder for the decoder. That is, the GMCA layer takes the encoder's

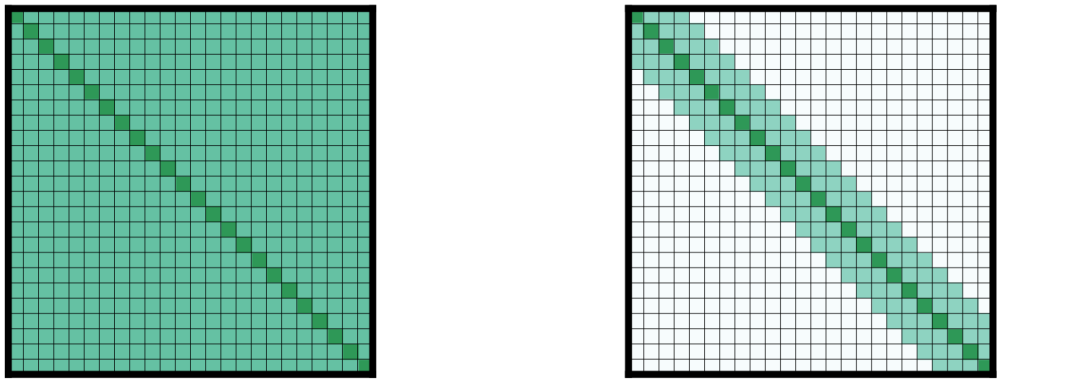

Figure 2.6: The attention region difference between global attention (full n^2 attention) and local attention (sliding window attention). The x-axis is the index of queries and the y-axis is the index of keys. Picture courtesy: [20]

hidden representations as key and value and the output of the previous decoder layer as the query. GMSA layer, on the other hand, is used to replace recurrent neural networks in modeling long-range dependency due to the ability of being parallelized and the path length of the forward and backward signals. In terms of the sequential operation, while the GMSA layer connects all with a constant number of sequentially executed operations, a recurrent layer requires $O(n)$ sequential operations [155]. For the path length between long-range dependencies in a DNN, the shorter these paths between any two vectors in the input and output sequences, the easier it is to learn long-range dependencies [74]. The maximum path length for the GMSA layer is $O(1)$ while the maximum length for any recurrent layer is $O(n)$.

In addition to studying the GMA layer, the local multi-head self-attention (LMSA) layer was proposed to extract contextual representations through drawing dependencies on local receptive fields [124, 33, 127, 20], due to the importance of local context [79]. Figure 2.6 shows the attention region difference between global attention and local attention. The x-axis is the index of queries and the y-axis is the index of keys. For global attention, for any query, attention scores are calculated with every key. However, for local attention,

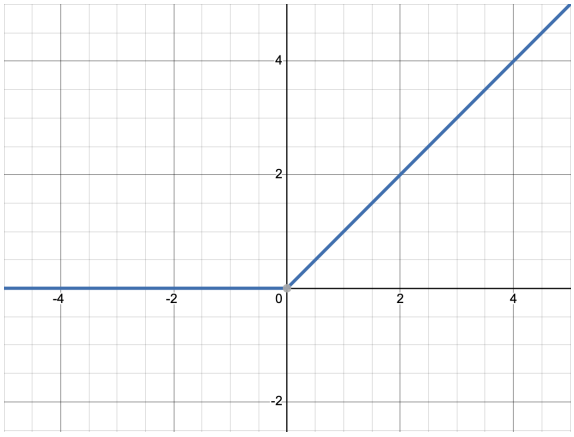

Figure 2.7: This graph shows how the ele_{input} (i.e., x-axis) is transformed into the ele_{output} (i.e., y-axis) by ReLU [110].

attention scores only consider keys close to a query. Window size defines the local region.

2.3.4 The rectified linear unit

The rectified linear unit (ReLU)[110] layer is an activation layer that separately does the non-linear transformation to every element in an input matrix, and increases a DNN's capability to learn complicated tasks. It is commonly used to process output from linear operations (e.g., outputs from Conv2d). It was proposed by Nair et al. [110] for preserving information about relative intensities as the information travels through multiple DNN layers and remained a competitive engineering solution that enables more effective convergence than other types of activation layers (e.g., sigmoid, tanh). The mathematical equation of a ReLu is:

$$ele_{output} = \max(0, ele_{input}) \tag{2.19}$$

where ele_{output} is the output value and ele_{input} is the input value. The function max() outputs the largest value between 0 and ele_{input}. Figure 2.7 shows how ReLU transforms the ele_{input} (i.e., x-axis) to the ele_{output} (i.e., y-axis).

2.3.5 Gaussian Error Linear Units

Gaussian Error Linear Units (GeLU) [58], which was proposed years after ReLU, is a popular activation layer that has been widely used in seq-to-seq DNNs in recent years

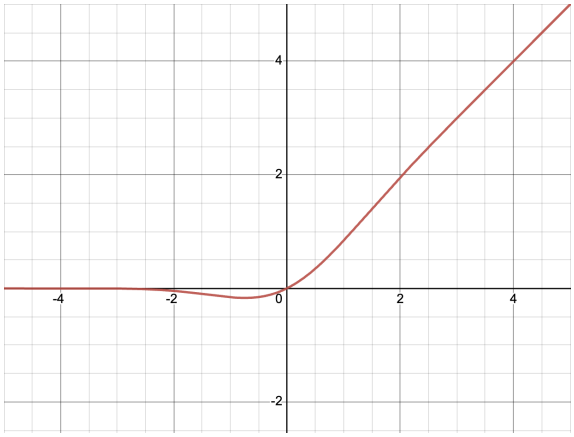

Figure 2.8: This graph shows how GeLU [58] transforms the ele_{input} (i.e., x-axis) to the ele_{output} (i.e., y-axis).

and shows better performance than ReLU. Dan et al. [58] empirically showed that GeLU outperforms ReLU across a range of computer vision, natural language processing, and speech tasks. Like other activation layers, it is also used to process output from linear layers. GeLU's math equation is:

$$ele_{output} = 0.5 * ele_{input} * (1 + \text{Tanh}(\sqrt{(2/\pi)} * (ele_{input} + 0.044715 * ele_{input}^3))) \quad (2.20)$$

where ele_{output} is the output value and ele_{input} is the input value. Figure 2.8 shows how GeLU transforms the ele_{input} (i.e., x-axis) to the ele_{output} (i.e., y-axis). The GELU weights ele_{input} by its values instead of gates by their sign as in ReLU.

2.3.6 Batch Normalization

Batch Normalization [64] (BatchNorm) layer was proposed to reduce internal covariate shift during the training process and can be used to process any layers' outputs. Ioffe et al. [64] evaluated this layer by comparing the difference between a state-of-the-art image classification DNN at that time and the same DNN with BatchNorm layers. The DNN with BatchNorm layers achieves the same accuracy with 14 times fewer training steps as the original DNN and outperforms the DNN model with significantly fewer training steps [64]. This layer is widely used in DNNs, especially in the computer vision field. Let the mini-batch contains N inputs, $B = X_1, X_2, \ldots, X_n, \ldots, X_N$, and the following shows

how BatchNorm normalize this mini-batch:

$$\mathrm{E}_B = \frac{1}{N}\sum_{n=1}^{N} X_n \tag{2.21}$$

$$\mathrm{Var}_B = \frac{1}{N}\sum_{n=1}^{N} X_n \tag{2.22}$$

$$\hat{X_n} = \frac{X_n - \mathrm{E}_B}{\sqrt{\mathrm{Var}_B + \epsilon}} \tag{2.23}$$

$$Y_n = \hat{X_n} * \gamma + \beta \tag{2.24}$$

Given N inputs, the mean E_B and variance Var_B of the mini-batch (i.e., N inputs) are first calculated. Then, all inputs are normalized to have mean 0 and variance 1, which is denoted as $\hat{X_n}$s. ϵ refers to a small number (e.g., 1×10^{-8}) that is used to prevent the denominator from being 0. After $\hat{X_n}$s are scaled and shifted by γ and β, which are two learned parameters, the outputs, Y_ns, are passed to the DNN layers. It is worth noting that, in order to calculate the mean and variance, BatchNorm requires a mini-batch to have more than one input. According to Ioffe et al. [64], the normalized $\hat{X_n}$s are internal to the transformation but their presence is crucial.

2.3.7 Layer Normalization

Layer Normalization [14] (LayerNorm) is a variation of BatchNorm layer and is proposed for sequence modeling. While, empirically, adding BatchNorm to an existing DNN boosts the trained model's performance, it is not obvious how to apply BatchNorm to RNN in which the lengths of inputs are not the same. Jimmy et al. [14] modified BatchNorm into LayerNorm by computing the mean and variance from hidden representations of a RNN layer. In other words, while BatchNorm considers an input sample as the basic unit, LayerNorm considers a hidden representation as the basic unit. LayerNorm directly estimates the normalization of the representations. Thus, it does not require any dependency on the number of training samples (i.e., the size of the mini-batch). The normalization process is the same as the BatchNorm. Jimmy et al. [14] shows that LayerNorm can substantially reduce the training time compared with RNNs without it. This layer is widely used in seq-to-seq DNNs, including all DNNs that we will present in later sections.

2.4 Seq-to-seq DNN Architectures

In this section, we present the general architecture of two seq-to-seq DNNs used in ASR. DNN layers used in these architectures are presented in Section 2.3. We present the general architecture of CNN-RNN DNN in Section 2.4.1. We present the general architecture of Transformer in Section 2.4.2.

2.4.1 CNN-RNN architecture

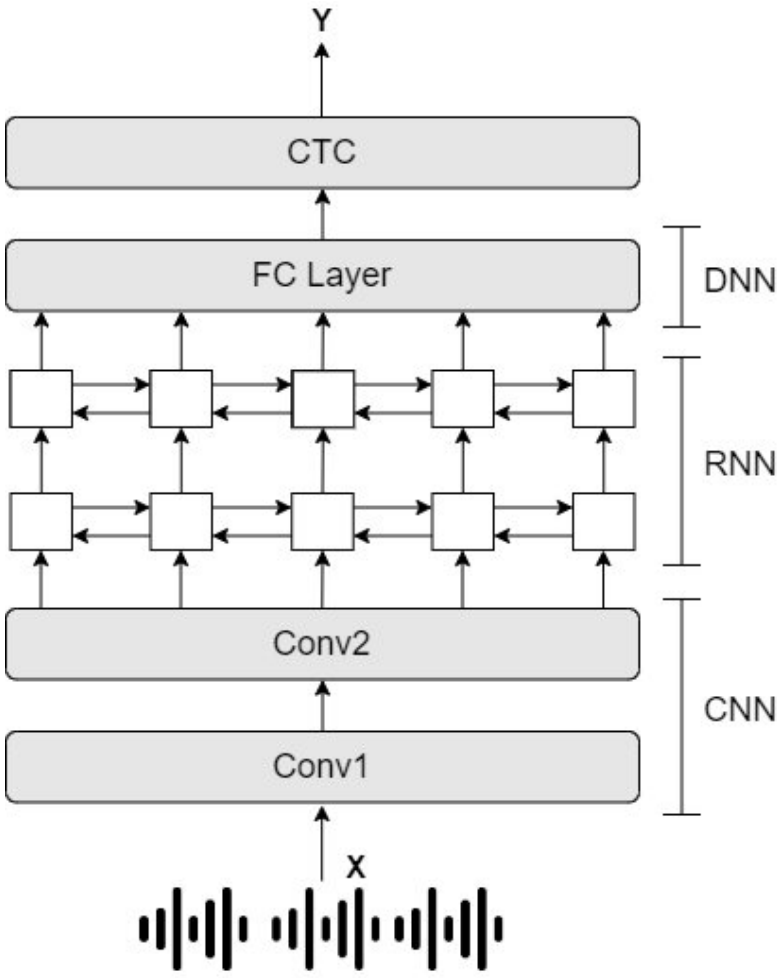

Figure 2.9: The general architecture of CNN-RNN DNN for ASR task. Picture courtesy: [91]

In this section, we present the general idea of CNN-RNN DNN. A CNN-RNN DNN is

a type of DNN architecture that has at least one convolution layer for processing the input and multiple RNN layers to process the output of the last convolution layer. It was first proposed by Donahue et al. [37] for video activity recognition, image caption generation, and video description tasks. Convolution layers capture the spatial information from the input images and RNN layers capture the temporal information from a sequence of image representations [37]. Due to the innate characteristics of CNN-RNN DNN (i.e., mapping an input sequence to an output sequence), it is widely used in sequence-related tasks, such as ASR. In the ASR field, CNN-RNN DNN is used to map the speech signals to transcriptions. Figure 2.9 shows the general architecture. Convolution layers at the bottom serve as feature extractors that extract low-level local acoustic features from the input sequences (i.e., sequences of audio representations) and serve as a sub-sampling method that shortens the length of input sequences and is essential to make RNN computationally tractable with high-sampling rate audio [7]. RNN layers above convolution layers take the sub-sampled low-level acoustic features as input and map features to labels/transcriptions through modeling long-range dependencies. Long short-term memory [60] (LSTM) and GRU are two popular RNN layers, and, in the ASR field, GRU is more popular than the other. Bidirectional GRU is commonly used to model the long input sequences, which is the exact case in ASR. The output layer is a linear layer (i.e., FC layer in Figure 2.9 that maps the hidden representation to labels). The detailed architecture of a CNN-RNN DNN depends on the characteristics of tasks and researchers' experiences. We will present a well-known CNN-RNN architecture (i.e., DeepSpeech2) used in this dissertation in Section 2.9.1.

2.4.2 Transformers

In this section, we present the achievements of Transformers and two types of Transformers: an autoregressive Transformer and a non-autoregressive Transformer. Since Transformers was first proposed and evaluated on NLP tasks, certain modifications are made when adapting them to transcription tasks (i.e., ASR). Thus, in this section, we focus on presenting general architectures of Transformers as the foundation for Section 2.9.2 and 2.9.3, where we present Transformer-based architectures for ASR in detail.

An autoregressive Transformer has an encoder and a decoder, and both consist of

multiple blocks which are stacked multi-head attention layers and linear layers. The overall architecture is shown in Figure 2.10. The encoder block contains a multi-head

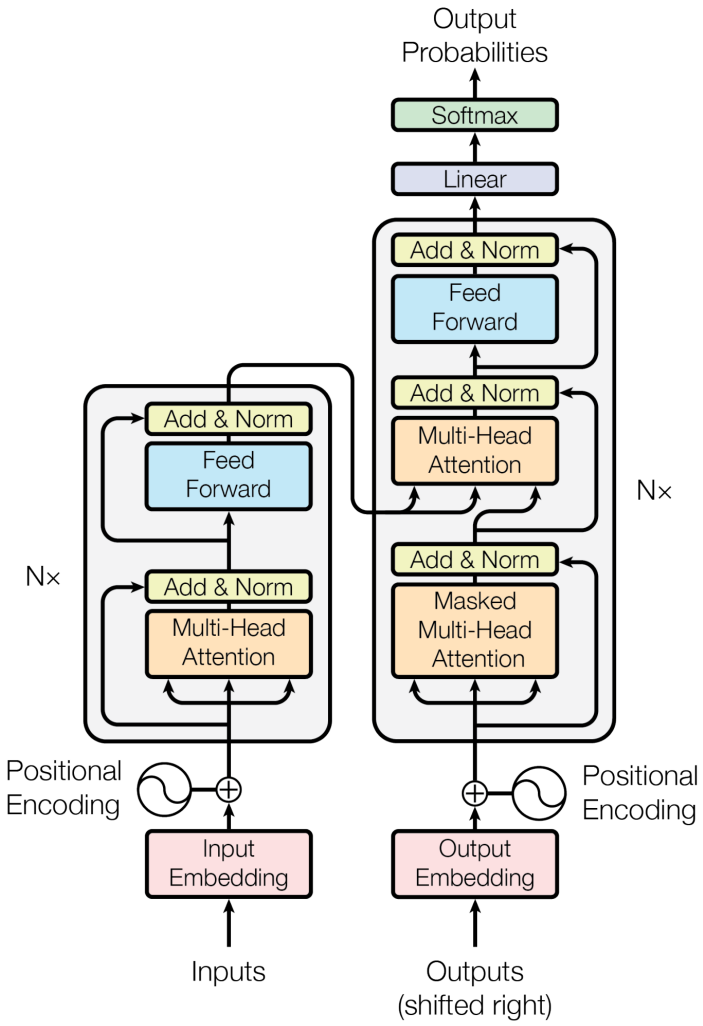

Figure 2.10: The general architecture of autoregressive Transformer. Picture courtesy: [155]

self-attention layer and two linear layers. On the other hand, the decoder block contains a multi-head self-attention layer, a multi-head cross-attention layer and two linear layers. Moreover, to ensure that the decoder only relies on known outputs to predict current output, a mask is deployed during training. In order to ensure the awareness of the order of the sequence, Vaswani et al. [155] add positional encoding to the input embeddings.

Except for the autoregressive Transformer, researchers also developed non-autoregressive

Transformers. Figure 2.11 shows the general architecture that was first proposed by Devlin et al. [35]. Devlin et al. [35] designed the non-autoregressive Transformer to learn

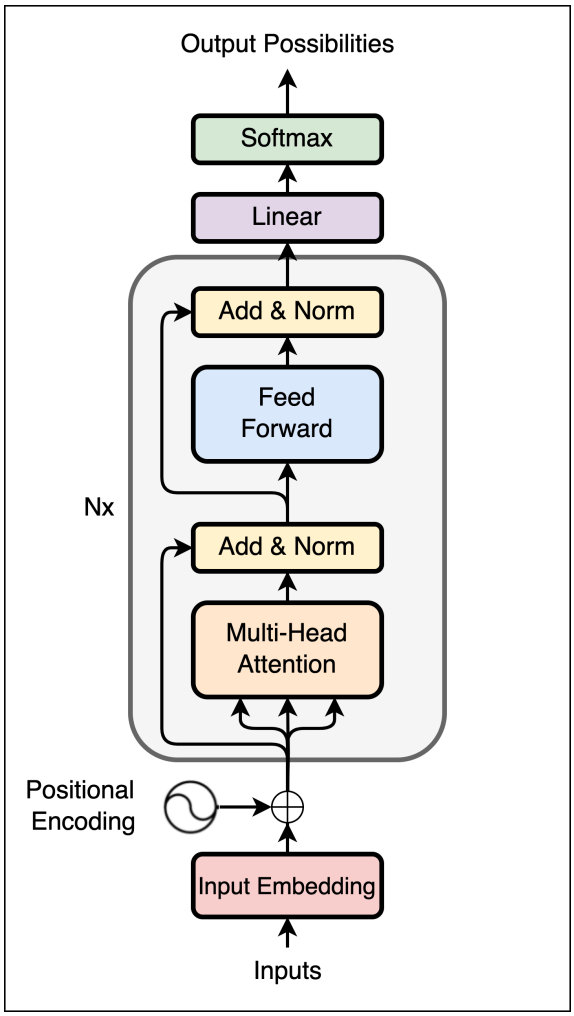

Figure 2.11: The general architecture of non-autoregressive Transformer. We follow similar color codes as Vaswani et al. [155]

bidirectional representations through joint conditioning on both left and right contexts in all layers. The main difference between these two types of Transformers is that the non-autoregressive Transformer excludes the multi-head cross-attention mechanism. Wav2Vec 2.0 [17], which is one of the popular ASR architectures, utilizes the non-autoregressive Transformer.

2.5 DNN Learning Methods

In this section, we present general ideas of various learning methods that are used in this dissertation. In Section 2.5.1, we present supervised learning. In Section 2.5.2, we present contrastive learning. In Section 2.5.3, we specify the definition of weakly-supervised learning that we use in our research.

2.5.1 Supervised learning

Supervised learning refers to learning methods that train a DNN with labeled data. Different tasks will have different input and output label pairs. For ASR, the pair consists of an audio recording (i.e., input) and the corresponding transcription (i.e., output labels). The learning method guides a DNN to map the input to the output as accurately as possible. The loss functions that are used with the supervised learning method measure the similarity between predicted labels and the groundtruth labels. It is a well-studied area and has been widely used in various tasks. In Section 2.6, we present commonly used loss functions for ASR for supervised learning.

2.5.2 Contrastive Learning

Contrastive learning has become a prominent method for training large models and it also enlarges the size of training data in the fields of natural language processing, computer vision and ASR. The general process is: first, train a large DNN model to learn hidden representation through deploying contrastive learning on unlabeled large datasets, and then, adapt the model to downstream tasks through deploying supervised learning and transfer learning on labeled but small datasets. The core idea of contrastive learning is learning representation through comparing similar/dissimilar samples, which will be presented in the next paragraph. Enlarging the data size is one of the benefits that contrastive learning provides because no label is required.

The basic hypothesis of contrastive learning is that the quality of a DNN model's representation can be approximated through measuring the representation's performance on separating similar and dissimilar input samples. Unlike supervised learning methods

that measure a model's ability on mapping input samples to output labels, contrastive learning measures how easily a model's hidden representation can pull similar samples together while pushing dissimilar samples away. In other words, the goal of contrastive learning is: given a function that measures the distance between two representations, the distance of two similar samples should be close, while, at the same time, the distance of two dissimilar samples should be far away. In general, a loss function that belongs to the contrastive learning method contains two parts: measurement functions for measuring the distance and selecting methods that pick similar (positive) and dissimilar (negative) samples.

The contrastive loss function that is widely used in ASR is proposed by Wav2Vec 2.0 [17]. Since the selecting method is bounded with the architecture of Wav2Vec 2.0, the loss function will be presented in Section 2.9.2, where the architecture of Wav2Vec 2.0 is presented.

2.5.3 Weakly-supervised learning

Unlike supervised learning and contrastive learning, which have generally agreed definitions, weakly supervised learning covers a wide range of techniques. In this dissertation, weakly supervised learning refers to training a model with a supervised loss function on a large dataset where the quality of labels is low (i.e., more labeling/transcription errors than found in typical publicly available data sets). The key challenge of weakly supervised learning is how to balance the quality and quantity of data.

In the ASR field, researchers [44, 54, 166] have shown that training an ASR through the combination of multiple datasets shows higher robustness and generalizability. However, due to the limitation of high-quality data (i.e., few transcription errors), the size of training data still is much smaller than the unlabeled dataset, which can be more than 1,000,000 hours [166]. To enlarge the size of labeled data, researchers reduce the requirement of data quality in exchange for quantity. Chen et al. [54] selected 10,000 hours acceptable data from podcasts, YouTube and audiobooks using sophisticated automated pipelines, and demonstrated that a model trained with large data outperforms the model's siblings who are trained with smaller datasets. To have even larger training dataset, Galvez et al. [44]

selected data from The People's Speech [2] and showed that the model trained with 20,000 hours of data achieves acceptable performance on out-of-domain (OOD) testing set (i.e., Librispeech [117]). However, the data size is still much smaller than the unlabeled dataset. Radford et al. [122] closed the gap by expanding the data quantity to 680,000 hours through collecting data from the Internet and adopting multilingual and multitasking (i.e., audio transcription and translation) training. They showed that models trained on large datasets with low label quality can outperform models trained on relatively small datasets with high label quality as long as the former can balance the data quantity and quality. In addition to demonstrating the power of joint multilingual and multitask training, Fadford [122] published models used in their research providing a firm ground for study transfer learning. Detail of Radford's DNN models, which are called Whispers, will be presented in Section 2.9.3.

2.6 Loss Functions

In this section, we introduce supervised loss functions that are commonly used in ASR field and this dissertation. In Section 2.6.1, we introduce Connectionist Temporal Classification [51] (CTC) loss, which is a popular loss function in ASR field. In Section 2.6.2, we introduce cross-entropy (CE) which is a classic loss function that has been around for decades.

2.6.1 Connectionist Temporal Classification

Connectionist Temporal Classification [51] (CTC) loss is designed to resolve the length mismatch between the input sequence and the output sequence. More specifically, it resolves the problem that input sequences (e.g., audio representations) are longer than output sequences (i.e., transcriptions) which is a characteristic of various tasks (e.g., ASR). To resolve the length mismatch problem, CTC utilizes placeholders (i.e., special blank labels) and the repetition of labels to expand the output sequence to be the same length as the input sequence. This means that there is more than one expanded sequences for any output sequence. For example, let the input sequence's length to be 5 and the

output sequence to be "cat". Following CTC's rules, the expanded sequences can be "c_aat", "ccaat", "c__at" and so on, where "_" represent the black label. Since all possible expanded sequences are valid outputs, CTC loss maximizes the probability of all possible expanded sequences during training.

To formally describe the loss function for training, let the input speech sequence be X, and the output label sequence be Y. The function $B(Y)$ refers to the expanding process and $\tilde{Y}$ is mapping to the set of all possible expansions of Y. The total probabilities of all expanded sequences related to Y is:

$$P(Y|X) = \sum_{\tilde{Y} \in B(Y)} P(\tilde{Y}|X) \tag{2.25}$$

Moreover, the CTC also assumes that the outputs of the network at different times are conditionally independent. Thus, the $P(\tilde{Y}|X)$ is calculated through:

$$P(\tilde{Y}|X) = \prod_{t=1}^{T} P(\tilde{y}_t|X) \tag{2.26}$$

where t refers to the index of output labels and there are T labels in $\tilde{Y}$. $\tilde{y}_t$ is the tth output label in $\tilde{Y}$. In practice, since the training process should minimize the loss function, CTC loss is defined as a form of negative log probabilities to minimize the loss (i.e., maximize the probability) and Equation 2.25 is transformed into the following form:

$$L_{CTC} = -lnP(Y|X) \tag{2.27}$$

The training process minimizes L_{CTC}.

The inference/decoding process generates the most likely sequences/transcriptions given input sequences:

$$Y^* = \arg\max_{Y} P(Y|X) \tag{2.28}$$

Due to CTC's hypothesis of label independence, the Y^* usually contains orthographic errors. Researchers found that having an external language model eases the situation. The language model can be a n-gram model or a DNN-based model. After including a language model to the inference process, the process is expressed as:

$$Y^* = \arg\max_{Y} P(Y|X) * \alpha P(Y) \tag{2.29}$$

where α is a hyperparameter that decides the importance of the language model and $P(Y)$ refers to the probability of sequence Y in the language model.

2.6.2 Cross Entropy

Cross-entropy (CE) is a loss function used in classification tasks and it is commonly used in autoregressive DNNs. CE measures the distance between a softmaxed output from a DNN and a one-hot encoded vector (i.e., the encoded groundtruth label) where the number of the elements of the vector is the total number of classes I (e.g., potential tokens in ASR). The loss is the sum of the negative logarithmic probabilities and the logarithmic value is used for numerical stability:

$$CE = -\sum_{i=0}^{I} y_i \log y_i^* \tag{2.30}$$

where y_i^* is the likelihood of the ith class predicted by a DNN and y_i is the groundtruth of the class. When using CE to train a seq-to-seq DNN, since the DNN yields a sequence of outputs, the function is commonly rewritten as:

$$CE = -\frac{1}{TN}\sum_{t=0}^{T}\sum_{n=0}^{N}\sum_{i=0}^{I} y_i \log y_i^* \tag{2.31}$$

where T refers to the total number of elements in a sequence and N refers to the total number of input sequences that are trained together. Empirically, choosing to average over all sequences and outputs ensures all samples contribute to the gradient equally regardless of the length of the sequence.

2.7 DNN Transfer Learning

In this section, we present the general idea of transfer learning. Transfer learning is a model adaptation method that is widely used in data-scarce scenarios. The core idea of transfer learning is reusing the learned knowledge of a pre-trained DNN model for other domains. In other words, instead of training a DNN with random weights, transfer learning refers to training a DNN whose weights have been updated with certain training datasets on another dataset. The dataset can either be the same tasks as the pre-trained model or different ones. The training process is the same as the one presented in Section 2.1 except the

learning rate is empirically smaller than the latter. In addition to adapting a model trained on one domain/dataset to another, nowadays, transfer learning is also used to adapt a large model that learns a universal representation of natural language to downstream tasks (e.g., question answering, commonsense reasoning, sentence similarity, textual entailment) [123, 35]. The same idea has also been explored in image processing [73], text-to-speech [8, 9, 169, 168, 167] and ASR [17, 131].

Weight transfer learning, which is also known as fine-tuning, and hidden linear transfer learning are three well-studied approaches. 1) Weight transfer learning initializes the target model with a pre-trained base model where both models share the exact same DNN architecture. Then, tune the entire or part of the DNN with a small learning rate [137, 154, 149]. 2) Hidden linear transfer learning builds upon a hypothesis where similar tasks share common low-level features. It uses the pre-trained model as a feature extractor to extract low-level features from input data [86]. The feature extractor is frozen during training. Researchers have a new DNN to learn high-level features for different domains/tasks. In practice, researchers may use both ideas together. For example, first train a new output layer by adopting hidden linear transfer learning's hypothesis, and then, further fine-tune the entire DNN model. 3) Factorized Hidden Layer Adaptation is a structured parameterization of the hidden layers's weights and was proposed as a speaker/domain-adaptation training method. It adapts weights of a pre-trained model with a linear interpolation of a set of bases [130, 140]. The bases are represented by low-rank matrices and shift DNN's weights based on the input of speaker embedding [140]. For all transfer learning methods, a key ingredient to a successful model adaptation is a well-trained base model from a similar task that has learned from relatively large and yet diverse training samples [163, 46].

2.8 Data Representation for ASR

In this section, we introduce data representation methods used in seq-to-seq DNN-based ASRs. Since an ASR takes audio representation as input and outputs transcriptions in text form, we will introduce the commonly used audio representation and text/token

representation in this section.

2.8.1 Audio Representations

In this section, we present the common audio representation forms used in seq-to-seq ASR: waveform and spectrogram. We mainly follow the introduction from Section 2 and Section 4 in digital speech processing [121].

Waveform

In speech production, the information to be transmitted is encoded in the form of a continuously varying (analog) waveform that can be transmitted, recorded, manipulated and ultimately decoded by a human listener. The fundamental analog form of the message is an acoustic waveform, which is known as the speech signal. Figure 2.12 shows the waveform at the top. Since the waveform is the fundamental analog form and is the input for generating other engineered signal representations (e.g., the spectrogram which will be introduced below), using DNN to learn to represent waveform is one of the research directions in acoustic-related research fields. In the ASR field, which is one of the acoustic-related research fields, researchers use DNNs to directly extract information from the waveform [17, 131].

Spectrogram

The spectrogram is a basic tool for understanding how the sounds of speech are produced and how phonetic information is encoded in the speech signal. The spectrogram is derived from the waveform using a fourier transform, and so it has the same information that the waveform has but is expressed in terms of frequency decomposition. The length of the window has a major effect on the spectrogram image. The upper spectrogram in Figure 2.12 was computed with a window size of 10 ms (i.e., each frame of the spectrogram encodes 10 ms of waveform). The lower spectrogram in Figure 2.12 was computed with a window size of 40 ms (i.e., each frame of the spectrogram encodes 40 ms of waveform). In practice, the window size decides a spectrogram's characteristics of resolution. If the window size is short (i.e., the upper spectrogram in Figure 2.12), each individual pitch

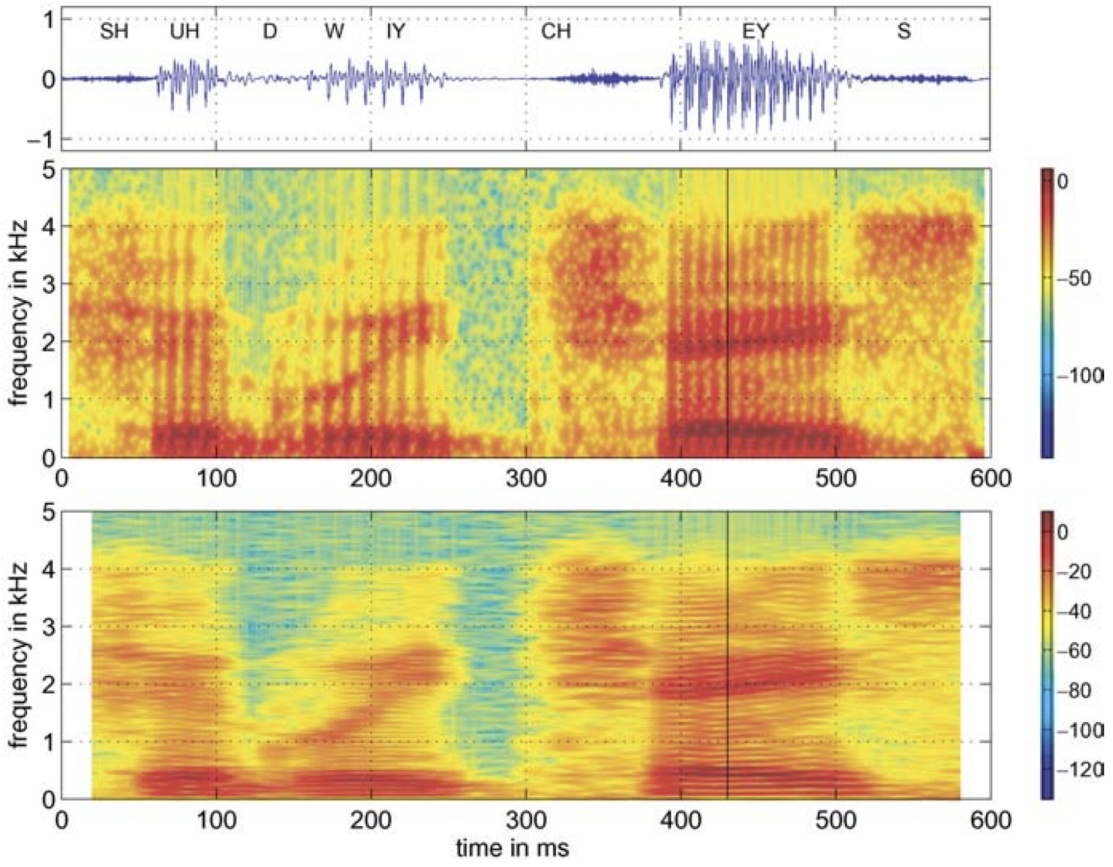

Figure 2.12: Examples of a waveform and two spectrograms with different window sizes. Picture courtesy: [121]

period is resolved in the time dimension, but the resolution in the frequency dimension is poor. On the other hand, if the window length is long (i.e., the lower spectrogram in Figure 2.12), the generated spectrogram will have good frequency resolution but poor time resolution. In other words, this spectrogram is not sensitive to rapid time variations but the good at revealing frequency changes. Compared to the waveform, the spectrogram is a more popular choice among ASR researchers. The spectrogram can reduce the sequence length of representing an audio signal. This advantage can not only reduce the hardware burden of training a model but also increase the inference efficiency for models that have RNN layers.

2.8.2 Tokenization

Tokenization refers to the text preprocesses step that transforms the string of text into a sequence of numbers that can be processed by a DNN. The program that does the process is known as the tokenizer. In this section, we present three tokenization methods that are commonly used in the DNN field.

Word-based tokenization

Word-based tokenization takes words as the basic units (i.e., tokens) and the tokenizer gives each word a unique number (i.e., index), which starts from 0 to the total number of unique words. The downside is that this method generates a huge number of tokens. Take English as an example: there are over 500,000 words and word frequencies are different. If a researcher encodes all these words with a word-based tokenizer, the size of tokens is 500,000 even though some words may only appear once in the corpus. To balance the token size and the model performance, researchers usually keep N most frequent words, where N is decided by researchers' experience, and mark the rest words as out-of-vocabulary (OOV). Other word-based tokenizers for other languages follow the same idea to reduce the token size. The downside is that balancing the token size and model performance is challenging. This tokenization method is not used in this dissertation.

Character-based tokenization

Character-based tokenization takes characters as the basic units and each word is represented as a sequence of characters. Thus, the token size is much smaller than the size of the word-based tokenization. The token size is the total number of characters in that language plus the total number of special symbols. For example, if a researcher uses CTC [51] as the loss function, the blank symbol is one of the special symbols. This tokenization method is widely used in the ASR field when using CTC as the loss function. The downside of character-based tokenization is that, for a DNN, learning meaningful representations for characters is harder than learning meaningful representations for words.

Subword-based tokenization

Subword-based tokenization takes informative character strings as the basic units and the informativeness of a string depends on the learning algorithm and the training corpus. Its core idea is that frequently used words should be retained, but rare words should be decomposed into meaningful subwords. Byte-Pair Encoding (BPE) [158] is one of the widely used tokenizers. It first initializes tokens with characters and represents each word as a sequence of these tokens and a special end-of-word token, which serves as the between-word boundary. Then, the program counts the co-occurrence of token-pair permutations and replaces the most frequent one (e.g., "X" and "Y") with a new token (i.e., "XY"). This process is repeated multiple times with the newly created token and existing tokens until the token size equals the desired size which is decided by researchers' experience and preference. This tokenizer can ensure that a model can learn meaningful representations for high-frequent words and does not need OOV to represent unseen words.

2.9 Seq-to-seq DNN-based ASR Systems

In this section, we focus on presenting DNN architectures from the perspective of developing ASR. In recent years, remarkable progress has been achieved in the ASR field and the state-of-the-art DNN-based ASR changes during these years. Our research closely follows the progress, thus, we present multiple DNN architectures that are used in our research. We present a CNN-RNN DNN known as DeepSpeech2 [7] in Section 2.9.1. Since we use its pre-trained model for transfer learning research in Chapter 4, we introduce the specific pre-trained model and its training data. In Section 2.9.2, we present a non-autoregressive Transformer, Wav2Vec 2.0 [17]. We introduce the architecture of Wav2Vec 2.0 used in the Chapter 5 and present the training process in detail. In Section 2.9.3, we present an autoregressive Transformer, Whisper [122]. We introduce its architecture, its sequence encoding process that is designed for multi-task training and the pre-trained models. We use Whisper in Chapter 6.

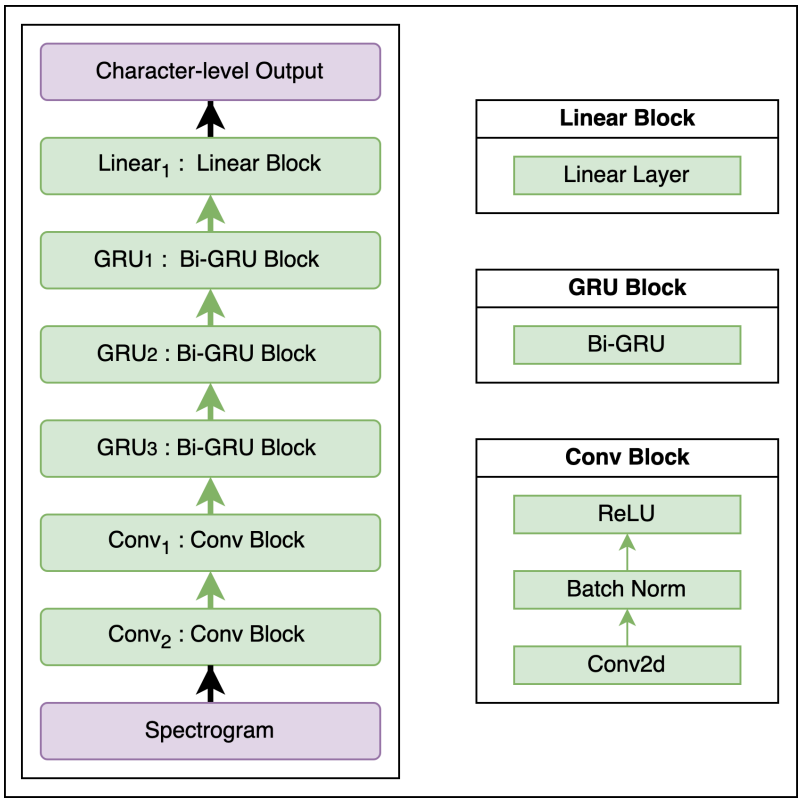

Figure 2.13: The architecture of DeepSpeech2 and the blocks in detail.

Block Name	Channel Size	Kernel Size	Stride	Padding Size
$Conv_1$	32	(11, 41)	(3, 2)	(5, 20)
$Conv_2$	32	(11, 21)	(1, 2)	(5, 10)

Table 2.1: Configuration of DeepSpeech2's Conv blocks.

2.9.1 DeepSpeech2

In this section, we present the architecture of DeepSpeech2. More specifically, we present the architecture of the open-sourced model that is provided by Amodei et at. [7] which we use this model in Chapter 4. We present the data processing step, the architecture and the training/inference process, respectively.

DeepSpeech2 [7] is a seq-to-seq non-autoregressive DNN, that maps the spectrogram to the sequence of 26 English letters. The model was trained with 8000 hours of transcribed private audio recordings. Transcriptions are encoded by a character-based tokenizer. All audio recordings are forced aligned with corresponding transcriptions and segmented into

clips no more than 7 seconds [7]. Then, these clips are transformed into a spectrogram with a window of 20 milliseconds and a stride of 10 milliseconds. Last, the spectrogram of each audio clip is normalized to be between -1 and 1 with approximately zero mean.

DeepSpeech2 takes the spectrogram as input and utilizes Conv blocks for low-level information processing and GRU blocks for high-level processing. The output layer is a linear layer. Figure 2.13 shows the architecture with configuration details. It contains two Conv blocks, and each block is a stack of a Conv2d, a BatchNorm and a ReLU layer, respectively. Table 2.1 presents the configuration of convolution layers in the Conv Blocks. There are three GRU blocks and each contains a bi-GRU layer and the GRU layer is slightly different from the GRU layer introduced in Section 2.1. Amodei et al. [7] found that applying LayerNorm [14] on the transformed X improves the model performance. The modified GRU is:

$$r_i = \sigma(LayerNorm(W_{ir}x_i) + W_{sr}s_{(i-1)} + b_r) \tag{2.32}$$

$$z_i = \sigma(LayerNorm(W_{iz}x_i) + W_{sz}s_{(i-1)} + b_z) \tag{2.33}$$

$$\tilde{s}_i = f(LayerNorm(W_{in}x_i) + r_i * (W_{sn}s_{(i-1)}) + b_n) \tag{2.34}$$

$$s_t = (1 - z_i) * \tilde{s}_i + z_i * s_{(i-1)} \tag{2.35}$$

where s_i is the hidden state and x_i is the input at ith step, respectively. $s_{(i-1)}$ is the hidden state of the previous step. $\tilde{s}_i$ is the candidate hidden state. r_i and z_i are the reset and update gates, respectively. σ is the Sigmoid function, $*$ is the Hadamard product and $f()$ refers to the activation function. In other words, instead of having a LayerNorm before GRU, Amodei et al. [7] make the LayerNorm part of their GRU. After the GRU blocks, the Linear block maps the hidden representation into character-based outputs.

Amodei et al. [7] use CTC [51] to train their models. In the inference process, DeepSpeech2 [7] utilizes a beam search approach [160] to search for the transcription with the highest probability based on the combination of probabilities from the model and a separate n-gram language model. We refer to Section 2.6 for more detail about the inference process related to CTC [51].

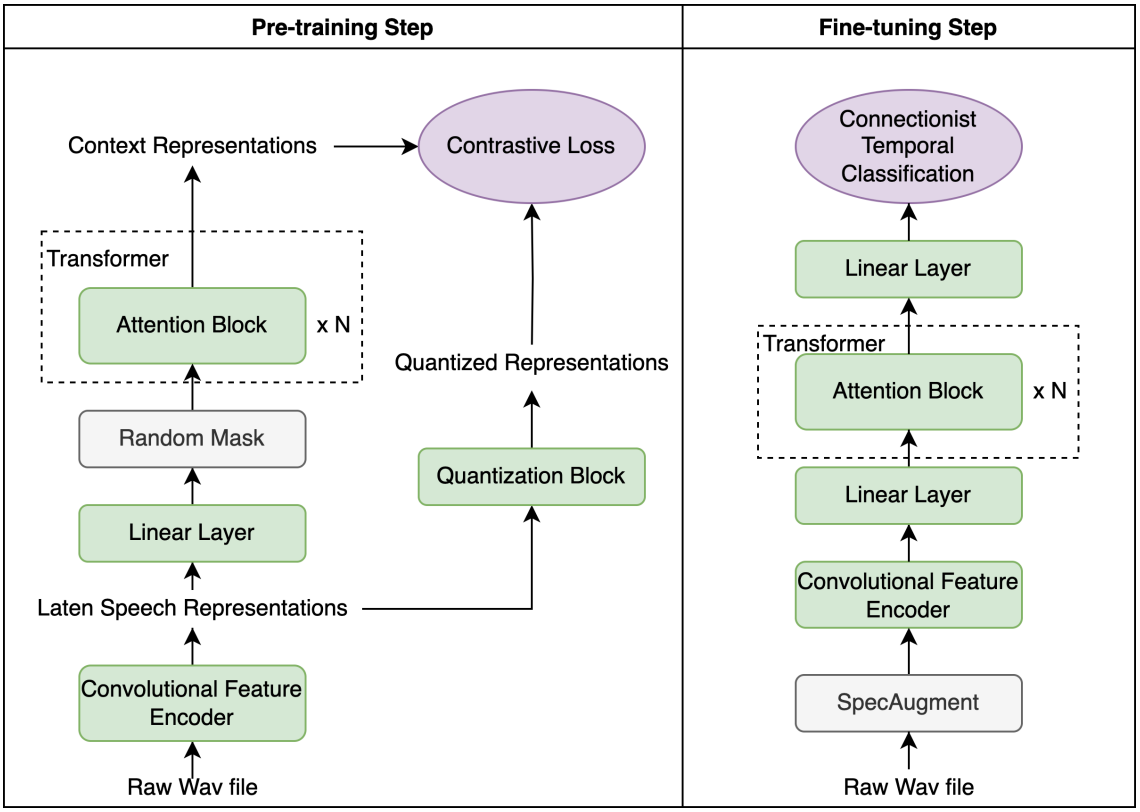

Figure 2.14: The general architecture of Wav2Vec 2.0 and its training process.

2.9.2 Wav2Vec 2.0

In this section, we present the seq-to-seq non-autoregressive DNN architecture: Wav2Vec 2.0. [1] We present the data processing step, the DNN architecture, the training process along with loss functions, and the inference process.

Wav2Vec 2.0 takes 1,8000Hz raw waveform as input and the waveform is normalized to zero mean and unit variance. A character-based tokenizer is used to encode transcriptions.

The DNN architecture is shown in detail in Figure 2.14. The architecture contains three main components: a convolutional feature encoder, a quantization block, and a non-autoregressive Transformer. The feature encoder extracts latent representation from raw audio input. The left graph of Figure 2.15 shows the Conv1d block, which is a stack

[1] We closely follow the source code published at github.com/facebookresearch/fairseq by Baevski et al. [17].

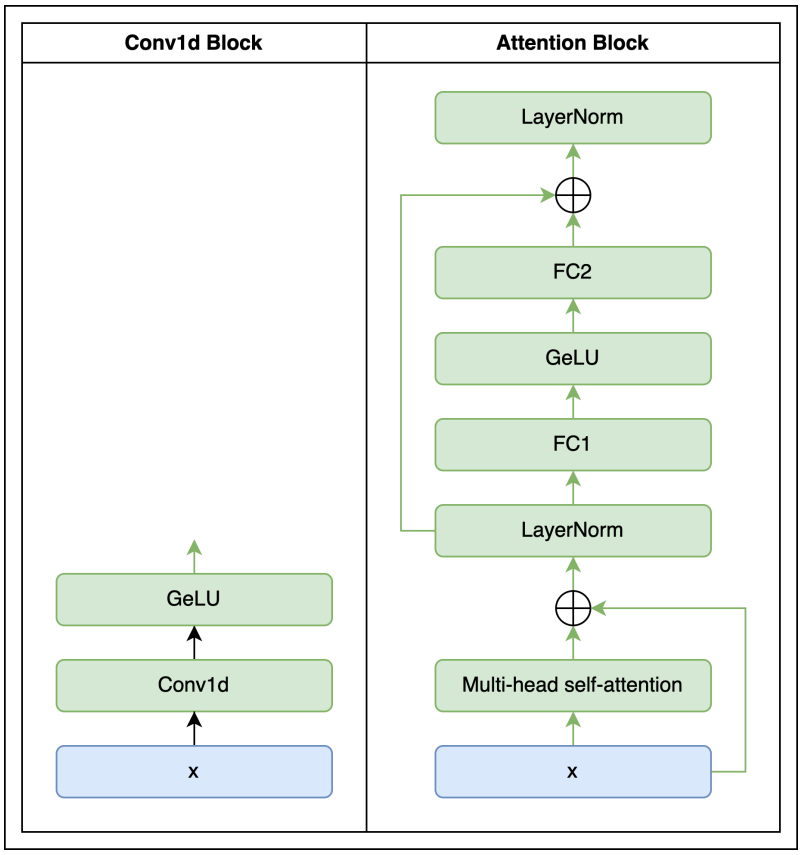

Figure 2.15: The detail architecture of Conv1d block and Transformer block.

of a Conv1d, a LayerNorm and a GELU layer. The quantization block discretizes latent representation to a finite set of quantized representations. Baevski et al. [17] choose to quantize the latent representation due to its good performance on Wav2Vec 1.0 [131]. The non-autoregressive Transformer, which consists of multiple attention blocks, transforms the latent speech representation into content representations. Figure 2.15 shows the attention block in detail. Each block consists of a multi-head self-attention mechanism and two fully connected (FC) layers. The first residual connection adds up the input and the output of the global multi-head self-attention mechanism and a LayerNorm layer follows the connection (i.e., normalized output). Then, the normalized output is transformed by two FC layers (i.e., FC1 and FC2). The second residual connection adds up the normalized output and the transformed output and another LayerNorm layer follows this connection.

Next, we present the training process. Following Baevski's terminology, the training

process consists of two steps: the pretraining step and the fine-tuning step. At the pre-training step, Baevski use contrastive learning which is presented in Section 2.5.2. The model randomly masks some parts of the latent speech representation, and the transformer must reconstruct these masked parts. Thus, no labeled data is required. Contrastive loss [17] is used to evaluate the similarity between a reconstructed representation and the matched quantized representation whose corresponding latent representation is masked. After the convolutional feature encoder extracts latent representations from raw waveform, Wav2Vec 2.0 first uniformly samples multiple latent representations and their adjacent frames and creates a copy of the latent representation sequence with these frames masked. A sampled frame's index (i.e., its time step t) refers to the corresponding masked part. Then, the Transformer generates the content representations of masked parts (i.e., c_t) given the masked sequence as input and the quantization block generates the quantized representation (i.e., q_t) given the no-mask sequence as inputs. Following is the equation of the contractive loss used by Baevski:

$$L_{contrastive} = -log\frac{exp(sim(c_t, q_t)/k)}{\sum_{q_i \in Q} exp(sim(c_t, q_i)/k)} \tag{2.36}$$

where $exp()$ is the exponential function. q_t refers to the masked quantized representation of time step t and c_t refers to the corresponding reconstructed content representation. Q stands for a collection of q_t where all q_ts are uniformly sampled from the same utterance. $sim()$ refers to the function that measures the similarity of two representations. Baevski chooses the cosine similarity, $sim(a, b) = \frac{a^T b}{\|a\|\|b\|}$, as the measurement function. The loss function encourages the Transformer to reconstruct each masked part accurately while discouraging a reconstructed part from being similar to all other masked parts. That is, the similarity between the reconstructed representation, c_t, and its corresponding quantized representation, q_t, must be high, while the similarity between the reconstructed representation, c_t, and other quantized representations (e.g., q_{t-2} and q_{t+4}) must be low. After the pre-training step, the pre-trained model is fine-tuned with labeled data and adopts CTC [51] as the loss function. A character-based tokenizer is used to encode recordings' transcriptions.

In the inference process, Wav2Vec 2.0 [17] utilizes a beam search approach [160] to

search for the transcription with the highest probability based on the combination of probabilities from the model and a separate n-gram language model. We refer to Section 2.6 for more detail about the inference process related to CTC [51].

2.9.3 Whisper

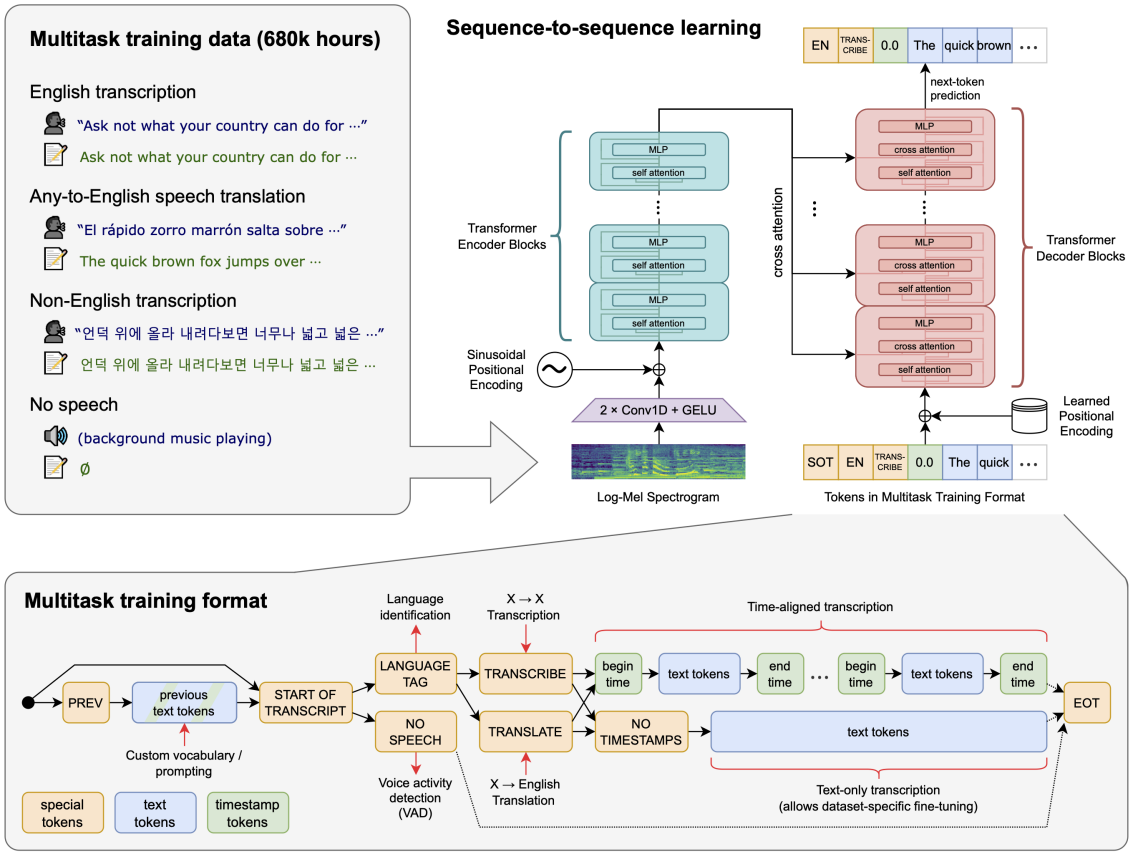

Figure 2.16: The general architecture of Whisper and the text formatting rules. Picture courtesy: [122]

Whisper [122] refers to a group of models trained with the weakly supervised learning method and are autoregressive Transformers [155]. In this section, we first present the training data and the data processing step including data representation and the text formatting. Then, we present the DNN architecture in detail. Third, we present the training process and the inference process. Fourth, we present the pre-trained models

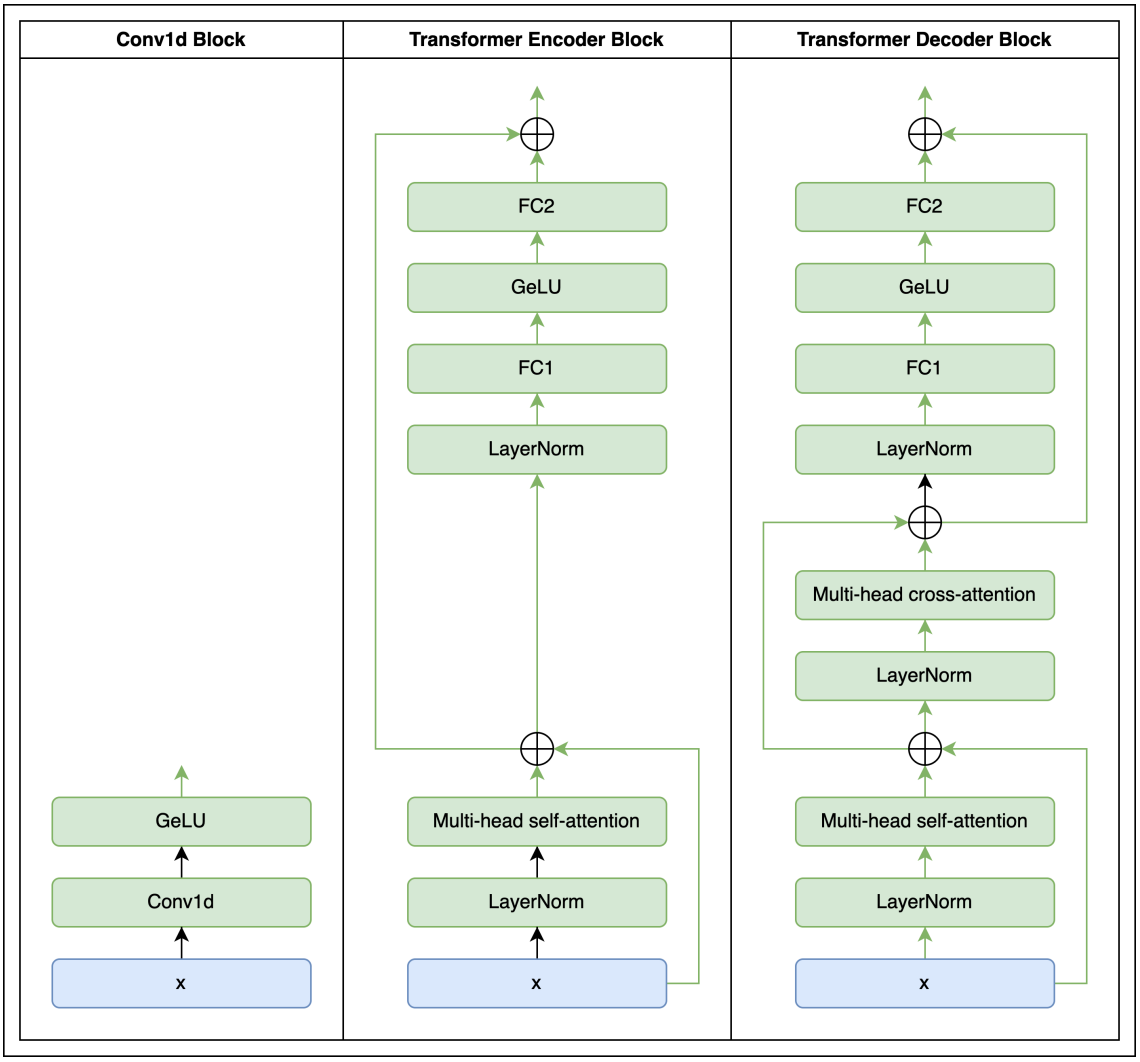

Figure 2.17: The detailed architecture of Conv1d block and Transformer encoder/decoder blocks.

provided by Radford et al. [122]. Last, we briefly introduce our research goal for Chapter 6.

Radford et al. [122] collected data (i.e., audio recordings and corresponding transcriptions) from the Internet. In total, they collected 680,000 hours of recordings to be training data. There are three types of recordings: 1) 438,000 hours of English recordings and transcriptions. 2) 117,000 hours of recordings and transcriptions from 96 non-English languages. 3) 125,000 hours of non-English recordings with English translations.

Radford et al. [122] segment recordings into clips and each clip is less than 30 seconds.

Then, these clips are re-sampled into 16,000 Hz and converted into an 80-channel spectrogram where the window size is 25-millisecond windows with a stride of 10 milliseconds. All spectrograms are normalized to be between -1 and 1 with approximately zero mean which is similar to the audio processing step of DeepSpeech2 [7]. Transcriptions are encoded by a sub-word tokenizer (i.e., BPE [135]).

Usually in multitask learning, each task is handled separately. In other words, there are multiple output layers one for each task. Instead, Radford et al [122] use a single output layer and propose a format that unifies all tasks' output formats, including voice detection, transcription (i.e., ASR), translation, and alignment. The graph at the bottom of Figure 2.16 shows the formatting rules. Radford et al. [122] define that prediction starts with a special token (i.e., `<|startofstranscript|>`). Following the start token, the next token is either `<|nospeech|>`, which indicates no speech in the input, or `<|{language}|>`, where `{language}` is a variable that can be any predefined language abbreviations, such as Chinese (`zh`) or English (`en`). The following token marks the transcription/translation tasks (i.e., `<|transcribe|>` or `<|translate|>`). The next predefined token specifies the intention of not predicting time alignment (i.e., `<|notimestamps|>`). Only text tokens follow this token. If `<|notimestamps|>` is not specified, text tokens are separated by predefined tokens of timestamps that mark the token's beginning and end in the audio file. At the end, the token `<|endofstranscript|>` marks the end of the sequence. To increase the performance on inferring audio longer than 30 seconds, Radford et al. [122] add `<|PREV|>` following with previous tokens to indicate that the inference was conditioned on previous texts.

Figure 2.16 top right shows the general architecture. In the encoder, the spectrogram input is first processed by 2 Conv1d blocks. After the Conv1d blocks, the extracted representation is added with sinusoidal positional encoding and is then fed to encoder blocks. The outputs from the top encoder block are viewed as the acoustic representation. In the decoder, both acoustic representations and tokens, which are encoded by BPE [135] and augmented with learned positional encoding, are fed to the decoder blocks. The outputs from the top decoder block are hidden representations that are used to make next-token predictions.

Figure 2.17 presents all three types of blocks in detail: Conv1d block, Transformer encoder block and Transformer decoder block. Conv1d block is a stack of a Conv1d layer and a GeLU layer. An encoder block consists of a global multi-head self-attention mechanism and two linear layers. First, the LayerNorm followed by a global multi-head self-attention mechanism generates the output from the input and the first residual connection adds up the input and the output (i.e., residual output). Second, the residual output is transformed by two linear layers after it is normalized by another LayerNorm layer. The second residual connection adds up the normalized output and the transformed output and another LayerNorm layer follows this connection. The transformer decoder block consists of a global multi-head self-attention mechanism, a multi-head cross-attention mechanism and two linear layers. First, the LayerNorm layer followed by a global multi-head self-attention mechanism generates the self-attention output from the input and the first residual connection adds up the input and the output (i.e., residual self-attention output). Second, the LayerNorm layer followed by a global multi-head cross-attention mechanism generates the cross-attention output from the self-attention output and the second residual connection adds up the self-attention output and the cross-attention output (i.e., residual cross-attention output). Third, the residual cross-attention output is transformed by two linear layers after it is normalized by another LayerNorm layer. The third residual connection adds up the normalized output with the transformed output and outputs the block's output.

To train a model, Radford used the cross entropy loss over tokens. The losses of previous tokens are ignored. The inference process is similar to the standard autoregressive DNN, the decoder takes the encoder's outputs, previous tokens and a dummy start symbol (i.e., `<|stratofstranscript|>`) as input and predicts tokens iteratively.

Radford et al. [122] published models with different parameter sizes, shown in Figure 2.18, as well as their code. Most models are trained with similar configurations, except whisper-large-v2 which is not presented in the figure. This is an additional model that Radford et al. [122] trained with a different training configuration and the model outperforms whisper-large-v1. Radford et al. [122] show that, with a good training configuration, whisper-large-v2 is more robust than Wav2Vec 2.0. That is, Whisper-large-v2

Size	Parameters	English-only model	Multilingual model
tiny	39 M	`tiny.en`	`tiny`
base	74 M	`base.en`	`base`
small	244 M	`small.en`	`small`
medium	769 M	`medium.en`	`medium`
large	1550 M	N/A	`large`

Figure 2.18: Information of pre-trained models offered by Radford et al. [122]. English-only models refer to models trained with the English-only subset. Picture courtesy: [122]

achieves lower WER on various testing sets than Wav2Vec 2.0.

Radford et al. [122] hypothesize that the robustness of large whisper models is mainly contributed by the powerful language model (i.e., decoder). This is an important hypothesis that needs to be proved because it is directly related to the transfer learning strategy (e.g., we do need to tune the decoder?) that researchers might choose. In Chapter 6, we present our research on confirming this hypothesis.

2.10 Mild Cognitive Impairment

Mild cognitive impairment (MCI) is an intermediate stage between normal cognition and dementia. Since there is still no cure available for Alzheimer's Disease, the most dominant form of dementia, detecting the MCI Distinguishing older people with MCI from those with normal cognition through applying machine learning and feature engineering to speech data is a research topic that we conduct in this dissertation.

In the medical field, cognitive test batteries, such as the Montreal Cognitive Assessment (MoCA) [111] and Mini-Mental State Exam (MMSE) [41], quantitatively measure a person's memory, executive functions, visuospatial and language ability. While those

batteries provide information to clinicians for diagnosing cognitive status, the requirement of specialists constrains using these batteries for daily monitoring. Using language and speech characteristics that can be automatically extracted by computer to diagnose a person's cognitive status, especially MCI stage, is of significant public health importance.

Prior literature has identified speech and language abnormalities are induced by cognitive impairment [147]. Progressive decline in linguistic ability [125], syntactic deficits [145], and abnormal patterns of spoken words used in daily conversations [11] have shown to be effective in distinguishing MCI from those with normal cognition (NC). Research on oral-based tasks, such as reading tasks, has shown promising results in both descriptive [102, 95] and predictive studies [96, 76, 42]. Focusing on the prosodic aspects of speech, Martin et al. [95] found more flat intonation in speech samples of older adults with Alzheimer's disease compared to those with NC. A similar study also showed less fluency among persons with MCI compared with those with NC [102]. In the context of predictive modeling, Toth et al. [150] showed the utility of speech-based measures (e.g., speech rate) extracted from continuous unstructured speech samples in classification models developed for the early detection of MCI. Similar findings have also been reported in samples collected from structured tasks, such as picture description and verbal fluency. Meilan et al. [102] showed that the average duration of silence segments was strongly correlated with cognitive test scores in a picture description task. Konig et al. [77] and Chen et al. [28] showed older adults with AD were slower to retrieve words at the beginning of the fluency test than NC. Past studies also showed promising results in identifying cognitive status [162, 148], stress [47, 22], depression [115] and aging [109, 89].

Chapter 3

Using ASR in Feature Extraction for Cognitive assessment

Unlike the following chapters in which we focus on developing ASR techniques, in this chapter, we utilize ASR as a tool to extract handcrafted time-base features from the animal fluency (AF) test to improve cognitive assessment. In this study, we have 1 hour of audio recordings in total from 70 subjects (28 with mild cognitive impairment and 42 demographically matched normal controls). Off-the-shell ASR systems did not perform well on the tiny dataset and this dataset is also too small to fine-tune a pre-trained ASR. Thus, we utilized an ASR to extract timestamps of transcribed words (i.e., forced alignment), which is widely used in phonetic studies [19, 81, 142] and focus on improving the cognitive assessment. We propose time-base features based on early cognitive studies on the AF test and utilize these features to distinguish older people with mild cognitive impairment (MCI) from those with normal cognition (NC).[1]

3.1 Introduction

Among cognitive tests, the AF test has been widely used as part of several dementia screening batteries. Participants are asked to retrieve as many animal names as possible in a short amount of time, typically one minute. In clinical practice, clinicians evaluate the

[1]This chapter is based on: L. Chen, M. Asgari, R. Gale, K. Wild, H. Dodge, and J. Kaye, "Improving the assessment of mild cognitive impairment in advanced age with a novel multi-feature automated speech and language analysis of verbal fluency," Frontiers in Psychology, vol. 11, p. 535, 2020 [28]. Liu Chen conducted the research reported in this chapter with discussion and advice from the other authors.

brain's ability on semantic retrieval through counting the total number of retrieved unique animal names. Typically, a person with MCI retrieves fewer names than those with normal cognition (NC). In addition to clinical practice, researchers are investigating the pattern of retrieval to both improve our understanding of MCI and improve cognitive assessment. Additionally, researchers are also developing fully automatic cognitive assessment systems to make them widely available.

In this preliminary study, we take a step toward improving the classification accuracy and making the assessment automatic. While previous research focused on representing the pattern with count-based features/measures to improve cognitive assessment, we propose time-based features inspired by Hills et al. [59]. We use forced alignment [99] to obtain the timestamps of the animal names and extract the time-based features. The classification improvement, which is obtained from combining time-based features with count-based features, demonstrates the potential of utilizing time-based features to distinguish older people with MCI from those with NC. Moreover, we develop an automatic assessment pipeline to extract features and make assessments/classifications.

In the rest of this chapter, we present the background information in Section 3.2. Computational methods are presented in Section 3.3, including count-based features and our proposed time-based features. In Section 3.4, we present the pipeline including data preprocessing, feature extraction, feature selection, and classification. In Section 3.5, we describe the experiment setup, including the dataset and evaluation criteria. We present our results in Section 3.6 that show the effectiveness of our time-based features. Finally, Section 3.7 is the discussion section.

3.2 Background

In this section, we present the background for this chapter. In Section 3.2.1, we review verbal fluency tests including the AF test. In Section 3.2.2, we review one of the retrieval theories for the AF test, namely the clustering-switching theory. In Section 3.2.3, we review the study of the optimal searching strategy, which is rooted in the clustering-switching theory. This strategy inspired our work.

3.2.1 Verbal Fluency Tests

Verbal fluency (VF) tests are cognitive tests that are widely used in dementia screening batteries. In a VF test, participants are asked to name as many words in a category (e.g., animals) as possible in a short duration of time, typically one minute. The VF test is administered in two different ways: 1) *semantic fluency*, in which participants are asked to generate words from a semantic category such as animals, fruits, or vegetables, and 2) *phonemic fluency* where participants must generate words that begin with a particular letter such as "F" or "S". In the conventional scoring of VF tests, the count of uniquely generated words in the test comprises the final score. Prior research suggests that verbal fluency is a function of an individual's age regardless of cognitive functioning with younger populations perform better in this test compared to older adults [6, 146]. Within older adults, Farina et al. [39] highlight that individuals with MCI achieve lower VF scores than those with NC. Among various semantic fluency tests, the AF test is one of the widely used tests for evaluating a person's cognitive status and is used in multiple research studies related to the human brain's semantic retrieval process. It is for these two reasons that we use it in this chapter.

3.2.2 Semantic Retrieval Process

The semantic retrieval process refers to how human beings retrieve words that match certain semantic requirements. Asking a person to retrieve animal names (i.e., the AF test) evokes this process as would any of the semantic-fluency VF tests. Previous research indicates that the retrieval pattern gives more information about the semantic retrieval process than the conventional score of the AF test. Troyer et al. [151] proposed a computational approach for characterizing the semantic retrieval process during an AF test that revolves around two sub-processes (known as two-part memory retrieval): clustering, in which a participant retrieves words that share some subcategories, and switching, in which a participant switches to a different semantic subcategory for retrieving new words. Troyer used two algorithms for identifying when a person switches to a new cluster: an

algorithm for quantifying the semantic similarity of any two adjacent words, and an algorithm for determining clusters and switches. Count-based features are used to represent the retrieval pattern.

To quantify the semantic similarity of animal names for switching and clustering, Troyer et al. [151] first manually developed a structured table that categorizes 545 animal names into 22 subcategories that cover a particular cluster of related animals (e.g., domestic animals, birds, etc.). Animal names in the table are not exclusive to a single subcategory and can be part of up to four subcategories. Next, Troyer segmented each subject's answer into multiple clusters and words in a cluster share at least one subcategory. Last, Troyer represented each participant's retrieval pattern with count-based features (i.e., the number of switches and the average number of words in a cluster). They showed that young participants generated more words and switched more frequently as compared to older participants. Moreover, their later research [152] verified the effectiveness of these two features for discriminating a group of participants with Alzheimer's disease from demographically matched participants with normal cognition.

Despite the potential usefulness of these features, there is a limitation that is related to computing switching and clustering patterns from the preexisting manually constructed table of animal names. The assignment of multiple subcategories to a word can cause ambiguity in the determination of subcategory switches [161]. To improve on this, Woods et al. [161] proposed a computational method, explicit semantic analysis (ESA), based on the semantic relatedness of subsequent words computed in a vector space by cosine similarity distance given the vector representation of words. To derive the clusters, firstly, Woods detects the occurrence of a switch by comparing the pairwise cosine distance of two successive words to a predefined threshold and, if the similarity is lower than the threshold, there is a switch between these words. Words between two switches form a cluster. Woods defined the threshold to be a participant's retrieved words as 75% the mean semantic similarity (i.e. cosine similarity). That is, the exact threshold value varies among examiners.

Both research studies pursued the same goal with different algorithms: using switching and clustering to describe the semantic retrieval pattern in the AF test. In our work, we

evaluate both algorithms/methods from these studies and refer to them as Troyer- and ESA-based methods.

An example: We present an example in the top half of Figure 3.1 to explain how the ESA-based method segments an AF answer into multiple clusters. The top table in Figure 3.1 displays a partial response of a subject to the AF test, {*cat, falcon, bat, elephant, shark, dolphin*}, and the corresponding cosine similarities between previous and current words $d(W_{i-1}, W_i)$. According to Woods et al. [161], the higher the cosine similarity, the stronger the relationship between the pair of words. Practically, the cosine distance of two words is compared to a predefined threshold value – a critical factor in the success of the ESA method. In this example, the 75% of average semantic similarity is 0.05, which is the threshold value for this example. The word pair of *cat* and *falcon* is semantically different because $d(cat, falcon) = 0.01$ is lower than the threshold. In this way, the partial response has two switches, resulting in 3 clusters.

Count-based features: In the previous paragraph, we introduced an example of using the ESA-based method to segment an AF answer into multiple clusters. After the clusters have been computed, count-based features can be extracted. In addition to the 2 features used by Troyer et al. [151], we use a compilation of features used by various researchers for decades. We use code from Woods et al. [161] to extract these features. We present these features in Table 3.1.

3.2.3 Optimal Searching Strategy

In the previous section, we introduced the clustering–switching hypothesis for the AF test and reviewed algorithms that segment retrieved words into clusters. Working with the clusters and switches, Hills et al. [59] investigated the search strategy of retrieving animal names through analyzing the cost of retrieving a name. Hills et al. adopted the marginal value theorem (MVT) [24] for characterizing the search strategy observed in individuals' AF test responses. Traditionally employed to model the foraging behavior of animals, MTV optimizes the benefit-cost ratio, the estimation of whether it is more

Animal Name	dog	cat	falcon	bat	elephant	shark	dolphin
Cosine distance		0.07725	0.0123	0.05299	0.05505	0.00748	0.06719

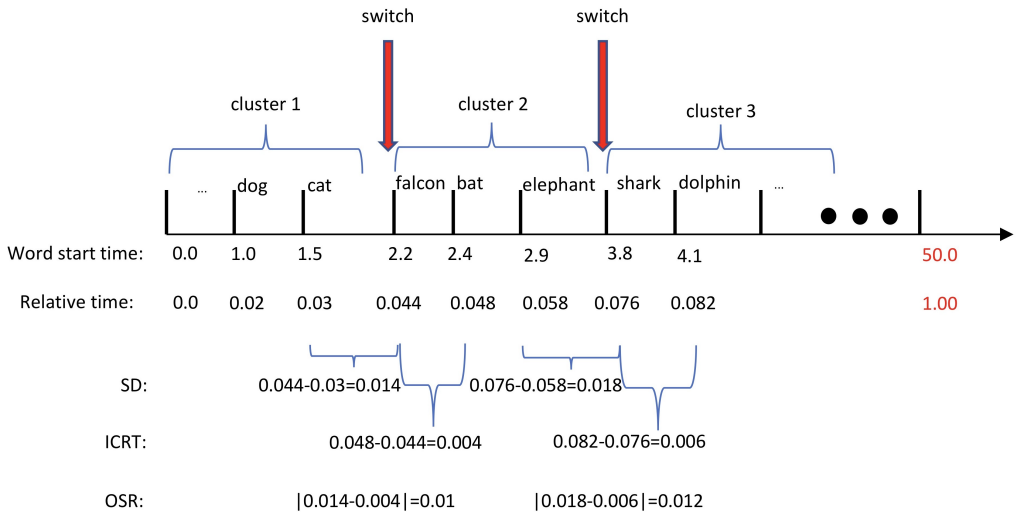

Figure 3.1: In this example, we set the threshold for this subject to be 0.05. Based on the threshold, there are two switching positions, which are marked as red arrows. The *SD* of the first switching is 0.7 which is the time difference between *falcon* and *cat*. The *ICRT* of the first switching is 0.2 which is the time difference between *bat* and *falcon*. So the OSR of the first switch is 0.5 which is the absolute difference between the *SD* and *ICRT* that we just calculated.

beneficial to continue searching for food at the current patch of food versus expending the effort to move some distance with the hope of discovering a more bountiful patch. Hills et al. hypothesized that this optimization problem in animals' foraging strategy resembles the semantic-retrieval strategy in the AF test in which one may retrieve more words over the course of the test if optimally choosing when to switch to a new cluster. They chose retrieval time (RT), which is the time cost of retrieving a new word and has been one of the popular methods to study VF tests [78, 92, 63, 59, 103, 156, 120, 98], to represent the retrieval cost. Hills demonstrated that their participants (college students) took significantly longer time to retrieve the first word in a cluster than the mean retrieval time over all retrieved names. That is the retrieval time for retrieving a word that is semantically similar to the previous word is shorter than retrieving a semantically different word. They also showed that retrieval time increases as more words are retrieved from the

Feature Name
The total number of unique animal words (standard AF score)
The total number of switches (NS)
The average number of words in clusters (ANWC)
The total number of unique words
The total number of duplicate words
The mean of the log of word frequency
The standard deviation of word frequency
The mean of words' syllables
The standard deviation of words' syllables
The mean of words' typicality
The standard deviation of words' typicality
The mean of ESA of adjoining words (MESA)
The mean of ESA between every word and every other word in the answer (MAESA)
The ratio between MESA and MAESA
The total number of switches
The average number of words in clusters
The total number of single-word clusters

Table 3.1: Count-based features that have been well-studied by various researchers.

same semantic region and participants who switched either too early or too late lead to retrieving fewer words than others. Our research is inspired by Hills' work and we describe our time-based features in Section 3.3.

3.3 Features for Characterizing Verbal Responses

In Section 3.2.2, we introduced the core of the computational method for characterizing verbal responses that revolve around the switch and cluster components. In this section, we describe two sets of features: our count-based and time-based features. In the following sections, we introduce a novel count-based feature and continue to explain the example presented in Section 3.2.2 to explain our time-based features.

3.3.1 Addional Count-based Features

Empirically, we noticed numerous cases in our dataset where a single word appeared in its own cluster. To capture this phenomenon, we introduce a new feature that measures the ratio between the count of single word clusters to the total number of clusters and refer

to it as *single cluster ratio* (SCR).

3.3.2 Time-based Features

In the previous section, we presented our novel count-based feature, SCR. In this section, we present our three time-based features.

Our approach for characterizing the search strategy is based on the *marginal value theorem (MVT)* [24] and the work of Hills et al. [59]. We hypothesize that individuals with normal cognition optimally switch to a new cluster leading to the production of more animal names while individuals with MCI are less capable of finding optimum transition points. With a poor switching strategy, a person either lingers too long in a cluster or moves too fast to a new cluster; ultimately producing fewer animal names.

Based on this hypothesis, we define two intermediate time-based measurements to capture the difficulty of retrieving a new word in an AF test. First, *Switching duration (SD)* is the elapsed time, which is the time difference between these two adjacent animal names' start times, between the last animal name in a cluster and the first animal name in the next cluster. For example, in Figure 3.1, the animal name "cat" is the last name in cluster 1 and the animal name "falcon" is the first name in cluster 2. Second, *Intra-cluster retrieval time (ICRT)* is the elapsed time between the first two animal names in a cluster (e.g., dog and cat in Figure 3.1). Along with the sequence of words, Figure 3.1 shows the sequence of timestamps {*1.0, 1.5, 2.2, 2.4, 2.9, 3.8, 4.2*} representing the onset of each word measured in seconds and the relative timestamps (i.e., rescaled timestamps) are presented under the actual timestamps {*0.0, 0.02, 0.03, 0.044, 0.048, 0.058, 0.082*}. We normalize the actual timestamps with the overall duration of the answer in order to contain the value to be lower than one which makes it easier to use in machine learning. Since all participants have one minute to give their answers, no information is lost in the rescaled timestamps. From the timestamps along with the ESA-derived switches and clusters, one can compute these features and the following is the SD of the first switch

and ICRT of the second cluster:

$$SD_1 = (t_{falcon} - t_{cat}) = 0.014 \tag{3.1}$$

$$ICRT_2 = (t_{bat} - t_{falcon}) = 0.004 \tag{3.2}$$

Note that the proposed method for measuring ICRT is only appropriate if subjects produce at least two animal names in a cluster. We utilize the SCR, which is described in the previous section, to capture information related to single-word clusters.

Next, we use SD and ICRT features to craft another feature, *optimal switch rate* (OSR), that estimates the success of a switch by measuring the difference between the switching duration and intra-cluster retrieval time of the first switch as follows:

$$OSR_1 = |SD_1 - ICRT_1| = 0.01 \tag{3.3}$$

where || is the absolute value operator. Our assumption is that the more successful a switch, the smaller the OSR. As the number of switches and clusters may vary across responses, the length of the SD and ICRT feature vectors vary accordingly. Thus, we use standard statistical aggregates (i.e., mean, median, variance, minimum and maximum) to represent participants' overall performance on optimal switching rate. In total, we have five candidate features. In practice, this proposed computational approach will face a few limitations to be addressed in Section 3.4.2.

3.3.3 Feature Selection

There are a large number of count- and time-based features. Training a statistical model with all features may lead to overfitting and result in a lack of generalizability. To select features that are both informative for classification and supported by early related clinical research, we use both manual feature selection and an automatic feature selection algorithm to select informative features. Three count-based features are expected to be kept because their effectiveness has been reported by various early research. AF score is the standard scoring method for animal fluency tests in clinical practice. ANWC and NC have been constantly evaluated by researchers since Troyer et al. [151] proposed the

clustering and switching retrieval process. Additionally, since SCR is our proposed count-based feature designed to preserve edge cases that OSR cannot capture, we expect it to be preserved. For time-based features, we expect some of the features to be preserved.

We use a feature selection algorithm known as recursive feature elimination with cross-validation (RFECV) [55], which ranks the importance of features based on a given scoring function and returns a subset of features to select the most informative features. This technique is provided by the Scikit-learn toolkit [118]. It constructs a smaller subset of features and calculates the model performance given each remaining subset. The elimination process continues until all features are exhausted and the feature set that maximizes the model performance across all feature sets is selected as the best-performing feature set. Then, among the selected features that are count-based, we select those features that either have been shown effective in early research or designed for time-based features, and we keep selected time-based features. Among the selected features that are time-based, we keep all selected features. Eventually, we have six selected features: AF score, ANWC, NC, SCR, mean OSR, and median OSR. We will use these selected features in Section 3.5.

3.4 Training Pipeline

In previous sections, we presented the count and time-based features. We present the pipeline of our cognitive assessment system as shown in Figure 3.2. The time alignment module (Section 3.4.1) takes the audio recordings and corresponding transcriptions as inputs and output time steps for each word. The normalization module (Section 3.4.2) removes all non-animal-name words and unifies synonyms. Time- and count-based features are extracted based on methods presented in Section 3.2.2 and Section 3.3. The machine learning module (Section 3.4.3) takes count- and time-based features from verbal responses to distinguish participants with MCI from those with NC.

3.4.1 Forced Alignment

Obtaining retrieval time is the foundation of both the research described in Section 3.2.3 and our time-based features. In early research, some researchers manually annotated the

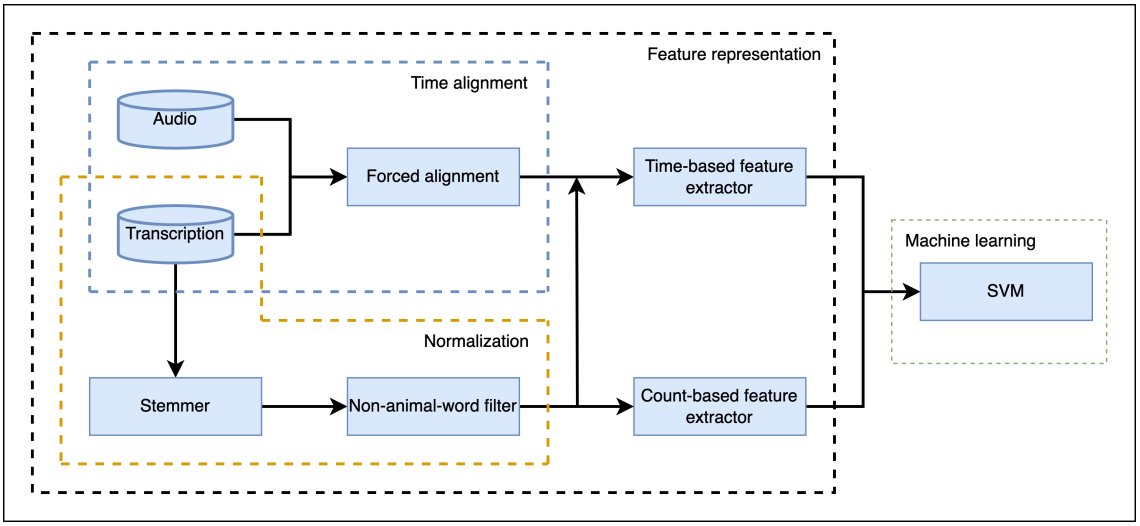

Figure 3.2: Diagram of the computational framework including to distinguish participants with MCI from those with NC based on audio recording and transcription of their responses to an AF test. The first module of this plot, *Feature Representation module* (shown by the black box), represents the characteristics of the response using *Time-based* and *Count-based* features. The second module, *Machine Learning module* (shown by the green box), predicts the participant's cognition status (MCI or NC).

timestamp of each word [78, 92, 63], while others recorded the timestamps of keystrokes through asking participants to type their answers and estimated the retrieval time based on certain assumptions, such as they define that the pressed letter right after pressing the "enter" key would be the first letter of the newly retrieved word and consider the corresponding timestamp to be the animal name's timestamp [59, 103, 156, 120, 98]. The downside of those methods is that they either require labor or inevitably constrain the potential participants to the good-at-typing populations. Compared to typing answers, recording answers does not have any constraints on participants' skills and is suitable for all ages.

Forced alignment [99], which is an automatic process that aligns spoken audio with the corresponding words/phones sequence, is a popular tool in phonetic studies, such as automatic measurement of vowel formants [81], voice onset time [142], and speech variation [19]. Since forced alignment has been used in phonetic research for decades, we consider forced alignment to be a mature technique and use it to obtain retrieval time. We use the forced alignment function provided by the Kaldi ASR toolkit [119] to

extract time information from a verbal response. We do not use pre-trained models from Kaldi due to the domain mismatch between these models and our recordings. We follow Kaldi's training process which involves forced alignment to align our recordings and their manual transcriptions.[2] We train an acoustic model, which only needs the transcription and audio as input. The training procedure finds the acoustic model that best aligns the entire transcription of the data with all of the audio files. This results in an acoustic model as well as the forced alignment between audio and transcription.

3.4.2 Normalization

Ideally, a given response would consist of only animal names, but in reality, the recordings often contain extraneous speech, including fillers and conversational words as well as interruptions from the examiner. Sometimes, a participant will ask how much time remains for the test, and other times the examiner offers words of encouragement. In more than half of the recordings, both the examiner and the participant engaged in a bit of casual conversational speech, and the duration and frequency of such interruptions varied across the recordings. Encouragement from the examiner is usually a few short words (e.g., "good"), while a response to a time check might be a whole sentence (e.g., "you still have ten seconds left"). Examiners never say any animal names. Prior to feature extraction, we remove all the non-animal words from the response. From a computational point of view, trimming out these extraneous words results in a shortened response with altered timestamps that will ultimately influence the uniformity of time-based features across different subjects. To compensate for this issue, we first rescale the timestamps to the length of "shortened" audio and then measure the SD and ICRT features. In our working example shown in Figure 3.1, the original 58-second long recording was "shortened" to 50 seconds after filtering out non-animal words. Then, the *rescaled timestamps*, which are computed by dividing the timestamps by 50, rather than 58, are used for computing SD and ICRT features. Figure 3.1 shows both SD and ICRT features before and after time rescaling. This is the major difference between our method and the method deployed in Hills et

[2]We use code from github.com/kaldi-asr

al. [59] who adopt the original time differences.

Another common issue in the AF test is the presentation of plural forms of animal names (e.g., "pigs" rather than "pig"). In order to normalize all animal names in transcriptions to their singular forms, To measure the category similarity, we apply a word stemmer algorithm on responses to remove morphological affixes from words, leaving only the word stem. Also note that many animal names follow irregular pluralization rules (i.e., "hippopotami" and "oxen."). Additionally, some animal names contain more than one word (i.e., "great white shark" and "mountain lion."). To tackle this problem and automatically identify animal names, we built a tool to retrieve the normalized form from the internet. First, we found the singular form of each noun from the Merriam-Webster website, then we referenced Wikipedia to decide whether a multi-word sequence is an animal name. Conveniently, the Wikipedia template for an article about a species includes a biological taxonomy table, so upon looking up a given species, we could confirm it belongs to the kingdom Animalia and ensure that the animal name is valid.

3.4.3 Machine Learning

The machine learning module takes count- and time-based features as input and output assessments.

We first scale the range of computed features into a constrained range using RobustScaler.[3] This is a necessary step in our computational framework as we noticed that the range of derived features greatly differs from each other. For example, the number of correct animal words ranges from 4 to 30 within the responses while the mean count of switches ranges from 2 to 9. Features with large scale will dramatically impact what the machine learning algorithm will learn and prevent the algorithm from gaining benefits from features with smaller scale. RobustScaler centers and scales the data according to the following equation, operating separately on each dimension of the global feature vector:

$$f(x_i) = \frac{x_i - Q_1(\boldsymbol{x})}{Q_3(\boldsymbol{x}) - Q_1(\boldsymbol{x})} \tag{3.4}$$

[3]We use the RobustScaler provided by python package Scikit-learn.

where x_i denotes to the i^{th} feature; $Q_1(\boldsymbol{x})$ and $Q_3(\boldsymbol{x})$ are the feature's 25^{th} and 75^{th} quantiles, respectively.

We choose support vector machine (SVM) [141] to classify participants as either MCI or NC. The SVM uses the features selected, as described in the previous paragraph. An SVM classifier is a discriminative model that attempts to distinguish between two classes of data points separated by a hyperplane in a high dimensional space. The parameters of the hyperplane are learned from a set of training examples. We trained linear and non-linear SVM classifiers employed from the open-source Scikit-learn toolkit [118]. All experimental results, presented in the later sections, are based on the linear SVM as it outperformed the non-linear SVM. We also repeated the experiment using a "Chance" classifier, which randomly assigned participants to MCI and NC classes.

3.5 Experiment

We will separately evaluate both count- and time-based features derived from the Troyes and ESA-based methods. The difference between these methods lies in the semantic representation of animal names and how names are clustered, which was introduced in Section 3.2.

In this section, we present the experiment setups. We present data, performance criteria and the chosen cross-validation method. In Section 3.5.1, we present the demographic information of the dataset. In Section 3.5.2, we present the evaluation metrics. In Section 3.5.3, we present the chosen cross-validation method for resolving the data imbalance problem.

3.5.1 Data

This study's participants come from community cohort studies of brain aging at the Layton Aging & Alzheimer's Disease Center, an NIA-funded Alzheimer's center for research at Oregon Health & Science University (OHSU). All participants' cognitive status are rated by the Clinical Dementia Rating (CDR) [107] score. If a participant gets a score of 0, it means the person's cognitive status is normal (NC). A score of 1 indicates dementia of

Alzheimer's disease and a score of 0.5 indicates mild cognitive impairment (MCI), which is considered to be a transitional stage between normal aging and Alzheimer's disease.

We have animal fluency test recordings from 98 subjects and corresponding manual transcriptions. Out of 98 participants, 28 were diagnosed with MCI and the remaining 70 participants were NC. Our statistical analysis using the Student's t-test showed a significant difference in participants' demographic factors between MCI and NC groups, namely age, sex, and education level. It is possible that observed changes in spoken language patterns of participants with MCI are the consequence of subject differences in demographic factors such as age or education level regardless of cognitive decline [97]. In order to control for demographic factors, we used a freely available package, ldamatch,[4] which selects the largest subset of the NC group that is statistically matched to the MCI group [80]. This resulted in a group of 70 participants: 28 with MCI and 42 with NC. Table 3.2 reports the basic characteristics of the selected participants. We report the mean and standard deviation for educational level, age, and sex. Using Kolmogorov-Smirnov test, we show that they are statistically insignificant. Additionally, Mini-Mental State Examination (MMSE) [153] and AF score, which should be significantly different between the two cognitive groups, are also presented in this table.

Variable	NC n=42	MCI n=28	p-value
Age	89.9 (5.55)	91.2 (5.17)	0.32
Gender (% Women)	64.3%	50%	0.85
Years of Education	14.3 (2.70)	14.8 (2.79)	0.53
MMSE	28.0 (1.63)	26.0 (3.16)	0.08
AF score	17.3 (4.99)	13.3 (4.12)	0.04

Table 3.2: Baseline characteristics of MCI and NC. The Kolmogorov–Smirnov test was used to calculate p-values

3.5.2 Performance Criteria

To evaluate the performance of the proposed classifier, we adopted the following evaluation metrics:

[4]The Idamatch package can be found at CRAN.R-project.org/package=ldamatch

Sensitivity: the portion of correctly identified MCI participants (true positives). Sensitivity assesses the capability of the model to distinguish MCI from NC participants.

Specificity: the portion of correctly identified NC participants (true negative). Specificity measures how well the model avoids false positives.

Area under the curve of receiver operating characteristics (AUC ROC): The most common method for evaluating the performance of a binary classifier is the ROC [56], which plots the *sensitivity* (true positive rate) of the classifier versus *specificity* (false positive rate) of the classifier as the classification threshold varies. We use a classification threshold in a grid search schema to cover the most positive threshold (everything true) to the most negative threshold (everything false).

3.5.3 Cross-validation on the Imbalanced Dataset

To demonstrate whether our statistical analyses and experimental results were independent of our data sets, we use cross-validation (CV) techniques in which the train and test sets are rotated over the entire dataset. In an imbalanced dataset, a machine learning algorithm receives more information from the class that has more samples and consequently may not learn properly from the smaller-sized class. To overcome this problem in our imbalanced dataset, we use a special leave-one-pair-out (LOPO) cross-validation scheme. LOPO cross-validation randomly selects one sample from each class to form a testing set and, from the rest, creates a balanced training set by randomly selecting the same number of samples from each class. In our example, with 28 MCI and 42 NC samples, leaving one pair for the test, LOPO randomly selects 27 NC out of the remaining 41 NC samples at each training iteration. To reduce the effect of randomization in our final results, LOPO shuffles the data and repeats the above procedure 500 times Lastly, LOPO averages across 500 scores and reports that as the performance of our model.

3.6 Results

In this section, we present the results of our experiments. In Section 3.6.1, we present statistical measurements of selected features. In Section 3.6.2, we compare the classification

Feature Name	p-value
AF score	0.04
ANWC	0.21
NS	0.39
SCR	0.91
mean OSR	0.03
median OSR	0.02

Table 3.3: Kolmogorov–Smirnov test results of features that use ESA for semantic representation.

results between using only count-based features and using both count- and time-based features. In Section 3.6.3, we explore the impact of the threshold used in the ESA-based approach.

3.6.1 Statistical Analysis

In this section, we extracted our proposed time and count-based features from *switch* and *cluster* components identified by the ESA-based method. To explore the effectiveness of these features in differentiating subjects with MCI from NC, we conducted a statistical analysis on each of our selected features using the Kolmogorov-Smirnov test. We present their p-values in Table 3.3 and consider a feature to be significantly different between the two groups if the p-value is lower than 5%. The table shows the AF score is significantly different between MCI and NC. However, the widely studied count-based features, ANWC and NS, are insignificant differences between MCI and NC groups. Even though SCR, a count-based feature proposed by us, is not significantly different between MCI and NC, SCR is one of the features selected by the RFECV process, as was discussed in Section 3.3.3. On its own, it is not useful, but is probably useful when considered in conjunction with the other selected features Both *mean OSR* and *median OSR* features significantly distinguish the two groups with p-values of $p < 0.03$, $p < 0.2$, respectively.

We present the distributions of these features in Figure 3.3 in order to see the value difference of these of our selected features between MCI and NC. The mean AF score of NC is higher than the MCI's. This is consistent with early research that NC can retrieve more animal names than MCI. For both *mean OSR* and *median OSR*, the average score

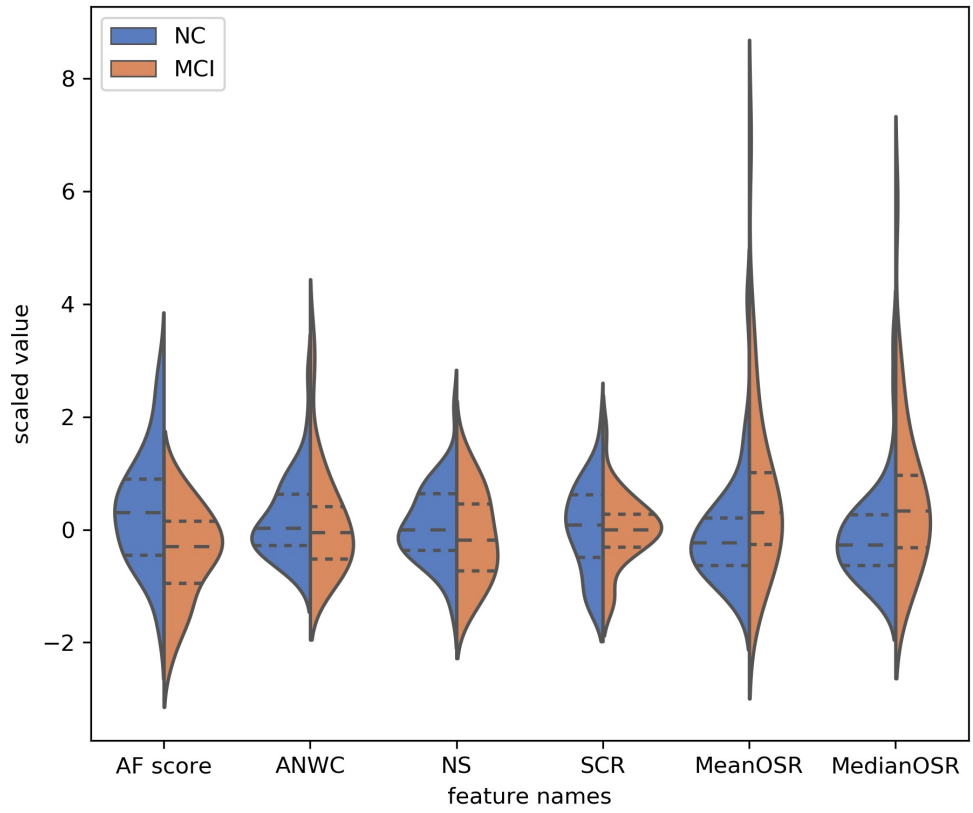

Figure 3.3: Probability distribution of features (y-axis) selected by the feature selection algorithm. The dynamic range of features has been normalized according to the *RobustScaler* approach. The dotted line from left to right are 25% quantile, 50% quantile and 75% quantile.

of NC is lower than the average score of MCI. This indicates that NC participants have less difficulty retrieving a new word upon switching to a new cluster than those with MCI.

3.6.2 Classification Results

In this section, we compare the performance of the SVM classifier between participants with MCI and the NC using different sets of features. We compare the performance difference between only using traditional count-based features and combining both count-

and time-based features.[5] To ensure that our features do not depend on the clustering and switching methods, we make two separate comparisons using features generated from either ESA- or Troyer-based method. To better understand the effectiveness of our features, we also created three baseline models.

Chance: randomly assign participants into MCI and NC classes.

Demographic: an SVM classifier trained with demographic features of subjects, namely age, gender, and years of education. We include this model because these features are widely used in medical research.

AF score: an SVM classifier trained with AF score.

Subjects in MCI and NC are demographically matched, and thus, the performance of the demographic model is close to the chance model.

Table 3.4 shows the performance of the models using LOPO cross-validation in terms of Sensitivity, Specificity, and AUC ROC. In terms of ROC AUC, count models outperform the three baseline models. The addition of time-based features — whether extracted based on Troyer or ESA methods — further improves ROC AUC. This indicates that our time-based features contain information that count-based features do not have, and are useful to distinguish MCI from NC. Our time-based features work well on both Troyer- and ESA-based methods. The count+time models also outperform the count models on sensitivity and specificity except when using the ESA-based method where there is a trade-off between sensitivity and specificity. As aside, it is worth noting that the demographic model performs better than the chance model even though the demographic features are not significantly different between MCI and NC. This indicates the demographic model's distinguishing power may come from the conjunction of demographic features. The AF score model outperforms the demographic model. This indicates that the AF score's distinguish power is more powerful than the conjunction of demographic features.

Features	Method	ROC AUC	Sensitivity	Specificity
Count	Troyer-based	70.25%	62.71%	66.48%
	ESA-based	73.81%	75.46%	60.46%
Count + Time	Troyer-based	77.76%	76.02%	67.11%
	ESA-based	77.09%	70.27%	69.68%
Chance		50.00%	49.56%	49.99%
Demographic		59.30%	49.25%	59.64%
AF score		65.63%	67.30%	63.04%

Table 3.4: Classification results using selected features (mean over 500 leave-pair-out spatial cross-validation repeats.

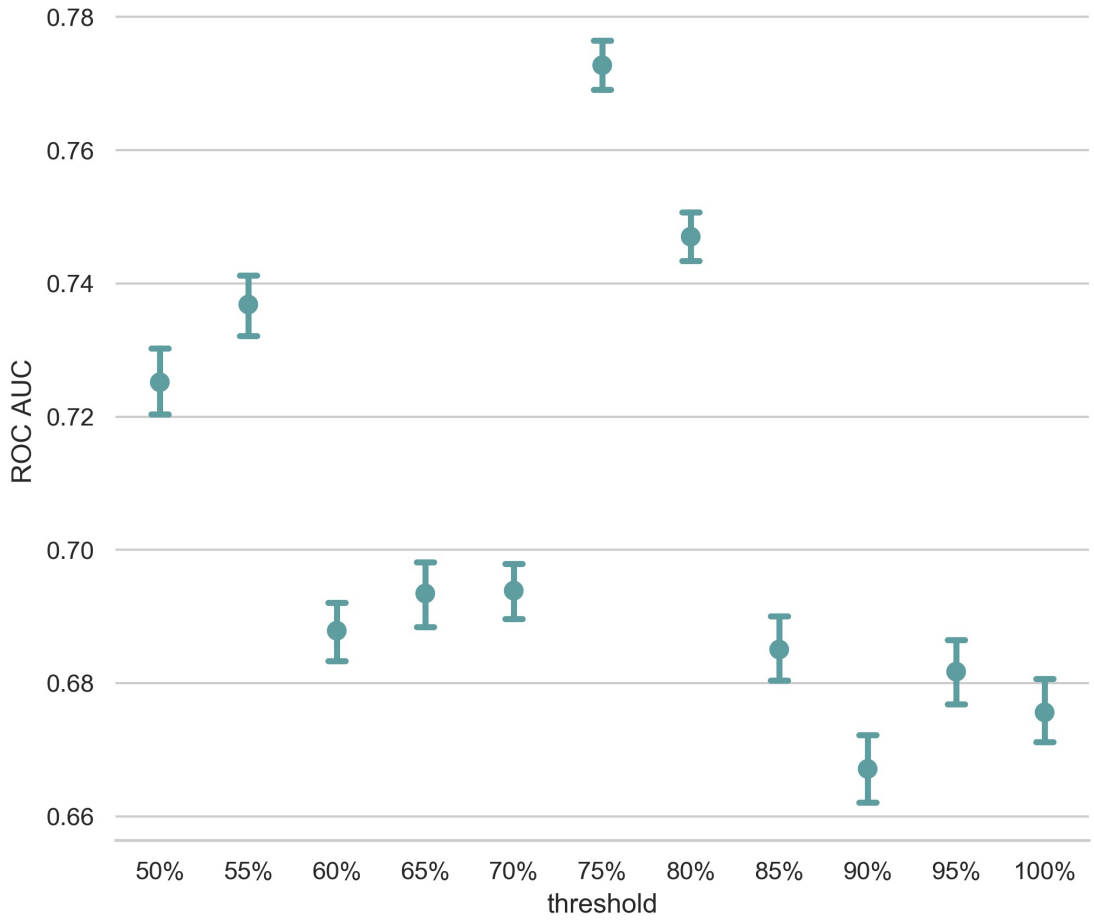

Figure 3.4: The x-axis is the different threshold setting ($xx\%$ of mean cosine similarity of an individual's answer). The y-axis is the ROC AUC score.

3.6.3 Impact of the Semantic Threshold

In the previous section, we have shown the effectiveness of count- and time-based features extracted based on Troyer- and ESA-based methods. While the Troyer-based method has a static rule for determining switches (i.e., two adjacent animal names do not share any common categories in a structured table), the ESA-based method determines switches based on a dynamic threshold that is defined as 75% mean semantic similarity across all possible animal name pairs in a given answer. Since the discriminative power of count-based and time-based features highly depends on the identification of switch and cluster components in the response, one question is how the value of the threshold setting influences the classification result. In order to gauge the influence of this factor, we incrementally increased the threshold from 50% to 100% with a step size of 5% and created 11 feature sets. We again use LOPO cross-validation to evaluate the feature sets and measured the AUC ROC of the classification task on the pairs of test subjects. Figure 3.4 presents the average AUC ROC at each threshold value. As shown in this figure, the classification model reaches its highest AUC ROC at a threshold of 75% which is the same threshold defined by Woods et al. [161]. This suggests that the threshold could be used for clustering and switching across various datasets of the animal fluency test. This plot also verifies the importance of the threshold value to the performance of our classification model.

3.7 Discussion

We use features extracted from the animal fluency test to distinguish MCI from demographically matched NC participants. Using count-based fluency scoring resulted in similar diagnostic category discrimination as reported by others using conventional counting [113]. Early research has used ASR techniques to examine the VF test. Pakhomov et al. [116] used Kaldi ASR toolkit [119] to transcribe responses and Konig et al. [75] used Google's automatic speech recognition service for the same purpose. These studies either attempt to predict the raw VF score based on automatically generated response [116] or only

[5]While technically SCR is a count-based feature, we put it into the time-based group as it is designed to be a compensation for OSR.

investigate count-based measures beside the raw VF score for differentiating MCI from NC participants [75]. In contrast, the crux of our work that differentiates it from these studies is how we intend to employ the ASR system not only for automatic transcription but to perform the "forced alignment" algorithm for quantifying the temporal properties of verbal responses leading to the extraction of time-based measures. As shown by our experimental results, our time-based measures significantly improved the accuracy of our classification model.

In considering the goal of automating the administration and scoring of this test, we developed a method to go beyond conventional scoring that relies on the number of correctly produced category items. This unsupervised approach ultimately will require an algorithm that can be objectively applied by employing machine learning to discern not only the simple counts but also other aspects that may add to the discriminatory power of the VF test. We experimentally showed that the conventional test score (i.e., the number of correctly recalled animal names within a minute) cannot capture other clinically useful information from the test and once it is solely used for training an SVM classifier, the resulting model achieved a poor performance. To mitigate this shortcoming, we proposed a computational approach for automatically analyzing the verbal responses via a set of time-based features that characterize the semantic search strategy during the word retrieval process. We statistically showed that these proposed features can differentiate individuals with MCI from NC controls. Additionally, they positively contributed toward the performance of an SVM classification model once they were added to standard count-based features. In spite of promising results achieved through the proposed computational model, considerable work remains to improve the accuracy of the classification algorithms. Our analysis relied on animal names that were included in the table of Troyer et al. [151]. However, there are always animal names that are unknown to this table and current analyses treat them as non-animal names and that impacts our assessment. A valuable avenue for future research would be to explore the feasibility of NLP techniques to address this limitation using more sophisticated methods of word representation that is not limited to a word table. Addressing these limitations in future work is expected to result in viable speech-based outcome measures, derived from the verbal fluency test, for

individuals with a range of neurodevelopmental disorders including MCI and Alzheimer's disease. The proposed methodology can increase the capacity for screening/detection of MCI by employing measures that cannot be easily computed manually in real-time.

Chapter 4

Refining Automatic Speech Recognition System for Older Adults

In the previous chapter, we presented our work on developing features to improve the accuracy of mild cognitive assessment. In this chapter, we present our work on developing a transfer learning method that uses limited data (i.e., 10 hours) to adapt a pre-trained ASR model for socially isolated older adults (75+ years old) with possible cognitive impairments.[1] We show that adapting the ASR with older adults' recording data through transfer learning can improve the ASR's performance on older speech. We propose a transfer learning method that can increase the fine-tuning efficiency with limited training data and can achieve better performance than the standard transfer learning method.

4.1 Introduction

ASR systems have been widely used in medical applications such as clinical documentation [170] and healthcare systems [65]. ASR systems have also been explored in medical research, such as automating the diagnosis of cognitive tests for Alzheimer's disease [77]. More recently, analyzing everyday conversation has received increasing attention as it opens a window toward individuals' personal world and could potentially reveal their cognitive and behavioral states [68]. This analysis relies on high-fidelity transcription, which is labor-intensive and costly. A high-quality ASR system could potentially serve as an

[1] L. Chen and M. Asgari, "Refining Automatic Speech Recognition System for Older Adults," ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 2021, pp. 7003-7007, doi: 10.1109/ICASSP39728.2021.9414207. [25]

alternative solution to facilitate the analysis process.

Nowadays, deploying DNN in key components of the ASR engine, namely the acoustic and language models, has significantly boosted performance. On the other hand, ASR systems inherit DNN's hunger for target-domain data [143] and the susceptibility to domain mismatch. Although training a domain-specific model with sufficient data is ideal, collecting training data is a challenging task in the medical field especially when facing a narrow target population [61].

This data collection challenge can be caused by multiple factors, such as small population size, financial limitations of the project and/or privacy concerns. Publicly available ASR systems, which are trained on large amounts of data from adults, perform well on their training domain (i.e., adults' audio recordings). However, their performance degrades in clinical applications for older adults as the acoustic characteristics of the target speakers deviate from those used in training examples. This is a critical limitation of using open-sourced ASR to transcribe the speech of older people (i.e., above 75 years old) since the strong acoustic mismatch leads to inaccurate recognition. Older adults' vocal characteristics are different from adults. Age-related speech deterioration begins around 60 years old [104] resulting in significantly different voice characteristics in comparison to the younger generation [88]. Moreover, the plausible influence of impaired cognitive functioning on acoustic features of MCI subjects [126] might serve as an additional source of acoustical mismatch. This means an ASR for older adults most likely faces more acoustic variation. To model this variation, DNN requires more data.

Our purpose in this chapter is to improve an ASR's performance on older speakers when our available data is limited. Our training dataset contains 12 hours of transcribed recordings collected from above 75-year-old older adults with possible cognitive impairments. To train an end-to-end ASR system, we propose using transfer learning (TL) to address the data limitation. Early work in transfer learning focused on utilizing a DNN's hidden output, in contrast, we explore utilizing not only the hidden output but also the intermediate outputs. We design a conditional-independent attention mechanism to leverage the intermediate outputs from a pre-trained model. Our method performs better than the standard fine-tuning method.

The remainder of this chapter is organized as follows. Section 4.2 presents background. Section 4.3 describes the conditional-independent attention mechanism and its variation. Section 4.4 describes our speech data. Section 4.5 describes the experimental setup and our results.

4.2 Background

This chapter makes use of DeepSpeech2, transfer learning and the attention mechanism, which we already discussed in Section 2.9.1, Section 2.7, and Section 2.3.3, respectively. In the rest of this section, we present a brief overview of deepspeech2, including its architecture and its decoding process.

Deepspeech2 [7] is an end-to-end ASR system that leverages CNN and bi-GRU for modeling the spectrogram to the sequence of 26 English letters. The left graph of Figure 4.1 is the architecture that we use in our experiments. This architecture utilizes convolution (Conv) blocks for low-level information processing and bi-GRU blocks for high-level processing. The output block has a linear layer. In the decoding process, Deepspeech2 [7] utilizes a beam search approach [160] to search for the transcription with the highest probability based on the combination of probabilities from the DeepSpeech2 acoustic model and a n-gram language model.

4.3 Conditional-independent Attention Mechanism

While early work in transfer learning focused on utilizing the hidden outputs of a DNN, we hypothesize that intermediate outputs contain useful information related to target domains. We propose a method to utilize these intermediate outputs for adapting a pre-trained model through transfer learning. We also extend the use case of the attention mechanism. Instead of long-term memory management, we utilize the mechanism to summarize information from hidden/intermediate outputs for the target domain. In Figure 4.2, we compare the difference between the general attention mechanism and our modified version. The left graph shows the general attention mechanism. The right graph shows the backbone of our mechanism. While the general attention mechanism generates

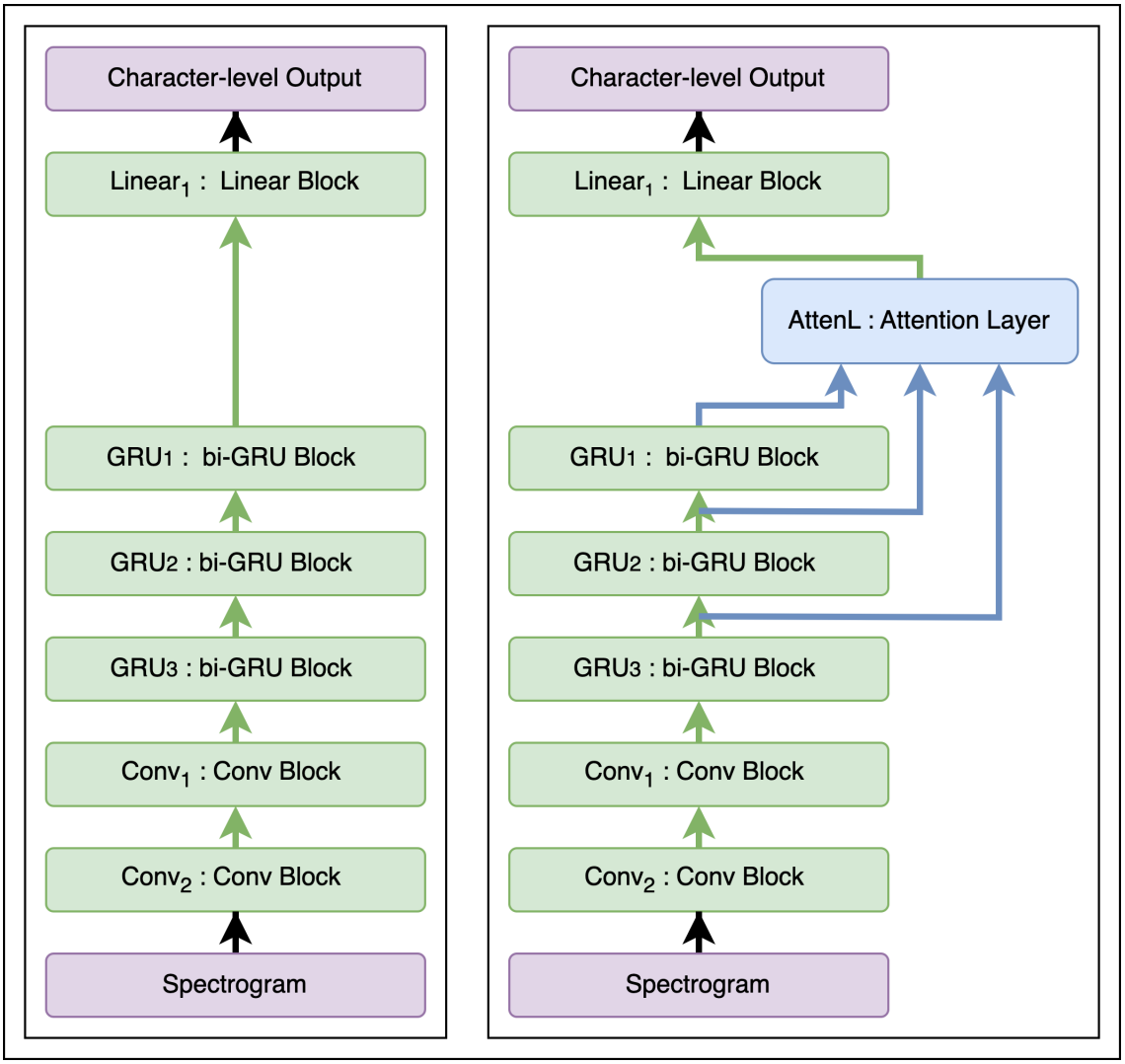

Figure 4.1: The left graph is the architecture of DeepSpeech2 which is our base model. The right graph shows the modified DeepSpeech2 architecture to leverage intermediate outputs. Each box contains the block's nickname and type.

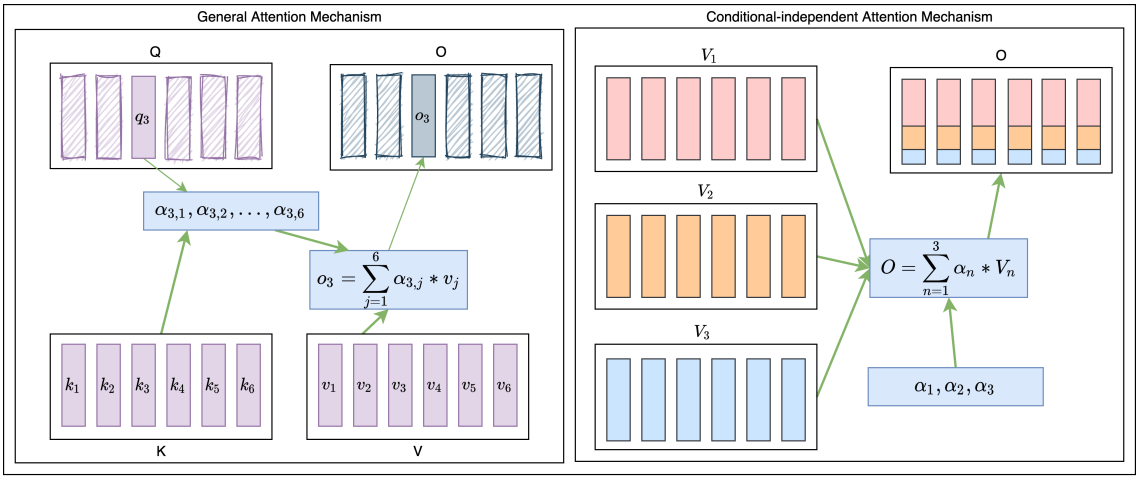

Figure 4.2: The left graph shows the backbone of the general attention mechanism. The right graph is the backbone of our conditional-independent attention mechanism.

output for each query through drawing attention to all values in one sequence (i.e., V), we use the attention mechanism to draw attention to multiple intermediate outputs (i.e., V_1, V_2 and V_3). That is, the general attention mechanism dynamically assigns importance to values in a sequence, which can be any V. Our mechanism assigns fixed importances to multiple Vs where each V represents an intermediate output from a DNN. The right graph in Figure 4.1 is the modified DeepSpeech2 architecture where we add an Attention layer (AttenL) to summarize bi-GRU blocks' outputs. Although all examples in both Figure 4.1 and Figure 4.2 take three inputs, theoretically, the mechanism is not limited to this number and we can use it in any network. But, we focus our research on the data-scarcity problem on a pre-trained model. The number of outputs Vs is contained by the pre-trained model which was the state-of-the-art model. We modified the general attention mechanism into two conditional-independent versions: (1) the manual attention mechanism and (2) the learnable attention mechanism.

4.3.1 Manual Attention Mechanism

In the manual attention mechanism, we remove the dynamic attention-assigning function from the general attention mechanism and manually assign attention to the outputs of GRU blocks based on our knowledge of the model. The following equation is the operations

of manual attention:

$$O = \sum_{n=1}^{3} \alpha_n * V_n$$

where V_n is the output of GRU_n and α_n is the attention that we manually assign to it. This format is the same as a weighted average of GRU outputs. This does not have additional parameters. Therefore, it is convenient for evaluating the effectiveness of utilizing intermediate information and we present results in Section 4.5.1. We name this mechanism as the manual attention mechanism.

4.3.2 Learnable Attention Mechanism

For the learnable attention mechanism, rather than manually assigning the weights, we learn them from a target dataset. To learn the attention from the target dataset, we apply a function to learn how to assign attention in a conditional-independent fashion. The learnable attention mechanism can be described with the following operations:

$$\begin{aligned} R &= \left[\begin{array}{c|c|c} R_1 & R_2 & R_3 \end{array} \right] \\ M &= \left[\begin{array}{c|c|c} M_1 & M_2 & M_3 \end{array} \right] \\ score &= R * M \\ \alpha_* &= softmax(score) \\ O &= \sum_{n=1}^{3} \alpha_n * V_n \end{aligned}$$

where *score* is a one-by-three matrix and V_n is GRU_n's output. The matrix R is a one-by-r non-negative matrix and is the additional learnable parameter matrix. R_1, R_2 and R_3 are partitioned matrices of R. Each represents the importance of the output of the GRU block with the identical subscript ID. R_n is a 1-dimensional vector of learned weights that will be learned. The actual length is hand-set. The overall length of R is r. If R_n contains more elements than others, it is more likely that GRU_n's output receives higher attention than others. Matrix M, which is a r-by-three binary matrix, summarizes each partitioned matrix by summing up its elements. M_n is a r-by-1 partitioned matrix of M and is used to sum the elements in R_n. Thus, M is not learned. This matrix is fixed once we set

the column sizes for all R_n where $n \in \{1, 2, 3\}$. Thus, we view matrix R as the matrix of representatives and matrix M as the matrix of tally. By setting the column size of R's partitioned matrices, we can gently encourage the mechanism to assign extra attention on blocks that are important in our prior knowledge. For example, if we think GRU_1 should receive more attention, we can define the column sizes for R_1, R_2, and R_3 to be 2, 1, and 1, respectively. The R and M should be in the following format:

$$R = \left[\begin{array}{cc|c|c} ele_1 & ele_2 & ele_3 & ele_4 \end{array} \right]$$

$$M = \left[\begin{array}{c|c|c} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right]$$

where ele_h is a scaler for all $h \in \{1, 2, 3, 4\}$. We name this mechanism as the learnable attention mechanism.

4.4 Data

The data for this chapter comes from a long-term behavioral research project that uses internet-based social interactions as a tool to enhance older adults' cognitive reserve. This project, titled as *I-CONECT* [164, 36], was conducted at Oregon Health & Science University (OHSU), University of Michigan, and Wayne State University. Socially isolated older adults above 75 years old were mainly recruited from the local Meals on Wheels program in Portland, Oregon and in Detroit, Michigan (with recruitment of African American subjects). Conversations are semi-structured, in which participants freely talk about a predefined topic (e.g., picnic, summertime, swimming and so on) with a moderator through online chatting. The corpus includes 30-minute recordings of 61 older adults (29 diagnosed with MCI, 25 with normal cognition, and 7 without clinical diagnosis) along with professionally annotated transcriptions.

4.4.1 Preprocess

As our target speakers are older adults, we remove moderators' utterances based on the speaker labels of utterances in the manual transcription. For each utterance of the older adults, we extract word-level timestamps using a force-alignment algorithm available in *Gentle*,[2] which is open-source software. We segment long utterances into multiple pieces that are less than 7 seconds long, which is the same process as DeepSpeech2 [7] used, by utilizing the word-level timestamps. Finally, we removed all utterances that are less than 3 seconds. Empirically, we found that the alignment quality on short utterances is not very good.

4.4.2 Data Splitting

For both validation and testing sets, we randomly selected 2 MCI and 2 healthy participants from both genders which left 53 participants for the training set. With 14 hours of transcribed speech, this leaves about 12.7 hours of audio recordings for training. The total recording duration in the validation set and testing set are 0.66 and 0.56 hours, respectively. We use the validation set to select hyperparameters (e.g., learning rate) and the testing set is only used for assessing the model's performance.

4.5 Experiment

Our base model is an open-sourced DeepSpeech2 model,[3] which is trained on Baidu's 8000 hours of internal data, as well as the corresponding n-gram language model. The language model is fixed throughout all experiments. We have two baselines: 1) we test the original model on our testing set. 2) We tune the entire model with our training set for 40 epochs and evaluate its performance. The tuned model is referred to as the standard weight transfer learning model.

[2]https://github.com/lowerquality/gentle
[3]github.com/PaddlePaddle/DeepSpeech

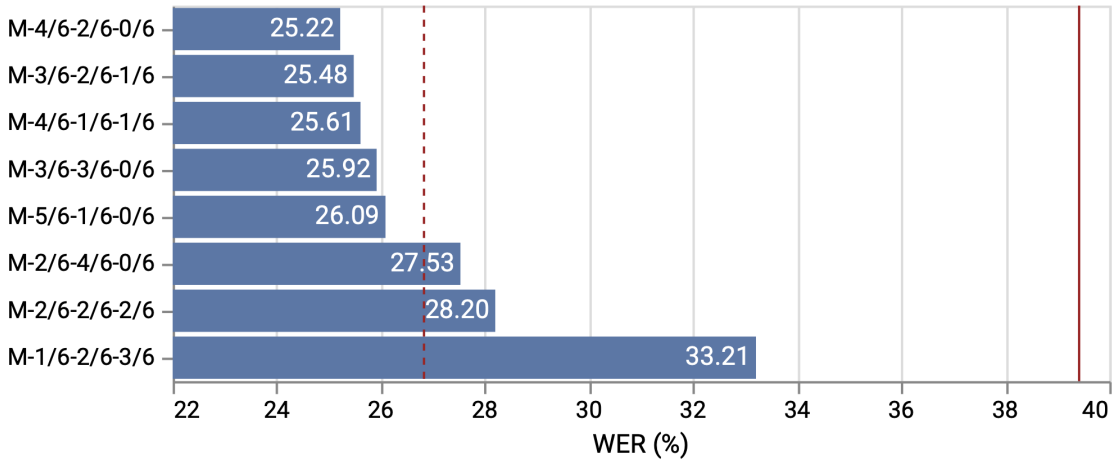

Figure 4.3: Model performance for manual attention settings. The red dotted line is the WER of the standard weight transfer learning model (26.8%). The red solid line is WER of the base model (39.42%).

4.5.1 Manual Attention Layer

We use the manual attention mechanism at the AttenL. We want to keep the number of models at a reasonable level in order to run a brute-force exploration of all settings. Thus, define the basic attention unit to be 1/6 and restrict all αs to be an integral multiple of the unit. We use *M*-α_1-α_2-α_3 to present the setting of attention in an experiment. For example, if we assign all attention to GRU_1, the attention setting is *M-6/6-0/6-0/6*. In the training process, we fine-tune the modified model for 40 epochs. Also, we set the linear block's initial learning rate to be 4e-5 and set other blocks to be 2e-5. Throughout the 40 epochs of training, we reduce the learning rates by 50% after every 15 epochs.

In Figure 4.3, shows the results of the two baselines and the top 8 performing manual attention models. The standard weight transfer learning model (i.e., the red dot line) outperforms the base mode (i.e., the red solid line) on the testing data. The top 5 settings with manual attention all have more than 50% attention assigned to GRU_1's output, and these all outperform the standard weight transfer learning model. The *M*-4/6-2/6-0/6 achieves 1.58% absolute improvement over the standard weight transfer learning model. That is, lowering the error rate from 26.80% to 25.22%. Since we use the pre-trained Linear_1, which is used to receive GRU_1's output only, GRU_1 is naturally strongly related

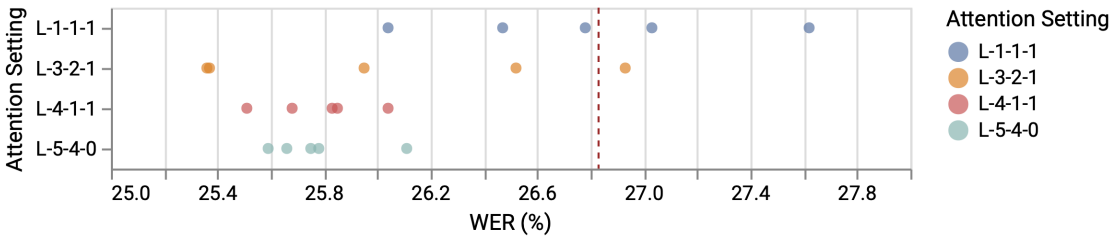

Figure 4.4: Performance on learnable attention settings. The red dotted line is WER of the standard weight transfer learning model.

to Linear_1. On the other hand, unreasonable settings, assigning small attention to the output of GRU_1, perform worse than the standard weight transfer learning model. Overall, we have several models (i.e., *M*-4/6-2/6-0/6, *M*-3/6-2/6-1/6, *M*-4/6-1/6-1/6, *M*-3/6-3/6-0/6, *M*-5/6-1/6-0/6) that utilize outputs from GRU_2 and GRU_3 outperforms only using outputs from GRU_1 (i.e., the red dot line in Figure 4.3). These results bring us confidence in utilizing intermediate outputs.

4.5.2 Learnable Attention Layer

We adopt the learnable attention mechanism at the AttenL to learn the attention from the target dataset. We use *L*-r_1-r_2-r_3 to specify the column sizes of partitioned matrices in R. We evaluate four settings: *L-1-1-1*, *L-4-1-1*, *L-3-2-1* and *L-5-4-0*. We use the first one to evaluate the learnable attention mechanism's learning ability. The other settings are designed to encourage more attention to GRU_1's output. All experiments first train the AttenL for 5 epochs while freezing other blocks. Then, we reverse the freezing status and train the model for 40 epochs. For the training process, we set the linear block's initial learning rate to be 4e-5 and set other non-frozen blocks to be 2e-5. Throughout the training process, we schedule a 50% reduction on learning rates every 15 epochs. We try each setting 5 times to evaluate the influence of random initialization on additional parameters.

In Figure 4.4, shows the results of running each setting 5 times. Random initialization dramatically influences *L-1-1-1*'s final outcome. On the contrary, we achieve more stable performance when applying prior knowledge through the column setting. This proves that

setting the column sizes, based on prior knowledge, positively influences a tuned model's performance. Further evidence comes from the comparison between *L-4-1-1* and *L-3-2-1*. Although the total column sizes of both configurations are the same, the performance of *L-4-1-1* is more stable than the other's performance. Both *L-4-1-1* and *L-5-4-0* outperform the standard weight transfer learning model over all 5 trials. Our results are marginally worse than the optimal WER in Section 4.5.1, but we cannot exclude the negative influence from the small training set. Overall, we show that the learnable attention mechanism with reasonable configuration can stably outperform the baseline (i.e., the red dot line in Figure 4.4). We will analyze how to size of training data impacts the performance in the future. Moreover, the learnable attention mechanism can be transformed into a conditional-dependent form, which is a flexibility that the manual attention mechanism does not have. In the future, we will develop a conditional-dependent form that can dynamically assign attention based on the input's characteristics.

4.6 Conclusion

We propose a conditional-independent attention mechanism to leverage a pre-trained model's intermediate information for model adaptation on the older-adult domain. We experimentally identify the domain mismatch between the pre-trained DeepSpeech2 model and older adults and the benefit of applying transfer learning. Our method, which builds on transfer learning, can further reduce the mismatch. Also, our experiments support that guiding the training direction with prior knowledge reduces the negative influence caused by random initialization.

Chapter 5

An Efficient Architecture for Small Datasets

In the previous chapter, we presented a novel transfer learning method to adapt a pre-trained ASR model for a target population when we only have limited training data (less than 20 hours of recordings). We used an open-source DNN model as the pre-trained (base) model. In this chapter, with the goal of training base models, we focus on developing a DNN architecture that can be trained efficiently with a small training dataset (i.e., about 100 hours of recorded audio). We present our preliminary research using a small academic dataset as a proof of concept.[1] We quantitatively validate each of our modifications and show that our refined DNN is more efficient than the original DNN when the training dataset is small.

5.1 Introduction

Transformer [155] is a popular attention-based DNN architecture in recent years and was presented in Section 2.4.2. Transformer has been used to train large models [23]. A transformer contains multiple blocks with the same structure. Each block contains multiple DNN layers including a multi-head self-attention (MSA) layer. With enough training data, deep transformers outperform their shallower siblings on mainstream tasks [23, 53, 17]. These deep transformers require large datasets so that they can be trained to perform well

[1]This chapter is based on: L. Chen, M. Asgari, and H. H. Dodge, "Optimize wav2vec2s architecture for small training set through analyzing its pre-trained models attention pattern," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7112–7116. [27]

on the general population. But, in medical research, we are facing special target populations, such as non-typical developed children, vocal injured seniors and older adults with MCI. Their vocal characteristics are different from the general population [126, 132, 157]. Moreover, collecting thousands of hours of data from these special target populations is also financially expensive. Hence, there is a need to increase Transformer-based ASR's training performance under a limited data (about 100 hours of recordings) condition so that the medical field gains benefits from ASR research's achievements.

Recent studies on the attention patterns of BERT indicate that we can gain valuable information by analyzing the pattern of what the attention layers are focused on [79]. Inspired by this research, we hypothesize that blocks' general attention patterns are input-independent and the patterns of models, that are trained on large datasets, are close to the optimal pattern. We introduce inductive bias to the Transformer encouraging it to learn these patterns when the training data is limited. We choose Wav2Vec 2.0 [17] as our research target.

Through summarizing block-level attention patterns of a pre-trained Wav2Vec 2.0 model, which has 12 attention blocks, we use two separate techniques to improve its architecture: using local attention to avoid abnormal patterns and parameter sharing. First, we apply local attention to the top 11 blocks and global attention to the bottom block. Second, we let the top 11 blocks share the same set of parameters while leaving the bottom block having its own parameters. We experimentally validated that each modification improves the training efficiency using *Librispeech-100* [117] as the training data. The architecture with both modifications further improves the efficiency. Our modifications are different from prior research. Prior research treats all attention blocks equally. For example, adopt one type of attention mechanism [127] and sharing parameter among all blocks [83]. Unlike them, we consider the bottom block different from the rest based on their block-level patterns. Our experiments show that this difference is the key to the improvement of training efficiency.

In the remaining part of this chapter, we present the background information in Section 5.2. The analysis of the heatmap of the attention mechanism is presented in Section 5.3, where we explain the rationale for using local attention and parameter sharing.

In Section 5.4, we describe the experiment setup, including the dataset, training configuration, decoding process, and evaluation criteria. Finally, we conduct experiments in Section 5.5 to evaluate the proposed methods separately and together.

5.2 Background

In Section 2.3.3, we have described the attention mechanism and presented prior research on categorizing attention patterns. In this section, we provide the background which is relevant to our research. In Section 5.2.1, we introduce an overview of the architecture of Wav2Vec 2.0 and its training process. Our research refines the architecture while employing the same training process. We refer readers to Section 2.9.2 for more details. In Section 5.2.2, we discuss cross-block parameter sharing, which is related to one of our architecture modifications. Lastly, in Section 5.2.3, we introduce model interpretation, which is the core technique used in our research.

5.2.1 Wav2Vec 2.0

Wav2Vec 2.0 was introduced by Baevski et al.[17]. Figure 5.1 shows its architecture in detail. It contains three main components: a convolutional feature encoder, a quantization block, and a Transformer. The feature encoder extracts latent representation from raw audio input. The quantization module discretizes latent representation to a finite set of quantized representations. The Transformer [155], which consists of N global multi-head self-attention blocks, transforms the latent speech representation into content representations.

The training process contains two steps: the pretraining step and the fine-tuning step. In the pretraining step, the model randomly masks some frames of the latent speech representation and the transformer must reconstruct these masked parts. Contrastive loss [17] is used to evaluate the similarity between the reconstructed representation and the matched quantized representation. This objective function encourages the Transformer to reconstruct masked parts accurately. In the fine-tuning step, the pre-trained model is fine-tuned with labeled data and adopts Connectionist Temporal Classification (CTC) [51]

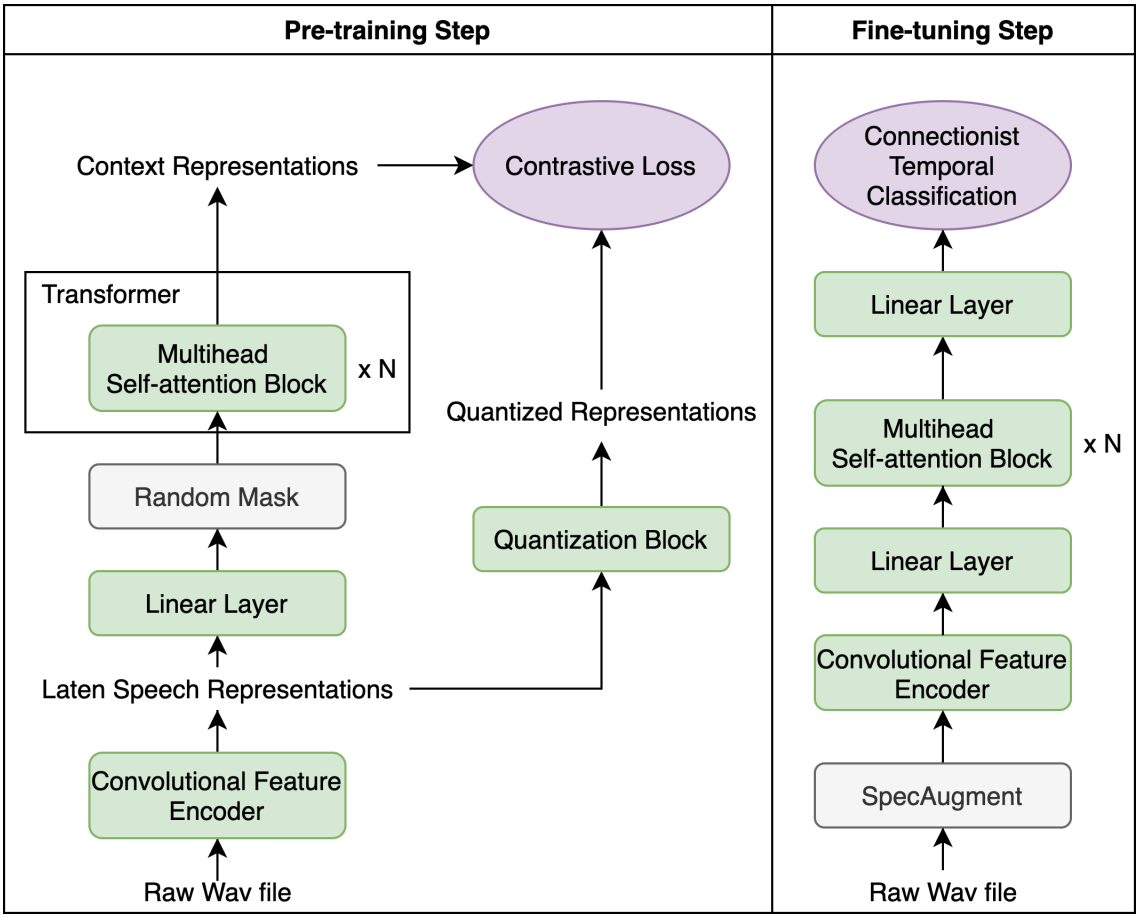

Figure 5.1: It shows the model architecture of Wav2Vec 2.0 and its training process. We use green to indicate there are learnable weights in these subnetworks and adopt gray to mark processing steps. And a purple ellipse represents a loss function.

as the objective function.

5.2.2 Cross-block Parameter Sharing

An intuitive interpretation of cross-block parameter sharing is a recurrent neural network (RNN) considering layer depth as its timestep [34]. This type of network requires a method to decide the network depth. Lan et al. [83] set a fixed network depth. Dehghani et al. [34] leveraged adaptive computation time (ACT) [50] to dynamically decide the depth for each sample and demonstrated its effectiveness on text understanding and generation. Bai et al. [18] proposed a method named Broyden iterations for the same purpose. A common

characteristic of these techniques is that all attention blocks in DNNs share the same parameter. We hypothesize that we can find a better sharing strategy through analyzing the attention pattern.

5.2.3 Model Interpretation

Model interpretation allows researchers to analyze why a model makes certain decisions [106]. Through utilizing interpretation techniques, researchers can argue whether a model extracts valid information that is hidden in the training data instead of biases. Model interpretation techniques can be categorized into two categories: *post-hoc* techniques and *intrinsic* techniques, depending on the way of obtaining the result [106]. The post-hoc techniques attempt to find baseline points of a model and evaluate an input's attribution by measuring the cost of moving it to the baseline point, such as DeepLift [138], Layer-wise relevance propagation [15], Deconvolutional networks [165], Guided back-propagation [5] and Integrated gradients [144]. With Integrated gradients, Mudrakarta et al. [108] capture erroneous logic issues where related samples are inadequate in the validation set. The limitation of post-hoc techniques, however, is that they assume a model is a black box. They cannot provide clues related to a specific layer or layer group.

The intrinsic techniques, on the other hand, build DNN architectures composed of self-explanatory layers. This allows researchers to visualize and analyze the DNN based on layers' outputs. For a self-explanatory layer, researchers have used capsule networks [129], recurrent networks [60], attention networks [52], and so on. The visualization methods depend on the type of layer being analyzed. For long short term memory (LSTM) [60], researchers visualize the gate status and the magnitude of its candidate states [67].

For attention networks, researchers use heatmaps from sample inputs [33, 38] to visualize attention. Heatmaps are utilized for two main purposes: first, to verify whether a model can accurately capture certain properties that can be analyzed through the use of such heatmaps, and second, to explore the model's capacity when attempting to gain deeper knowledge. The first purpose is commonly adopted in text-to-speech. Researchers constructed heatmaps to demonstrate that their models monotonically align the input

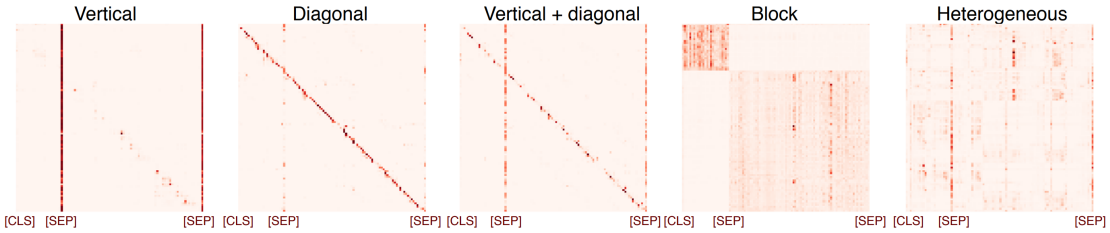

Figure 5.2: The five patterns found by Kovaleva, Olga, et al. [79]. Figure taken from their paper.

phonemes with output audio representations [16, 136, 87] where monotonicity is a commonly agreed innate characteristic of phoneme-to-audio alignment. For the second purpose, Kovaleva, Olga, et al. [79] analyzed pre-trained NLP models' capacity of capturing linguistic information through summarizing patterns of heatmaps. Through a manual visual inspection of around 400 attention maps for both basic pre-trained and fine-tuned BERT models, Kovaleva, Olga, et al. [79] categorizes heatmaps that were collected from BERT [35] into five categories: **vertical**, **diagnoal**, **vertical+diagnoal**, **heterogeneous** and **block** as shown in Figure 5.2. In this chapter, we ignore **block** since this is only observed in tasks/datasets where two distinct sentences form an input (such as, the Recognizing Textual Entailment datasets or Microsoft Research Paraphrase Corpus). Below, we present the other four categories and Kovaleva's explanation of them.

- Vertical: mainly corresponds to strong attention on special symbols, such as sentence delimiters. In Figure 5.2, the symbol, [SEP], is the sentence delimiter. In general, it separates two sentences, such as two alongside sentences, a question and a corresponding answer, or question pairs.

- Diagonal: formed by the attention to the previous/following tokens. In other words, the attention of a query token focuses on that query's neighbor tokens (local context).

- Vertical+Diagonal: a combination pattern of Vertical and Diagonal.

- Heterogeneous: representing all other patterns.

In the next Section, we adopt the categorized patterns to analyze a pretrained ASR model.

5.3 Attention Visualization and Analysis

In this section, we analyze attention heatmaps of multiple validation samples from a pre-trained model to determine how we should improve the architecture. We use a pre-trained Wav2Vec 2.0 model[2] from Fairseq [114]. This model has 12 transformer blocks and is trained on *Librispeech-960* which has 960 hours of training data. We leverage the visualization of mean attention to analyze the attention patterns. We manually inspected all recordings in Librispeech's validation set (*dev-clean*) and noticed that heatmaps across inputs from the same block share similar characteristics. This observation inspired us to categorize the block-level patterns as opposed to Kovaleva, et al. [79] who focused on input-level patterns. In other words, we aim to identify the shared characteristics of attention heatmaps produced by an attention block across all inputs. In contrast, Kovaleva focused on summarizing the relationship between different types of inputs and different types of attention heatmaps. In Figure 5.3, we show heatmaps of the 12 blocks of six recordings from *dev-clean*. As can be seen, heatmaps in each column are similar.

We refer to the block patterns with the corresponding block ids and categorize these 12 patterns into three categories:

- Heterogeneous pattern: *Block 1.*
- Diagonal pattern: *Block 2, 3, 4, 5 and 12.*
- Vertical+Diagonal: *Block 6, 7, 8, 9, 10 and 11.*

We categorize *Block 1* as the heterogeneous pattern. While we can see a clear diagonal line in all samples' *Block 1*, we also observe an almost uniform attention over the whole sequence. A similar observation on low-level transformer blocks has also been reported in Dai, Zihang, et al. [33]. Thus, this pattern is not unexpected. We categorize *Block 2,3,4,5 and 12* as diagonal patterns. A speech frame is strongly related to its neighbors. Therefore, the diagonal patterns are also not unexpected. We categorize the remaining blocks as vertical+diagonal. The categorization is straightforward because, in addition to

[2]Wav2Vec 2.0 Base resulting from the pre-training step from Fairseq [114].

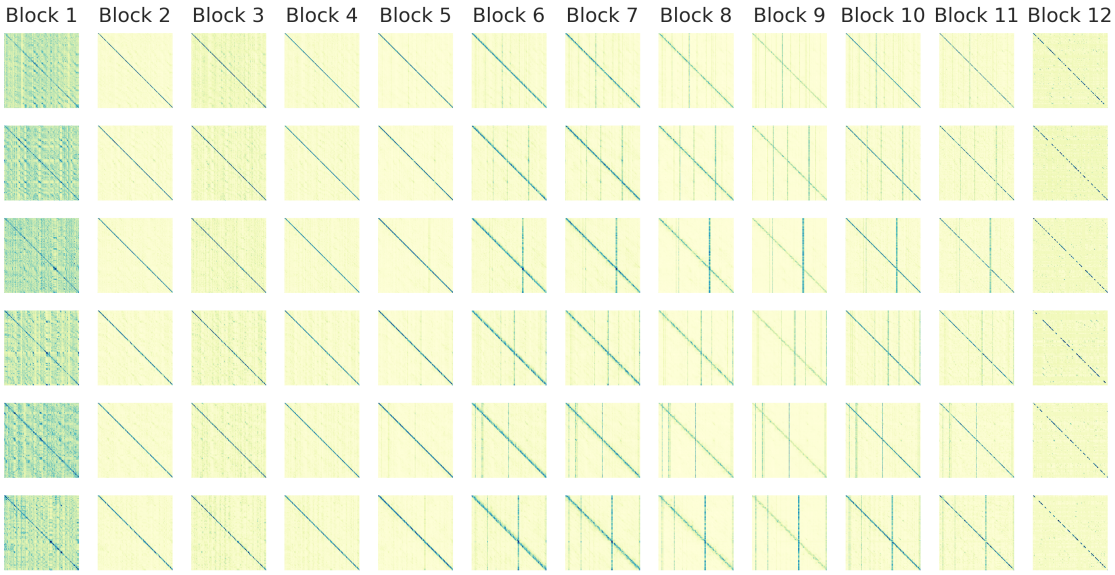

Figure 5.3: The heatmaps of 6 randomly selected audio inputs. A heatmap's x-axis is K's timestep and y-axis is Q's. *Block 1* is the bottom MSAB in Figure 5.1 and *Block 12* is the top one. While the duration of these recordings are different, we present all heatmaps with the same figure size in order to show the similarity of attention patterns.

diagonal lines, there are also a lot of vertical lines in the heatmaps of dev-clean recordings. In the following sections, we will discuss the unexpected vertical pattern and our hypothesis that sharing parameters across blocks in the same category can improve performance.

5.3.1 Using Local Attention to Avoid Abnormal Pattern

We did not expect that a *vertical pattern* would appear in the Wav2Vec 2.0 model. According to Kovaleva et al. [79], vertical patterns strongly correlate to special tokens, such as delimiters, which are designed to modify input sentences in exchange for not having to make minor architecture modifications across tasks after the pre-training step [123] for NLP tasks. Wav2Vec 2.0, which operates on audio instead of sentences, does not modify input with any special symbols, which is why we were not expecting the vertical pattern. From the perspective of matrix operations, this pattern indicates the magnitude of vectors in K are large (where K is from the attention mechanism discussed in Section 2.3.3). Since both W^Q and W^K take the same input, W^K is overly sensitive to these input vectors and it could be a sign of overfitting. Thus, we think vertical patterns are abnormal and should

be avoided.

To avoid vertical patterns, we explore the use of the local multi-head self-attention block to constrain the attention to the local region. In other words, we force the attention pattern to always be diagonal. The major difference between global multi-head self-attention blocks and local multi-head self-attention blocks is that local multi-head self-attention block constrains attention to the region close to the position of a query with an additional parameter named window size. Suppose we set the window size to be 31 which is much smaller than the sequence length. The attention generated for the 20th query will focus on $[v_5, \ldots, v_{20}, \ldots, v_{35}]$ where each element v_t is the tth vector in matrix V which was described in Section 2.3.3. By having attention concentrated on the local region for every query, we will observe high attention along the diagonal line on the corresponding heatmap. We assume this constraint will benefit training when the training set is small (about 100 hours) because this prevents the attention layers from learning vertical patterns.

To validate this choice, we compare two architectures: the original architecture and replacing global multi-head self-attention blocks for block 6 through 11 with local multi-head self-attention blocks. Second, since the attention patterns of blocks 2 through 12 are all related to the diagonal-related patterns (i.e., **Diagonal** and **Vertical+Diagonal**), we constrain all of them to diagonal through replacing these global multi-head self-attention blocks with local multi-head self-attention blocks. This not only avoids vertical patterns but also forces the top 11 blocks' attention pattern to be **Diagonal**. We assume that we should not apply the local multi-head self-attention block to block 1, whose attention pattern is **Heterogeneous**. To validate this, we compare the performance between replacing global multi-head self-attention blocks for block 2 through 12 with local multi-head self-attention blocks and replacing all global multi-head self-attention blocks with local multi-head self-attention blocks. Last, to demonstrate that the key is constraining the attention to a local region instead of constraining to a specific region, we compare three window sizes: 61, 121 and 481. We choose window sizes that are wider than the diagonal blue line in Figure 5.3 and the wider the window size the weaker the constraining power. The corresponding experiments are given in Section 5.5.1.

5.3.2 Parameter Sharing Based on Patterns

The pattern of *Block 1* is different from the rest of the blocks. Its attention over the whole sequence prompts us to question the cross-block parameter sharing strategy discussed in Section 5.2.2. We hypothesize that blocks that belong to the same pattern category can effectively share parameters. Thus, we assume *Block 1* should have its own parameters instead of sharing with other blocks. To evaluate our hypothesis, we fix the network depth to be the same as the depth of the analysis model, which is 12. This is slightly different from what is done in ALBERT [83], as ALBERT shares parameters across all attention blocks.

To evaluate the importance of excluding *Block 1* from sharing parameters with other blocks, we compare sharing parameters across block 2 through 12 with 1) sharing parameters across all blocks, and 2) sharing parameters across block 1 through 11. In order to understand the parameter sharing on its own, we first evaluate parameter sharing strategies through only using global multi-head self-attention block which will be presented Section 5.5.2. Then, we combine both constraining attention and parameter sharing, and evaluate the performance.

5.4 Experiment Setup

In this section, we introduce the experiment dataset and the training process. Moreover, we present the configuration for the decoding process and the evaluation method. Since we are interested in having pre-trained models that are good for building ASR, we adopt the fine-tuning step of Wav2Vec 2.0 to compare these models' WER on the testing dataset instead of their reconstruction power.

5.4.1 Dataset

For training an ASR, we assume 100 hours of transcribed recordings is an achievable data size for collecting data. Thus, we adopt the *train-clean-100* set from Librispeech corpus [117] as training data for both the pre-training step and the fine-tuning step. We evaluate all models on the standard Librispeech dev and test sets: *dev-clean* and *test-clean*.

5.4.2 Training

For both the pre-training and fine-tuning steps, we mainly follow configurations from Fairseq [114], in which the transformer contains 12 global multi-head self-attention blocks. We leverage two Nvidia RTX 3090 GPUs and simulate parallel training on 8 GPUs through setting the update frequency to be 4. We set the maximum token size to be 1.3m per GPU. The equivalent total batch size is 47 audio recordings. In the pre-training step, we train our model for 220k steps. In the fine-tuning step, the total training iteration is 20k steps.

5.4.3 Decoding and evaluation

We leverage beam search with a language model (LM). We adopt a 4-gram language model from Openslr[3] for all acoustic models. The beam size is 1500. We adopt WER to evaluate a model's training efficiency. A model with lower WER means it is more efficient with limited training data.

5.5 Experiments and Results

In Section 5.3, we proposed two modifications: constraining attention to the local region and sharing parameters across attention blocks. We evaluate the first modification in Section 5.5.1. We first demonstrate the benefit of adopting local multi-head self-attention block to the top 11 blocks and next show that the modification is the major contributor instead of the window size of local multi-head self-attention block. Section 5.5.2 evaluates the second modification. Finally, we evaluate combining both modification in Section 5.5.3.

5.5.1 Local Attention

In Section 5.3.1, we argued that the vertical patterns are abnormal. In this section, we show that training effeciency is improved by avoiding the vertical patterns through constraining attention to the local region. We train four models, all with 12 blocks but with different attention blocks.

[3]http://www.openslr.org/11/

G_B1-12: All 12 blocks are global multi-head self-attention block.

L_B1-12: All 12 blocks are local multi-head self-attention block.

L_B6-11: Blocks categorized as vertical+diagonal are local multi-head self-attention block (*Block 6, 7, 8, 9, 10 and 11*), and the rest blocks are global multi-head self-attention blocks.

L_B2-12: Top 11 blocks are local multi-head self-attention block and the bottom block is global multi-head self-attention block.

We set the window size of all local multi-head self-attention block to be 61. We will refer to G_B1-12 as the baseline throughout this section as there are no modifications.

Table 5.1 shows the experiment results. The fact that *L_B1-12* performs worse than the baseline (*G_B1-12*) shows that applying local multi-head self-attention block to all blocks harms the training efficiency. However, *L_B6-11* outperforms the baseline. This supports our claim that the vertical pattern harms the performance. *L_B2-12* outperforms all the other models including *L_B1-12* in Table 5.1. This confirms our assumption that applying local multi-head self-attention block to the top 11 blocks increases the training efficiency.

Window size

We just showed that L_B2-12 outperforms baseline when we set the window size of local multi-head self-attention blocks to 61. In order to demonstrate that the improvement primarily comes from constraining the attention to local region instead of a specific window size, we train the same DNN as L_B2-12 with window sizes of 121 and 481.

Table 5.2 shows that both L_B2-12_W121 and L_B2-12_W481 outperform the baseline in both dev-clean and test-clean. This indicates that adopting local multi-head self-attention block is the causal factor with the improvement instead of a specific window size. Since the window size, 61, performs the best in dev-clean among all three window sizes, we use this configuration in the rest experiments.

Model Name	WER[%]		Param Size
	dev-clean	test-clean	
G_B1-12	7.62	8.33	95M
L_B1-12	7.67	8.45	95M
L_B6-11	7.16	7.90	95M
L_B2-12	6.76	7.49	95M

Table 5.1: Effectiveness of applying local attention to blocks 2 through 12. The model name is formed as [domain attention type]_B[block ID range] where the attention type can only be either local multi-head self-attention block (L) or global multi-head self-attention block (G). The block ID range indicates the blocks that leverage the domain attention type. We always apply global multi-head self-attention block to unspecified blocks.

Model Name	WER[%]		Param Size	Window Size
	dev-clean	test-clean		
L_B2-12	6.76	7.49	95M	61
L_B2-12_W121	6.90	7.58	95M	121
L_B2-12_W481	7.40	7.95	95M	481

Table 5.2: Impact of local attention's window size. We adopt similar naming rule as Table 5.1, except add the window size at the end.

Model Name	WER[%]		Param Size
	dev-clean	test-clean	
G_BS1-12	8.14	8.82	17M
G_BS1-11	7.96	8.82	24M
G_BS2-12	7.43	8.15	24M
L_BS2-12	5.87	6.97	24M

Table 5.3: Effectiveness of parameter sharing and the benefit of modifying the Wav2Vec 2.0 architecture with both local attention and parameter sharing. We adopt a similar naming rule as Table 5.1, except the second part starts as BS which stands for blocks[B] that share[S] parameters. The last column is the model size.

5.5.2 Parameter Sharing

As discussed in Section 5.3.2, we evaluate three parameter sharing configurations using global multi-head self-attention block: 1) all 12 blocks share parameters, 2) top 11 blocks share a set of parameters and *Block 1* has separate parameters, and 3) bottom 11 blocks share parameters and *Block 12* has its own parameters. They are named as G_BS1-12, G_BS2-12 and G_BS1-11, respectively. The parameter size of G_BS1-12 is the smallest while G_BS2-12 and G_BS1-11 are the same.

Table 5.3 shows that G_BS1-12 performs worse than the baseline from Table 5.1, however, G_BS2-12 outperforms the baseline. This suggests that the configuration of parameter sharing is an essential factor. Arbitrarily sharing parameters among all attention blocks may not unleash this technique's full potential. Next, we compare G_BS2-12 and G_BS1-11. The parameter size of both models are the same. But, G_BS2-12 outperforms G_BS1-11. This supports that, instead of parameter size, the sharing configuration is the key.

5.5.3 Combining Both Modifications

In Section 5.5.1 and Section 5.5.2, we evaluated two architecture modifications, which are local attention and parameter sharing, separately, and showed their effectiveness. In this section, we evaluate if these two modifications can be combined together and further improve the performance. We adopt local multi-head self-attention block to the top 11 blocks and these blocks also share the same set of parameters. We apply global multi-head self-attention block to *Block 1*. We refer this model as L_BS2-12.

Table 5.3 shows that L_BS2-12 achieves lower WER than L_B2-12 and G_BS2-12 in both dev-clean and test-clean. This suggests that the modifications are not in conflict with each other. Moreover, since the only difference between L_BS2-12 and G_BS2-12 is the type of attention block, this indicates that focusing attention on the local region is an essential factor in reducing WER.

5.6 Conclusions

Through analyzing the block-level attention pattern, we optimize the transformer's architecture by applying local attention and cross-block parameter sharing on the top 11 blocks. Our optimized architecture is more efficient on a small dataset than the original Wav2Vec 2.0 [17]. We show that sharing parameters among blocks with similar patterns is more effective than arbitrarily sharing parameters among all blocks.

Our experiments also demonstrate that both constraining attention to the local region on the 1st attention block and sharing parameters with other blocks results in higher WER than the baseline (G_B1-12). This indicates that, when developing local attention mechanisms or cross-block parameter sharing techniques based on Wav2Vec 2.0 [17], we have to be cautious about the potential negative effect of modifying the 1st block.

After we completed our analysis and experiments, we also reran the heatmap analysis on a Wav2Vec 2.0 Large model trained on 68000 hours of speech. No attention blocks in this model exhibit a vertical pattern, reinforcing our assessment that the vertical pattern in Wav2Vec 2.0 Base which is trained on 960 hours of speech was overfitting.

Chapter 6

Adapt a Large ASR for Transcribing fillers

In previous chapters, we presented our work on building DNN-based ASR models that perform well on older adults with possible cognitive impairment. In Chapter 4, we presented a novel transfer learning method to adapt a pre-trained DNN-based ASR model for older adults. In Chapter 5, we refined a DNN architecture that can fully utilize a small training dataset for use in creating pre-trained ASR models for transfer learning purposes. In this chapter, we focus on the second goal of this dissertation: improving an ASR to improve performance on transcribing fillers, which is a well-studied digital biomarker in the cognitive research field, through transfer learning. This problem is under-explored and most research is more focused on preserving the semantic meaning of utterances. We first show that even training an ASR with a large dataset cannot guarantee accurate transcription of fillers regardless of the model's parameter size. Then, we analyze the reason for causing this problem on one pre-trained ASR model. Last, based on our analysis, we propose a transfer learning strategy that can adapt the pre-trained model with minor negative side effects.

6.1 Introduction

In recent years, ASR models that are trained with data from multiple datasets/domains exhibit higher robustness than models trained on a single dataset/domain [122]. Radford quantitatively evaluated how the size of training data and model size impact the trained model's performance (i.e., WER). They showed that, with the same model size (i.e.,

the total number of weights in a DNN), models trained with a larger dataset achieve lower WER than those trained with a smaller dataset. They also showed that, with the same datasize, large models outperform small models on WER across multiple academic datasets. We can expect that we may have an ASR with an even lower WER if we can further enlarge the model size and the size of the training data.

However, there is an innate limitation on WER: it treats every word equally meaning that this measurement prefers models that can correctly high-frequency words. In medical research, researchers might be interested in a specific type of low-frequency words (e.g., fillers, repeated words) and WER cannot correctly reveal an ASR's performance on transcribing those words. For example, let's assume that we have a testing set that has 1000 recordings/utterances. On average, each utterance has 10 words and a 50% chance it has a filler. If an ASR does not transcribe any fillers and makes no other errors, the WER of this ASR on this testing set is 5%. We can consider the ASR performs great on this testing set because its WER is low and it makes no errors that change any utterance's semantic meaning. But, since no fillers are transcribed, we cannot rely on it to study anything related to fillers. Let's call this ASR a flawed ASR where the flaw/problem is not transcribing fillers and cannot be directly used in related research.

The filler is an important and widely studied digital biomarker in medical research, such as autism spectrum disorder [82, 101, 48], Parkinson's disease [62, 139], Alzheimer's disease [45] and mild cognitive impairment [49, 125]. Having an ASR that correctly transcribes fillers is essential for medical research and applications that use filler-related features. In this chapter, we focus on analyzing Whisper ASR models, which are state-of-the-art ASR models and exhibit robustness across multiple domain/testing sets.

In the rest of this chapter, we present the background in Section 6.2. In Section 6.4, we introduce the evaluation matrix. In Section 6.5, we first experimentally show that all cross-lingual Whisper ASR models share the same problem: rarely transcribing fillers (i.e., "um" in English and "嗯" in Chinese). In Section 6.6, using the Whisper ASR model with the largest parameter size as the research target, we show that the problem is mainly caused by the model's acoustic encoder. In Section 6.7, based on the analyzing result from the previous section, we compare the performance of two tuning strategies, only

tune the encoder and only tune the decoder, and present the recommended strategy that brings minimum negative side-effect to the Whisper ASR model. The former not only outperforms the latter on multiple in-language-domain testing sets (i.e., English) but also outperforms the latter on out-of-language-domain test sets (i.e., Chinese).

6.2 Background

In this section, we present an overview of Whisper ASR models in Section 6.2.1. We refer readers to Section 2.9.3 for the details of these models. In Section 6.2.2, we introduce a bootstrap method that is designed for WER's significance analysis.

6.2.1 Whisper

Whisper ASR models [122] refer to a group of models trained with weakly supervised learning that joins multilingual and multitask training. Details of the training procedure are presented in Section 2.9.3. Radford et al. [122] chose a Transformer [155], which is an autoregressive DNN, as the architecture. A Whisper ASR model contains an acoustic encoder that encodes audio representation into acoustic representation and a language decoder that makes next-token predictions conditioned on the acoustic representation and history predictions. Figure 6.1 shows the general architecture. In the encoder, the input spectrogram is first processed by 2 Conv1d blocks. After the Conv1d blocks, the extracted representation is augmented with Sinusoidal positional encoding and then is fed to encoder blocks. The outputs from the top encoder block are referred to as the acoustic representations. In the decoder, both acoustic representations and embedded tokens added with learned positional encoding are fed to decoder blocks. The outputs from the top decoder block are hidden representations that are used to make next-token predictions.

Table 6.1 are experiment models and corresponding model sizes. Radford et al. [122] demonstrated that, with the same large training dataset and training hyperparameters, a larger model achieves lower WER on multilingual speech recognition.

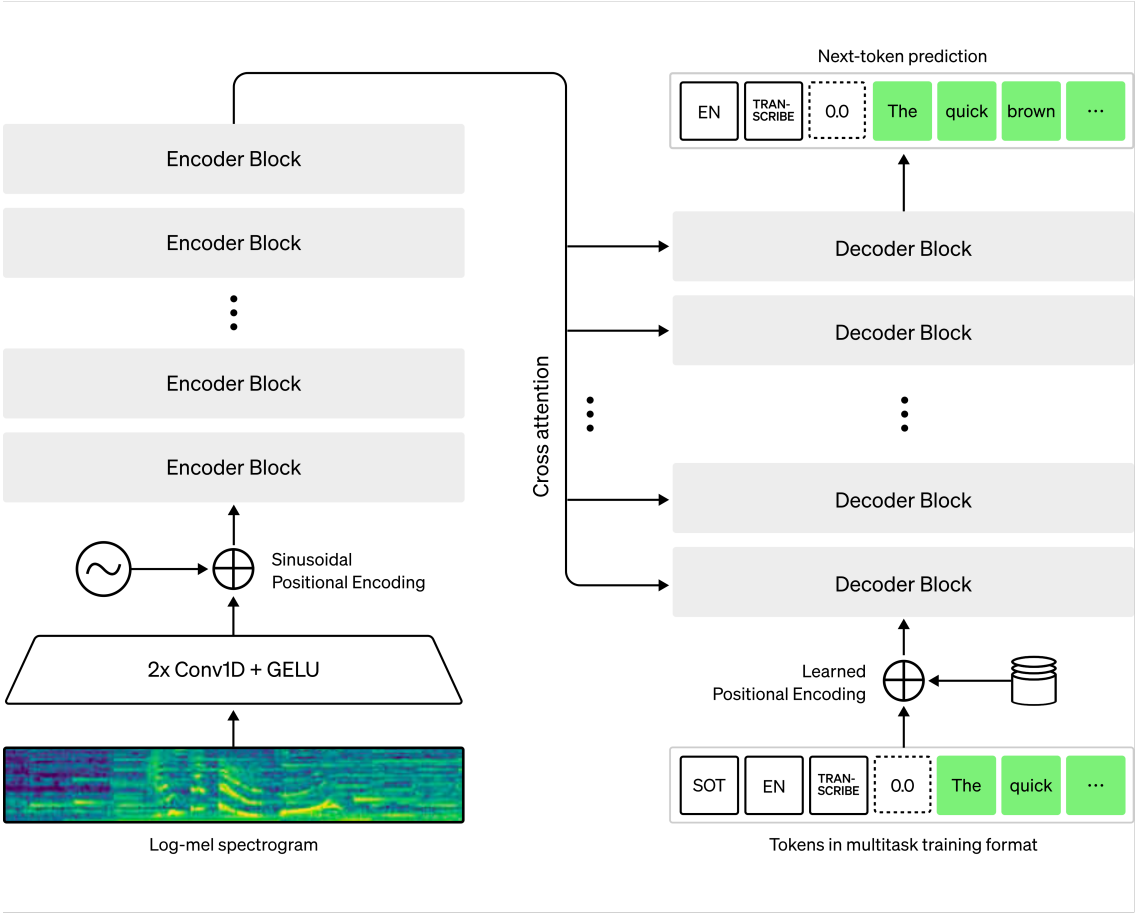

Figure 6.1: The general architecture of Whisper ASR models. The details of the training method including text formatting and training configurations are presented in Section 2.9.3. Picture courtesy: [122].

Model name	Parameters
Whisper-tiny	39 M
Whisper-base	74 M
Whisper-small	244 M
Whisper-medium	769 M
Whisper-large-v1	1550 M

Table 6.1: The parameter size of Whisper models that are used in this chapter.

6.2.2 Speaker-wise Bootstrap Estimation

Many ASR academic datasets provide only one testing set, which makes it difficult to do significance analysis to ensure observed performance differences are not effects of chance. To overcome this limitation, Bisani et al. [21] proposed a bootstrap method based on the assumption that the number of errors for each speaker is independent. Let's assume a testing set has s speakers. The basic procedure is that s speakers' recordings are randomly selected with replacements from the testing set and calculate desired evaluation matrices (e.g., WER) on the sampled testing set. The procedure is repeated B times (e.g., B=10^4) and let the error rate of the bth procedure to be E_b. Then, the bootstrap estimation of the error rate is:

$$E_{boot} = \frac{1}{B}\sum_{b=1}^{B} E_b \tag{6.1}$$

The uncertainty of E_{boot} can be quantified by the standard error, which has the following bootstrap estimate:

$$se_{boot}(E) = \sqrt{\frac{\sum_{b=1}^{B}(E_b - E_{boot})^2}{B-1}} \tag{6.2}$$

Bisani et al. [21] hypothesized that the distribution of E_bs is approximately Gaussian making the true error rate lies with 90% probability in the interval $E_{boot} \pm 1.64 se_{boot}(E)$. One advantage of this bootstrap method is that it does not make any modifications to evaluation matrices. Thus, in this chapter, we use this estimation method on all evaluation matrices including WER and filler error rate which will be introduced in Section 6.4.

6.3 Data

In this section, we describe datasets used in later sections. We evaluate Whisper ASR models on multiple domains/datasets from two languages, namely English and Chinese, whose recordings are recorded from spontaneous conversations where speakers may use fillers: The AMI Meeting Corpus [100], A Spontaneous Chinese-English Dataset (ASCEND) [90], and I-CONECT[1].

[1]More information about I-CONECT can be found at ClinicalTrials.gov: NCT02871921

The AMI Meeting Corpus: consists of 100 hours of meeting recordings. These recordings can be classified into two recording scenarios: individual headset microphones (IHM) and a single distant microphone (SDM). All recordings are manually transcribed and annotated and all speakers are Europeans. The IHM recordings are recorded by headset microphones. The audio quality is clear with minor environmental noises. We refer IHM's domain to the close talk. The SDM recordings are recorded with a distant microphone in a meeting room. The audio quality is poorer than IHM making them harder to be accurately transcribed. We refer SDM's domain to the far field.

ASCEND: is a high-quality Mandarin Chinese-English code-switching corpus built on spontaneous multi-turn conversational dialogue sources collected in Hong Kong. It consists of 10.62 hours of clean speech, collected from 23 bilingual speakers of Chinese and English. Chinese is the native language of all speakers and English is their second language. This corpus contains monolingual Chinese and English utterances and intra-sentential code-switching utterances (i.e., utterances mixed with Chinese and English).

I-CONECT: was conducted at Oregon Health & Science University (OHSU), University of Michigan, and Wayne State University. Socially isolated older adults above 75 years old are recruited from the local Meals on Wheels program in Portland, Oregon and in Detroit, Michigan (with recruitment of African American subjects). Conversations are semi-structured, in which participants freely talk about a predefined topic, i.e., picnic, summertime, swimming and so on, with a moderator online. The speech data was collected from a long-term behavioral research project that uses internet-based social interactions as a tool to enhance older adults' cognitive reserve. English is the native language of all speakers.

6.3.1 Data Process

In this section, we describe the process step of each dataset. Since we use multiple datasets, we unify the name as the following format: "{Source}-{Language}-{Purpose}", where Source refers to the source of the dataset, Language refers to the language of that set and

Purpose refers to the purpose of the set. For example, the training set from AMIIHM would be named AMIIHM-en-train and the testing set from AMISDM would be called AMISDM-en-test.

The AMI Meeting Corpus: to ensure the reproducibility of our experiments, we use a preprocessed corpus from Huggingface [2] and make no modifications. The corpus contains all AMI's datasets used in our experiments, including AMIIHM-en-train, AMIIHM-en-valid, AMIIHM-en-test and AMISDM-en-test.

ASCEND: since we are only interested in the monolingual Chinese/English utterances and only use this corpus to evaluate a model, we select all monolingual English utterances and monolingual Chinese utterances with two or more words from all datasets (i.e., training/validation/testing sets) to create two testing sets: ASCEND-en-test and ASCEND-zh-test. The chosen Chinese word, "嗯", not only serves as a filler in an utterance but is also used to show agreement when using it alone. Since we are only interested in how Whisper ASR models performs on recognizing fillers, we removed all single-word utterances. This process merely impacts the ASCEND-en-test.

I-CONECT: I-CONECT's recordings are manually transcribed and corresponding timestamps are provided. We clip recordings based on the given timestamps and discard audio clips that are longer than the Whisper ASR model's maximum acceptable duration (i.e., 30 seconds). After the process, we have 20 hours of audio clips and they form the testing set: ICONECT-en-test.

6.4 Evaluation Matrix

WER is the commonly used evaluation matrix in the ASR field. But, the downside of WER is that it treats each word equally and mistranscribes low-frequency yet informative words minorly impacting the final WER of that testing sentences group. In the example that is presented in Figure 6.2, the groundtruth sentence is "um ideas are worth nothing

[2] edinburghcstr/ami

um unless they are executed" and the machine transcription is "am ideas are worth um um nothing unless they are executed um". There are 4 errors (i.e., 2 substitutions and 2

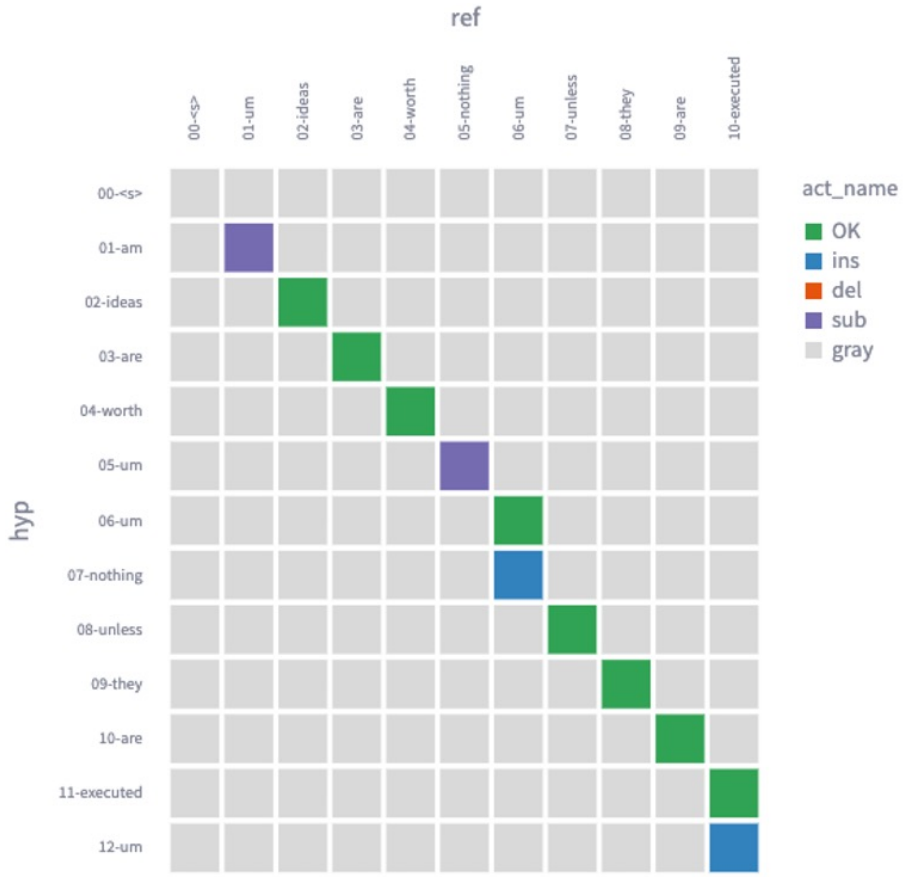

Figure 6.2: A example to compare the difference between WER and FER. The x-axis presents the groundtruth transcription with the word index and the y-axis presents the ASR transcription. An ASR can make three types of errors: transcribes non-existent words (i.e., ins), does not transcribe existent words (i.e., del), and mistranscribes words (i.e., sub). "OK" refers to correct transcriptions.

insertions) in this example, thus, the WER is 40%. But, it is also noticeable that most errors are related to fillers (i.e., "um"). These errors moderately impact the research related to the semantic meaning of the sentence but dramatically impact the research related to the filler of the sentence.

To quantify an ASR's performance on fillers (i.e., um), we can adapt the algorithm of WER and focus on two types of errors: 1, the ASR does not transcribe existing fillers

(false negative), and 2, the ASR transcribes non-existent fillers (false positive). We call this measurement filler error rate (FER). In this example, there is 1 false negative error where "um" is transcribed as "am", and 2 false positive errors where "nothing" is transcribed as "um" and an extra "um" at the end. Since there are two fillers in the groundtruth sentence, then, FER is 150%. Moreover, we can further decompose FER into false-negative FER (FN-FER), where an ASR does not transcribe existing fillers, and false-positive FER (FP-FER), where an ASR transcribes fillers that do not exist in the groudtruth utterance. These two measurements provide information about what types of errors an ASR makes. In our example, the FN-FER is 50% and the FP-FER is 100%. This means that, for this ASR, transcribing non-exist fillers mainly contributes to the high FER. In the following sections, we mainly report two matrices (i.e., WER and FER). When we are interested in the cause of high FER, we report FP-FER and FN-FER.

In the coming sections, we report the statistical results of WER and FER including the mean WER/FER based on bootstrap E_{boot} and its 90% intervals where $B = 10,000$. For Chinese, since Chinese writing is logographic (i.e., every character either represents a word or a minimal unit of meaning) and our chosen Chinese filler is represented by one Chinese character, we decided to treat each Chinese character as a word. We will introduce chosen fillers for both English and Chinese in the coming Section.

6.5 Preliminary Analyses on Transcribing Fillers

Empirically, we notice that Whisper families perform well on retaining the semantic meaning of an utterance, but they do not transcribe fillers that appear in most utterances (e.g., "um" in English and "嗯" in Chinese). An ASR with such characteristics cannot be used by research that uses fillers as an indicator. Thus, it is important to identify this observation.

We conduct our experiment to show that not transcribing filler is a universal problem across all cross-lingual Whisper ASR models that are trained with the same configuration (i.e., whisper-tiny, whisper-base, whisper-small, whisper-medium, and whisper-large-v1). To ensure the reproducibility of this experiment, we choose public validation/testing sets

Model Name	WER	FER	FP-FER	FN-FER
whisper-tiny	35.2(31.7,38.7)	88.1(85.0,91.3)	1.3(0.7,1.8)	86.8(83.7,90.0)
whisper-base	29.5(26.8,32.1)	86.5(82.5,90.5)	0.5(0.2,0.9)	86.0(82.1,89.8)
whisper-small	27.3(24.1,30.6)	89.2(86.2,92.2)	0.7(0.4,1.1)	88.5(85.5,91.5)
whisper-medium	22.1(20.1,24.2)	89.3(86.5,92.1)	0.7(0.4,1.1)	88.5(85.8,91.2)
whisper-large-v1	22.4(20.5,24.3)	86.6(83.1,90.1)	0.6(0.2,1.0)	86.0(82.7,89.4)

Table 6.2: Whisper ASR models' performances on AMIIHM-en-valid.

Model Name	WER	FER	FP-FER	FN-FER
whisper-tiny	40.7(35.7,45.7)	89.4(84.9,94.0)	2.6(1.3,4.0)	86.8(82.5,91.1)
whisper-base	34.2(28.9,39.5)	88.0(83.4,92.6)	3.4(1.9,4.9)	84.6(79.4,89.8)
whisper-small	25.6(23.7,27.4)	89.9(85.0,94.9)	3.4(2.0,4.8)	86.5(81.6,91.4)
whisper-medium	24.0(22.5,25.5)	89.6(85.6,93.5)	2.3(1.0,3.6)	87.3(83.1,91.4)
whisper-large-v1	23.8(22.0,25.5)	87.6(83.7,91.5)	2.3(1.0,3.5)	85.3(81.3,89.4)

Table 6.3: Whisper ASR models' performances on ASCEND-en-test.

Model Name	WER	FER	FP-FER	FN-FER
whisper-tiny	30.2(28.1,32.3)	85.0(81.1,89.0)	3.9(2.7,5.1)	81.1(77.7,84.5)
whisper-base	23.3(21.7,24.9)	75.5(71.7,79.4)	3.4(2.0,4.9)	72.1(68.3,75.9)
whisper-small	16.8(15.6,18.0)	72.4(66.4,78.4)	1.8(1.1,2.6)	70.6(64.4,76.7)
whisper-medium	14.9(13.8,16.0)	76.0(69.5,82.5)	2.9(0.8,5.0)	73.1(66.0,80.2)
whisper-large-v1	14.2(13.1,15.3)	72.1(65.2,79.0)	3.8(2.9,4.7)	68.3(61.2,75.4)

Table 6.4: Whisper ASR models' performances on ASCEND-zh-test.

from the English corpus (i.e., AMIIHM-en-valid, ASCEND-en-test) and Chinese corpus (i.e., ASCEND-zh-test).

We choose fillers with nasal sounds in both languages as evaluating fillers to ensure that fillers from both languages share similar acoustic characteristics. For English, we choose "um" and, for Chinese, we choose "嗯". When evaluating each model's performance, we follow the same post-process as Radford's normalizing the predicted/groundtruth texts.

The results are shown in Table 6.5,Table 6.2 and Table 6.3 for AMIIHM-en-valid, ASCEND-en-test and ASCEND-zh-test, respectively. Regardless of the parameter size of these models and the testing sets, their FERs are all very high which means these models make many errors related to fillers. All models' FP-FER scores are close to 0 and this means these models rarely generate fillers that do not exist in groundtruth texts. All models' FN-FER scores are high and this means fillers in groundtruth texts are rarely transcribed by any model. The results of both FP-FER and FN-FER indicate that rarely

transcribing fillers is the main reason for high FER. Since all models exhibit this problem on both English testing sets (i.e., AMIIHM-en-valid and ASCEND-en-test), the problem is unlikely to be an edge case. Moreover, we also observe a similar situation on transcribing the Chinese testing set (i.e., ASCEND-zh-test), meaning that this problem does not only occur on English testing sets. Overall, these observations support our idea that not transcribing fillers is a universal problem across multiple testing sets and languages.

6.6 Does the encoder cause the problem?

In the previous experiment, we showed that all multilingual Whisper ASR models rarely transcribe fillers on three testing sets from two languages. Even though the training data is not publically available, empirically, it is unlikely that no utterance contains transcribed fillers, considering the fact that these models are trained with 680,000 hours of data collected from the Internet. For example, there might be recordings of daily spontaneous conversations in which speakers use fillers occasionally. Moreover, Radford et al. [122] hypothesize that the linguistic decoders in their large models (i.e., whisper-large-v1) mainly contribute to these models' robustness. Thus, we hypothesize that whisper-large-v1 may not be able to generate robust acoustic representations for fillers. In other words, this model's acoustic encoder might be responsible for this problem.

To prove this hypothesis, we have to find speakers where these speakers' recordings can be accurately transcribed by whisper-large-v1 including fillers. Let's have two testing datasets with the same utterances and the only difference between these two datasets is speakers. If whisper-large-v1 performs noticeably better on one testing dataset than the other (i.e., lower FER and WER), then, the language decoder is not the main reason for the problem. Because both testing datasets contain identical utterances and speaker voices are the only difference.

In the previous section, we found that whisper-large-v1 performs badly (i.e., high FER) on the AMIIHM-en-valid and ASCEND-zh-test. Thus, we only have to find speaker voices whose words can be accurately transcribed by whisper-large-v1. We find that synthesized speeches/recordings can be transcribed accurately including fillers. Although synthesized

recordings from both open-sourced and commercial speech synthesizers can be accurately recognized, we notice that open-sourced speech synthesizers provide limited voices (e.g., most synthesizers learn one female voice) and the acoustic characteristics of synthesized recordings are inconsistent (e.g., different tones for questions and statements). Thus, we choose Microsoft Azure's Speech Service for its diversity of speaker voices and consistent speech quality.

Since Microsoft Azure provides multiple English and Chinese voices for both males and females, we use its English voices to synthesize AMIIHM-en-valid's utterances and use its Chinese voices to synthesize ASCEND-zh-test's utterances. We refer synthesized datasets to S-AMIIHM-en-valid and S-ASCEND-zh-test, respectively. Since our chosen bootstrap method relies on speaker-wise sampling, we pair each speaker with a unique synthesized voice from the same gender group. We use that voice to synthesize all utterances that the speaker said. One limitation is that there are not enough synthesized voices to pair with all speakers in the original testing datasets (i.e., AMIIHM-en-valid and ASCEND-zh-test). To ensure the speaker's voices are the only difference between the synthesized testing dataset and its corresponding original testing dataset (i.e., S-AMIIHM-en-valid and AMIIHM-en-valid, S-ASCEND-zh-test and ASCEND-zh-test), we drop all speakers that cannot be assigned with paired synthesized voices in the original testing datasets and create subsets of these original testing datasets (i.e., C-AMIIHM-en-valid and C-ASCEND-zh-test). That is, S-AMIIHM-en-valid and C-AMIIHM-en-valid have the same number of speakers, gender ratio, and utterances and so do S-ASCEND-zh-test and C-ASCEND-zh-test.

Table 6.5 shows that whisper-large-v1 performs noticeably better on synthesized testing sets than the control testing sets on both WER and FER. Moreover, whisper-large-v1 exhibits much lower FN-FER and FP-FER on synthesized testing sets. Since both testing sets in each pair (i.e., S-AMIIHM-en-valid and C-AMIIHM-en-valid, or S-ASCEND-zh-test and C-ASCEND-zh-test) contain identical utterances, we can exclude all factors related to the language characteristics (e.g., word frequency). In other words, the results indicate that the acoustic encoder is responsible for the problem and the language decoder is affected by the flawed acoustic representation generated by the encoder.

Testing Set	WER	FER	FP-FER	FN-FER
S-AMIIHM-en-valid	8.6(7.6,9.6)	23.0(15.9,30.1)	0.2(0.0,0.5)	22.7(15.5,30.0)
C-AMIIHM-en-valid	22.5(20.0,25.0)	87.9(84.6,91.2)	0.5(0.1,0.8)	87.4(84.3,90.5)
S-ASCEND-zh-test	5.5(4.4,6.6)	16.2(11.8,20.6)	2.5(1.3,3.8)	13.7(9.5,17.8)
C-ASCEND-zh-test	16.3(14.3,18.3)	99.6(63.7,135.5)	31.6(0.0,67.0)	68.0(60.8,75.1)

Table 6.5: The performance of whisper-large-v1 on synthesized/experimental and controlled testing sets.

Name of hyperparameters	Value
Training batch size	64
Learning rate	1×10^{-5}
Training steps	4000

Table 6.6: Training configuration for fine-tune whisper-large-v1.

6.7 Evaluate various fine-tuning strategies

In the previous two experiments, we showed that Whisper ASR models rarely transcribe fillers regardless of the model size, and indicated that the acoustic encoder of whisper-large-v1 likely causes this problem. The latter suggests that we have two tuning strategies for resolving the problem: 1) only tune the encoder to improve the robustness of the encoder's acoustic representation; 2) only tune the decoder to adapt the encoder's acoustic representation. There are two differences between these two tuning strategies. First, the former may preserve the decoder's multi-lingual ability while the latter may not. Validating this is essential as it may change the data enlarging strategy (e.g., enlarge train data through gathering corpus from the same language domain, or, from multiple language domains). Second, empirically, tuning the decoder would be more efficient than tuning the encoder. However, it is unknown whether this empirical result is still valid on a large model (i.e., whisper-large-v1). Since our goal is to compare the tuning efficiency between the tuning encoder and the tuning decoder in this section, we do not evaluate the performance of tuning the whole model.

We tune the whisper-large-v1 with the same training configuration that is shown in Table 6.6. We use the tuning strategy to refer to tuned models (i.e., tune-encoder or tune-decoder) and we refer to the original model as "no tuning".

We use AMIIHM-en-train as the only training dataset and evaluate tuned models on

Data Name	Tuning Strategy	WER	FER
AMIIHM-en-test	No tuning	20.7(18.6,22.7)	87.8(86.3,89.3)
	Tune encoder	8.9(7.4,10.4)	9.5(7.2,11.9)
	Tune decoder	9.5(7.9,11.2)	11.1(8.4,13.7)
AMISDM-en-test	No tuning	42.4(38.5,46.2)	92.7(91.3,94.0)
	Tune encoder	31.3(27.3,35.3)	31.9(25.8,37.9)
	Tune decoder	34.5(30.1,38.9)	31.9(25.3,38.6)
ASCEND-en-test	No tuning	23.8(22.0,25.5)	87.6(83.7,91.5)
	Tune encoder	22.6(21.0,24.2)	69.2(61.2,77.2)
	Tune decoder	24.6(20.5,28.6)	68.8(61.0,76.5)
ICONECT-en-test	No tuning	20.4(19.3,21.5)	96.1(95.2,97.0)
	Tune encoder	19.3(17.5,21.2)	39.5(32.3,46.7)
	Tune decoder	17.2(15.7,18.7)	37.8(30.8,44.7)
ASCEND-zh-test	No tuning	14.2(13.1,15.3)	72.1(65.2,79.0)
	Tune encoder	12.8(11.3,14.3)	74.6(70.1,79.2)
	Tune decoder	17.6(16.3,18.9)	96.2(94.3,98.2)

Table 6.7: Comparing tuned models across multiple testing sets.

AMIIHM-en-test for in-domain evaluation, on AMISDM-en-test, ICONECT-en-test, and ASCEND-en-test for out-of-domain evaluation, and ASCEND-zh-test for cross-language evaluation. Recordings in AMIIHM-en-train and AMIIHM-en-test are recorded by headset microphones and are known as close-talk speech. Speakers in both datasets are European speakers and all utterances are English. Recordings in AMISDM-en-test are similar to AMIIHM-en-train except they are recorded by distant microphones and are known as far-field speech. Recordings in ASCEND-en-test are close-talk speeches but all English utterances belong to Chinese speakers. We use this testing dataset to evaluate the impact of accent difference. Recordings in ICONECT-en-test are also close-talk speeches but speakers are older adults (i.e., 75+ years old). We use this testing dataset to evaluate how models perform on aging voices. Recordings in ASCEND-zh-test are close-talk speeches but all utterances are Chinese. We use this testing dataset to evaluate how fine-tuned models (i.e., tune-encoder and tune-decoder) perform on out-of-language domain utterances.

We present experiment results in Table 6.7. First, we compare the three models' performance on the in-domain testing dataset. The tune-encoder model exhibits much lower WER and FER than the pre-trained model (labeled as "no tuning" in the table)

and also performs marginally better than the tune-decoder model. Second, we compare these models' performance on the far-filed testing dataset (AMISDM-en-test). The tune-encoder model exhibits noticeably lower WER and FER than the pre-trained model and also performs marginally better than the tune-decoder model. Third, we compare these models' performance on the Chinese accent testing dataset (ASCEND-en-test). The tune-encoder model exhibits marginally lower WER than the pre-trained model and performs noticeably better than the pre-trained model on FER. Moreover, the tune-encoder model is comparable with the tune-decoder model. Fourth, we compare these models' performance on aging voices (ICONECT-en-test). The tune-encoder model exhibits comparable WER to the pre-trained model and performs much better than the pre-trained model on FER. Interestingly, the tune-decoder model performs the best on both WER and FER. It might caused by the acoustic mismatch between the training dataset and this testing dataset. Because there are noticeable articulation differences between older adults and younger adults. Last, we evaluate two fine-tuned models' performance on the out-of-language domain testing dataset (ASCEND-zh-test). The tune-encoder model exhibits comparable WER and FER to the pre-trained model while the tune-decoder model performs worse than the pre-trained model on both WER and FER.

Overall, the tune-encoder model not only performs equivalently as the pre-trained model on WER but also outperforms the pre-trained model on FER in all testing datasets even on the ASCEND-zh-test which is not the same language as the training dataset (i.e., AMIIHM-en-train). However, the tune-decoder model does not perform better than the tune-encoder model in most testing sets and the tune-decoder model performs noticeably worse than the baseline on Chinese. The results support our hypothesis that tuning the encoder preserves the decoder's multi-lingual ability while achieving comparable performance to the other strategy. In other words, tuning the encoder is a recommended strategy for adapting whisper-large-v1 for transcribing fillers.

6.8 Conclusion

In this chapter, we showed that both large training datasets and scaling model size do not guarantee ASRs transcribing fillers through showing that all cross-lingual whisper models rarely transcribe fillers. More importantly, we demonstrated that WER cannot reveal the problem accurately. Then, considering the largest Whisper ASR model (i.e., whisper-large-v1) as the research target, we demonstrate that the model's acoustic encoder is the main reason. Based on the analyzing result, we show that tuning the encoder can resolve this problem on the large ASR while not negatively impacting the desired abilities that the ASR model originally had.

In the future, we will further analyze what acoustic characteristics cause the problem of not transcribing fillers. This is an important research direction for training data expansion.

Chapter 7

Conclusion And Future Work

In this dissertation, we focused on developing DNN-based ASR models relevant to automatic cognitive assessment systems. Oriented to the idea of transfer learning, we focused on resolving two specific problems. 1) Improve ASRs' performance in older adults with possible cognitive impairment. 2) Improve an ASR model transcribing accuracy on filled pauses, which is a well-studied indicator in the cognitive research field. Additionally, to improve accuracy in assessing mild cognitive impairment, we proposed time-related features that can be extracted from the animal fluency test.

For the first problem, we explored two aspects of transfer learning: develop transfer learning techniques that increase the tuning efficiency from a limited training dataset, and build pre-trained models with a small training dataset. In Chapter 4, we presented a transfer learning technique that utilizes the base model's intermediate outputs to improve the performance in older adults (75+ years old) and showed that our technique outperforms standard fine-tuning. We first demonstrated that speakers' age differences dramatically impact an ASR performance through comparing the performance difference between academic testing sets and audio recordings from older adults. Then, based on our hypothesis that intermediate outputs from a pre-trained DNN model contain useful information for the target domain, we proposed a conditional independent mechanism to assign arbitrary usefulness scores to all outputs including intermediate outputs and the final hidden outputs, and introduced a training method so that the mechanism can learn the usefulness from the training data. We experimentally proved our hypothesis on a pre-trained ASR model called Deepspeech2 [7]. Our method achieves better performance than the standard fine-tuning when there are only 10 hours of training data. Our best

model performs 6% better than the standard fine-tuned model.

Since we have shown the intermediate outputs contain useful information for the target domain, developing DNN layers that can utilize these outputs more efficiently would be one of our next goals. The other goal is to develop DNN layers that can dynamically assign usefulness scores based on certain factors (e.g., fundamental frequency, articulation, etc).

In Chapter 5, we assumed that collecting 100 hours of transcribed recordings for training a base/pre-trained model is achievable in the cognitive research field and present a DNN architecture that can fully utilize such a training dataset. We refine a well-studied DNN architecture in order to improve the training efficiency on a small training dataset and a small DNN architecture (i.e., having a small number of parameters). As a preliminary study, we evaluate the refined DNN on a small academic dataset. We first identified two potential refining ideas through analyzing a pre-trained ASR model: replacing several global attention mechanism layers in the DNN architecture with local attention mechanism layers and applying parameter sharing across multiple attention blocks. We evaluated these ideas separately and showed their effectiveness. Then, we showed that refining the DNN with both ideas further improves the training efficiency. Training on the same dataset, our refined DNN performed 16% better than the original DNN even though the parameter size of our DNN was 75% less than the latter. The reduction in parameter size makes our DNN suitable for increasing the mobility of DNN-based ASRs which enables the deployment of offline systems on mobile devices.

In Chapter 5, we only made minor modifications to the original DNN architecture in order to show that our ideas, replacing global attention mechanism layers with fixed-window-size local attention mechanism layers and applying parameter sharing to the architecture, mainly contribute to the improvement of training efficiency. In the future, we can further improve training efficiency through developing more efficient local attention mechanisms and parameter-sharing techniques. We believe developing local attention mechanisms that can dynamically adjust the window size will be beneficial because it allows the DNN model to automatically adjust the optimal window size based on the input and free us from interpreting the optimal window size. Various parameter-sharing techniques [34, 83, 18] can be adapted to further improve the training efficiency.

For the second problem of this dissertation, accurately transcribing fillers is essential but gains much less attention than it should be because various research has shown scaling training data and DNN model size increase the performance and robustness when evaluating on the mainstream evaluation metric of WER.

In Chapter 6, we first demonstrated that the untranscribed filler problem is universal across all multi-lingual Whisper ASR models and showed that the problem is unlikely to be resolved by scaling model size. We showed that these models do not transcribe filled pauses on both Chinese and English utterances. We demonstrate that the mainstream evaluation matric cannot reveal the problem accurately. Then, we focused on analyzing why the largest Whisper ASR (i.e., whisper-large-v1) does not transcribe filled pauses. We introduced our hypothesis that the model's acoustic encoder does not generate robust acoustic representations for filled pauses. To prove our hypothesis, we show that, with the same utterances, some voices' recordings can be accurately transcribed including transcribing filled pauses. This observation is universal on transcribing English and Chinese recordings. This indicated that tuning the acoustic encoder with spontaneous conversation recordings can resolve the not-transcribe-filled-pause problem. We compare two fine-tuning strategies: only tuning the acoustic encoder and only tuning the linguistic decoder. We showed that the former outperforms the latter in most testing sets including the out-of-language-domain testing set. We conclude that only tuning the acoustic encoder is an effective method that can resolve the not-transcribe-filled-pause problem on whisper-large-v1 while not negatively impacting the ability of multi-lingual transcription that the ASR originally had.

There are three future directions to further improve the ASR's performance on transcribing filler. One of the future directions is to increase the domain similarity between the training dataset and the testing dataset. In this chapter, we have shown that whisper-large-v1 performs great on synthesized audio recordings. Analyzing the acoustic difference between synthesized recordings and older adults' recordings will benefit synthesizing recordings that are acoustically similar to older adults. The second future direction is utilizing intermediate outputs from the acoustic encoder in order to generate robust acoustic representations for fillers. In Chapter 4, we have shown that intermediate outputs contain

useful information for target domains. We have evaluated the idea on a non-autoregressive DNN (i.e., Deepspeech2), but Whisper ASR models belong to the autoregressive DNN family. There should be further investigation on how to utilize the acoustic encoder's intermediate outputs when the pre-trained model belongs to the autoregressive DNN family. The third future direction, which is also targeted at generating robust acoustic representations, is training a new acoustic encoder using our DNN architecture proposed in Chapter 5. We can use our refined DNN architecture to replace the original acoustic encoder and train the new acoustic encoder on the target domain's recordings (e.g., recordings from older adults) to further improve Whisper ASR models' performance on the target domain.

In addition to improving ASRs when there is training data available, we also developed features for increasing the accuracy of cognitive assessment. In Chapter 3, we utilized a GMM-HMM ASR as an aligner to extract timestamps of manual transcriptions, which is widely used in phonetic studies [19, 81, 142] and developed time-base features extracted from the animal fluency test. We utilized these features to distinguish older people with mild cognitive impairment from those with normal cognition. We introduced the animal fluency test which evaluates a person's semantic retrieval ability. We introduced the semantic retrieval pattern, clustering and switching. We introduced the research based on the semantic retrieval pattern and this research inspired our time-based features. This research used the time difference between two adjacent animal names to quantify the retrieval difficulty. We show that, regardless of the clustering and switching method, using both count- and time-based features outperforms better than using only count-based features. When we use the Troyer-based method, the former is 10% better than the latter. When we use the ESA-based method, the former is 5% better than the latter. This result indicates that our features provide information that count-based features do not capture.

In this chapter, we evaluated our features that are extracted from manual transcriptions. To develop an automatic cognitive diagnosis system, evaluating these features extracted from ASR transcription is essential.

Appendix A

Research Summary

We conducted the following research at Oregon Health & Science University.

- Chen, Liu, Hiroko H. Dodge, and Meysam Asgari. "Older people with mild cognitive impairment exhibit lower semantic noise after six months of frequent social conversations." Alzheimer's Association International Conference. ALZ, 2023. [31]
- Chen, Liu, Meysam Asgari, and Hiroko H. Dodge. "Efficacy in linguistic characteristics: I-CONECT project." Alzheimer's & Dementia 18 (2022): e059652. [26]
- Chen, Liu, Hiroko H. Dodge, and Meysam Asgari. "Measures of Voice Quality as Indicators of Mild Cognitive Impairment." Alzheimer's & Dementia 18 (2022): e067393. [30]
- MacFarlane, Heather, et al. "Combining voice and language features improves automated autism detection." Autism Research 15.7 (2022): 1288-1300. [94]
- Chen, Liu, Meysam Asgari, and Hiroko H. Dodge. "Optimize Wav2vec2s Architecture for Small Training Set Through Analyzing its Pre-Trained Models Attention Pattern." ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022. [27]
- Asgari, Meysam, Liu Chen, and Eric Fombonne. "Quantifying voice characteristics for detecting autism." Frontiers in Psychology 12 (2021): 665096. [10]
- Chen, Liu, and Meysam Asgari. "Refining automatic speech recognition system for older adults." ICASSP 2021-2021 IEEE International Conference on Acoustics,

Speech and Signal Processing (ICASSP). IEEE, 2021. [25]

- Chen, Liu, et al. "Improving the assessment of mild cognitive impairment in advanced age with a novel multi-feature automated speech and language analysis of verbal fluency." Frontiers in Psychology 11 (2020): 535. [28]

- Chen, Liu, Hiroko H. Dodge, and Meysam Asgari. "Topic-based measures of conversation for detecting mild cognitive impairment." Proceedings of the conference. Association for Computational Linguistics. Meeting. Vol. 2020. NIH Public Access, 2020. [29]

- Gale, Robert, et al. "Improving asr systems for children with autism and language impairment using domain-focused dnn transfer techniques." Interspeech. Vol. 2019. NIH Public Access, 2019. [43]

Bibliography

[1] Neural Networks, Types, and Functional Programming – colahapos;s blog — colah.github.io. http://colah.github.io/posts/2015-09-NN-Types-FP/. [Accessed 25-07-2023].

[2] People's Speech — mlcommons.org. https://mlcommons.org/en/peoples-speech/. [Accessed 25-07-2023].

[3] Student Notes: Convolutional Neural Networks (CNN) Introduction — indoml.com. https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/. [Accessed 13-08-2023].

[4] Osama Abdeljaber, Onur Avci, Serkan Kiranyaz, Moncef Gabbouj, and Daniel J Inman. Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. *Journal of Sound and Vibration*, 388:154–170, 2017.

[5] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

[6] Minna Alenius, Sanna Koskinen, Ilona Hallikainen, Tiia Ngandu, Jari Lipsanen, Päivi Sainio, Annamari Tuulio-Henriksson, and Tuomo Hänninen. Cognitive performance among cognitively healthy adults aged 30–100 years. *Dementia and geriatric cognitive disorders extra*, 9(1):11–23, 2019.

[7] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang,

Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2 : End-to-end speech recognition in english and mandarin. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 173–182, New York, New York, USA, 20–22 Jun 2016. PMLR.

[8] Junyi Ao, Rui Wang, Long Zhou, Shujie Liu, Shuo Ren, Yu Wu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, Yao Qian, Jinyu Li, and Furu Wei. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. 2021.

[9] Junyi Ao, Zi-Hua Zhang, Long Zhou, Shujie Liu, Haizhou Li, Tom Ko, Lirong Dai, Jinyu Li, Yao Qian, and Furu Wei. Pre-training transformer decoder for end-to-end asr model with unpaired speech data. 2022.

[10] Meysam Asgari, Liu Chen, and Eric Fombonne. Quantifying voice characteristics for detecting autism. *Frontiers in Psychology*, 12:665096, 2021.

[11] Meysam Asgari, Jeffrey Kaye, and Hiroko Dodge. Predicting mild cognitive impairment from spontaneous spoken utterances. *Alzheimer's & Dementia: Translational Research & Clinical Interventions*, 3(2):219–228, 2017.

[12] Onur Avci, Osama Abdeljaber, Serkan Kiranyaz, Mohammed Hussein, and Daniel J Inman. Wireless and real-time structural damage detection: A novel decentralized method for wireless sensor networks. *Journal of Sound and Vibration*, 424:158–172, 2018.

[13] Onur Avci, Osama Abdeljaber, Serkan Kiranyaz, and Daniel Inman. Structural damage detection in real time: implementation of 1d convolutional neural networks for shm applications. In *Structural Health Monitoring & Damage Detection, Volume 7: Proceedings of the 35th IMAC, A Conference and Exposition on Structural Dynamics 2017*, pages 49–54. Springer, 2017.

[14] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[16] Rohan Badlani, Adrian Łańcucki, Kevin J Shih, Rafael Valle, Wei Ping, and Bryan Catanzaro. One tts alignment to rule them all. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6092–6096. IEEE, 2022.

[17] Alexei Baevski, Henry Zhou, Abdel rahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *ArXiv*, abs/2006.11477, 2020.

[18] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*, 2019.

[19] George Bailey. Automatic detection of sociolinguistic variation using forced alignment. In *University of Pennsylvania Working Papers in Linguistics: Selected Papers from New Ways of Analyzing Variation (NWAV 44)*, pages 10–20. York, 2016.

[20] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[21] Maximilian Bisani and Hermann Ney. Bootstrap estimates for confidence intervals

in asr performance evaluation. *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:I–409, 2004.

[22] Antoinette L Bouhuys, Harm K Schutte, Domien GM Beersma, and George LJ Nieboer. Relations between depressed mood and vocal parameters before, during and after sleep deprivation: a circadian rhythm study. *Journal of affective disorders*, 19(4):249–258, 1990.

[23] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[24] Eric L. Charnov. Optimal foraging, the marginal value theorem. *Theoretical population biology*, 9 2:129–36, 1976.

[25] Liu Chen and Meysam Asgari. Refining automatic speech recognition system for older adults. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7003–7007. IEEE, 2021.

[26] Liu Chen, Meysam Asgari, and Hiroko H Dodge. Efficacy in linguistic characteristics: I-conect project. *Alzheimer's & Dementia*, 18:e059652, 2022.

[27] Liu Chen, Meysam Asgari, and Hiroko H Dodge. Optimize wav2vec2s architecture for small training set through analyzing its pre-trained models attention pattern. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7112–7116. IEEE, 2022.

[28] Liu Chen, Meysam Asgari, Robert Gale, Katherine Wild, Hiroko Dodge, and Jeffrey Kaye. Improving the assessment of mild cognitive impairment in advanced age with

a novel multi-feature automated speech and language analysis of verbal fluency. *Frontiers in Psychology*, 11:535, 2020.

[29] Liu Chen, Hiroko H Dodge, and Meysam Asgari. Topic-based measures of conversation for detecting mild cognitive impairment. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2020, page 63. NIH Public Access, 2020.

[30] Liu Chen, Hiroko H Dodge, and Meysam Asgari. Measures of voice quality as indicators of mild cognitive impairment. *Alzheimer's & Dementia*, 18:e067393, 2022.

[31] Liu Chen, Hiroko H Dodge, and Meysam Asgari. Older people with mild cognitive impairment exhibit lower semantic noise after six months of frequent social conversations. In *Alzheimer's Association International Conference*. ALZ, 2023.

[32] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[33] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[34] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[35] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[36] Hiroko H Dodge, Kexin Yu, Chao-Yi Wu, Patrick J Pruitt, Meysam Asgari, Jeffrey A Kaye, Benjamin M Hampstead, Laura Struble, Kathleen Potempa, Peter Lichtenberg, Raina Croff, Roger L Albin, Lisa C Silbert, and I-CONECT team.

Internet-based conversational engagement randomized controlled clinical trial (I-CONECT) among socially isolated adults 75+ years old with normal cognition or MCI: topline results. *Gerontologist*, nov 2023.

[37] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691, 2017.

[38] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[39] Marianne Farina, Dalton Breno Costa, Joao Andre Webber de Oliveira, Manuela Polidoro Lima, Wagner De Lara Machado, Carmen Moret-Tatay, Regina Maria Fernandes Lopes, Irani Iracema De Lima Argimon, and Tatiana Quarti Irigaray. Cognitive function of brazilian elderly persons: longitudinal study with non-clinical community sample. *Aging & mental health*, pages 1–8, 2019.

[40] Siyuan Feng, Olya Kudina, Bence Mark Halpern, and Odette Scharenborg. Quantifying bias in automatic speech recognition. *ArXiv*, abs/2103.15122, 2021.

[41] Marshal F Folstein, Susan E Folstein, and Paul R McHugh. "mini-mental state": a practical method for grading the cognitive state of patients for the clinician. *Journal of psychiatric research*, 12(3):189–198, 1975.

[42] Kathleen C Fraser, Jed A Meltzer, and Frank Rudzicz. Linguistic features identify alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422, 2016.

[43] Robert Gale, Liu Chen, Jill Dolata, Jan Van Santen, and Meysam Asgari. Improving asr systems for children with autism and language impairment using domain-focused dnn transfer techniques. In *Proc. Interspeech 2019*, pages 11–15, 2019.

[44] Daniel Galvez, Greg Diamos, Juan Manuel Ciro Torres, Juan Felipe Cerón, Keith Achorn, Anjali Gopi, David Kanter, Max Lam, Mark Mazumder, and Vijay Janapa Reddi. The people's speech: A large-scale diverse english speech recognition dataset for commercial usage. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[45] Frédérique Gayraud, Hyeran Lee, and Melissa Barkat-Defradas. Syntactic and lexical context of pauses and hesitations in the discourse of alzheimer patients and healthy elderly subjects. *Clinical Linguistics & Phonetics*, 25:198 – 209, 2011.

[46] Roberto Gemello, Franco Mana, Stefano Scanzio, Pietro Laface, and Renato De Mori. Adaptation of hybrid ann/hmm models using linear hidden transformations and conservative training. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE, 2006.

[47] Keith W Godin, Taufiq Hasan, and John HL Hansen. Glottal waveform analysis of physical task stress speech. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[48] Kyle Gorman, Lindsay A Olson, Alison Presmanes Hill, Rebecca Lunsford, Peter A. Heeman, and Jan van Santen. Uh and um in children with autism spectrum disorders or language impairment. *Autism Research*, 9, 2016.

[49] Gábor Gosztolya, László Tóth, Tamás Grósz, Veronika Vincze, Ildikó Hoffmann, Gréta Szatlóczki, Magdolna Pákáski, and János Kálmán. Detecting mild cognitive impairment from spontaneous speech by correlation-based phonetic feature selection. In *Interspeech*, 2016.

[50] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[51] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[52] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[53] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. *ArXiv*, abs/2005.08100, 2020.

[54] Guanbo Wang Jiayu Du Wei-Qiang Zhang Chao Weng Dan Su Daniel Povey Jan Trmal Junbo Zhang Mingjie Jin Sanjeev Khudanpur Shinji Watanabe Shuaijiang Zhao Wei Zou Xiangang Li Xuchen Yao Yongqing Wang Yujun Wang Zhao You Zhiyong Yan Guoguo Chen, Shuzhou Chai. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio. In *Proc. Interspeech 2021*, 2021.

[55] Isabelle M Guyon, Jason Weston, Stephen D. Barnhill, and Vladimir Naumovich Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[56] James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143 1:29–36, 1982.

[57] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Gregory Frederick Diamos, Erich Elsen, Ryan J. Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and A. Ng. Deep speech: Scaling up end-to-end speech recognition. *ArXiv*, abs/1412.5567, 2014.

[58] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[59] Thomas T. Hills, Michael N. Jones, and Peter M. Todd. Optimal foraging in semantic memory. *Psychological Review*, 119 2:431–40, 2012.

[60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[61] Richard J Holden, Amanda M McDougald Scott, Peter LT Hoonakker, Ann S Hundt, and Pascale Carayon. Data collection challenges in community settings: Insights from two field studies of patients with chronic disease. *Quality of Life Research*, 24(5):1043–1055, 2015.

[62] Jessica E. Huber and Meghan Darling. Effect of parkinson's disease on the production of structured and unstructured speaking tasks: respiratory physiologic and linguistic considerations. *Journal of speech, language, and hearing research : JSLHR*, 54 1:33–46, 2011.

[63] P.P.M Hurks, J.G.M Hendriksen, J.S.H Vles, A.C Kalff, F.J.M Feron, M Kroes, T.M.C.B van Zeben, J Steyaert, and J Jolles. Verbal fluency over time as a measure of automatic and controlled processing in children with adhd. *Brain and Cognition*, 55(3):535–544, 2004.

[64] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[65] Ahmed Ismail, Samir Abdlerazek, and Ibrahim M El-Henawy. Development of smart healthcare system based on speech recognition using support vector machine and dynamic time warping. *Sustainability*, 12(6):2403, 2020.

[66] Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

[67] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[68] Ali Khodabakhsh, Fatih Yesil, Ekrem Guner, and Cenk Demiroglu. Evaluation of linguistic and prosodic features for detection of alzheimer's disease in turkish conversational speech. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):9, 2015.

[69] Joe Kilian and Hava T Siegelmann. The dynamic universality of sigmoidal neural networks. *Information and computation*, 128(1):48–56, 1996.

[70] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj. Real-time patient-specific ecg classification by 1-d convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, 63(3):664–675, 2015.

[71] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj. Personalized monitoring and advance warning system for cardiac arrhythmias. *Scientific reports*, 7(1):9270, 2017.

[72] Serkan Kiranyaz, Turker Ince, Ridha Hamila, and Moncef Gabbouj. Convolutional neural networks for patient-specific ecg classification. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2608–2611. IEEE, 2015.

[73] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *ArXiv*, abs/2304.02643, 2023.

[74] John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. 2001.

[75] Alexandra König, Nicklas Linz, Johannes Tröger, Maria Wolters, Jan Alexandersson, and Phillipe Robert. Fully automatic speech-based analysis of the semantic verbal fluency task. *Dementia and Geriatric Cognitive Disorders*, 45(3-4):198–209, 2018.

[76] Alexandra Konig, Aharon Satt, Alex Sorin, Ran Hoory, Alexandre Derreumaux, Renaud David, and Phillippe H Robert. Use of speech analyses within a mobile

application for the assessment of cognitive impairment in elderly people. *Current Alzheimer Research*, 15(2):120–129, 2018.

[77] Alexandra König, Aharon Satt, Alexander Sorin, Ron Hoory, Orith Toledo-Ronen, Alexandre Derreumaux, Valeria Manera, Frans Verhey, Pauline Aalten, Phillipe H Robert, et al. Automatic speech analysis for the assessment of patients with predementia and alzheimer's disease. *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring*, 1(1):112–124, 2015.

[78] Rinat Koren, Ora Kofman, and Andrea Berger. Analysis of word clustering in verbal fluency of school-aged children. *Archives of Clinical Neuropsychology*, 20(8):1087–1104, 12 2005.

[79] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, 2019.

[80] Geza Kiss Kyle Gorman. ldamatch: Selection of statistically similar research groups. In *https: //cran. r-project. org/web/packages/ldamatch/index. html*, 2016-06-27.

[81] William Labov, Ingrid Rosenfelder, and Josef Fruehwald. One hundred years of sound change in philadelphia: Linear incrementation, reversal, and reanalysis. *Language*, 89(1):30–65, 2013.

[82] Johanna K. Lake, Karin R. Humphreys, and Shannon Cardy. Listener vs. speaker-oriented aspects of speech: Studying the disfluencies of individuals with autism spectrum disorders. *Psychonomic Bulletin & Review*, 18:135–140, 2011.

[83] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2019.

[84] Linda Lang, Angela Clifford, Li Wei, Dongmei Zhang, Daryl Leung, Glenda Augustine, Isaac M Danat, Weiju Zhou, John R Copeland, Kaarin J Anstey, et al.

Prevalence and determinants of undetected dementia in the community: a systematic literature review and a meta-analysis. *BMJ open*, 7(2):e011146, 2017.

[85] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[86] Bo Li and Khe Chai Sim. Comparison of discriminative input and output transformations for speaker adaptation in the hybrid nn/hmm systems. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[87] Xiangyu Liang, Zhiyong Wu, Runnan Li, Yanqing Liu, Sheng Zhao, and Helen Meng. Enhancing monotonicity for robust autoregressive transformer tts. In *INTERSPEECH*, pages 3181–3185, 2020.

[88] Sue Ellen Linville. Source characteristics of aged voice assessed from long-term average spectra. *Journal of Voice*, 16(4):472–479, 2002.

[89] Sue Ellen Linville and Jennifer Rens. Vocal tract resonance analysis of aging voice using long-term average spectra. *Journal of Voice*, 15(3):323–330, 2001.

[90] Holy Lovenia, Samuel Cahyawijaya, Genta Indra Winata, Peng Xu, Xu Yan, Zihan Liu, Rita Frieske, Tiezheng Yu, Wenliang Dai, Elham J. Barezi, and Pascale Fung. Ascend: A spontaneous chinese-english dataset for code-switching in multi-turn conversation. In *International Conference on Language Resources and Evaluation*, 2021.

[91] Jian Luo, Jianzong Wang, Ning Cheng, Zhenpeng Zheng, and Jing Xiao. Adaptive activation network for low resource multilingual speech recognition. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2022.

[92] Lin Luo, Gigi Luk, and Ellen Bialystok. Effect of language proficiency and executive control on verbal fluency performance in bilinguals. *Cognition*, 114(1):29–41, 2010.

[93] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[94] Heather MacFarlane, Alexandra C Salem, Liu Chen, Meysam Asgari, and Eric Fombonne. Combining voice and language features improves automated autism detection. *Autism Research*, 15(7):1288–1300, 2022.

[95] Francisco Martínez-Sánchez, García Meilán JJ, Enrique Pérez, Juan Carro, and José María Arana. Expressive prosodic patterns in individuals with alzheimer's disease. *Psicothema*, 24(1):16–21, 2012.

[96] Francisco Martínez-Sánchez, Juan JG Meilán, Juan Antonio Vera-Ferrandiz, Juan Carro, Isabel M Pujante-Valverde, Olga Ivanova, and Nuria Carcavilla. Speech rhythm alterations in spanish-speaking individuals with alzheimer's disease. *Aging, Neuropsychology, and Cognition*, 24(4):418–434, 2017.

[97] Pavagada S Mathuranath, Annamma George, Praseetha Cherian, Aley Alexander, S. Gangadhara Sarma, and PS Sarma. Effects of age, education and gender on verbal fluency. *Journal of Clinical and Experimental Neuropsychology*, 25:1057 – 1064, 2003.

[98] Reinhold MAYR, Ulrich; KLIEGL. Task-set switching and long-term memory retrieval. *Journal of experimental psychology. Learning, memory, and cognition*, 2000.

[99] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi. In *Proc. Interspeech 2017*, pages 498–502, 2017.

[100] Iain McCowan, Jean Carletta, Wessel Kraaij, Simone Ashby, Sebastien Bourban, Mike Flynn, Maël Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska, Wilfried Post, Dennis Reidsma, and Pierre D. Wellner. The ami meeting corpus. 2005.

[101] Karla K. McGregor and Rex R. Hadden. Brief report: "um" fillers distinguish children with and without asd. *Journal of Autism and Developmental Disorders*, 50:1816–1821, 2018.

[102] Juan JG Meilán, Francisco Martínez-Sánchez, Israel Martínez-Nicolás, Thide E Llorente, and Juan Carro. Changes in the rhythm of speech difference between people with nondegenerative mild cognitive impairment and with preclinical dementia. *Behavioural neurology*, 2020, 2020.

[103] Drahomír Michalko, Martin Marko, and Igor Riečanskỳ. Executive functioning moderates the decline of retrieval fluency in time. *Psychological Research*, 87(2):397–409, 2023.

[104] Fred D Minifie. *Introduction to communication sciences and disorders*. Singular Publishing Group, Incorporated, 1994.

[105] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of convolution neural network advances on the imagenet. *Computer vision and image understanding*, 161:11–19, 2017.

[106] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[107] John Carl Morris, Christopher Ernesto, Kimberly Schafer, M Coats, S Leon, M Sano, LJ Thal, and P Woodbury. Clinical dementia rating training and reliability in multicenter studies: the alzheimer's disease cooperative study. *Neurology*, 48(6):1508–1510, 1997.

[108] Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. Did the model understand the question? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1896–1906, 2018.

[109] Peter B Mueller. The aging voice. In *Seminars in speech and language*, volume 18, pages 159–169. © 1997 by Thieme Medical Publishers, Inc., 1997.

[110] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[111] Ziad S Nasreddine, Natalie A Phillips, Valérie Bédirian, Simon Charbonneau, Victor Whitehead, Isabelle Collin, Jeffrey L Cummings, and Howard Chertkow. The montreal cognitive assessment, moca: a brief screening tool for mild cognitive impairment. *Journal of the American Geriatrics Society*, 53(4):695–699, 2005.

[112] Ali Bou Nassif, Ismail Shahin, Imtinan B. Attili, Mohammad Azzeh, and Khaled F. Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE Access*, 7:19143–19165, 2019.

[113] Se Jin Oh, Jee Eun Sung, Su Jin Choi, and Jee Hyang Jeong. Clustering and switching patterns in semantic fluency and their relationship to working memory in mild cognitive impairment. *Dementia and neurocognitive disorders*, 18(2):47–61, 2019.

[114] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*, 2019.

[115] Asli Ozdas, Richard G Shiavi, Stephen E Silverman, Marilyn K Silverman, and D Mitchell Wilkes. Investigation of vocal jitter and glottal flow spectrum as possible cues for depression and near-term suicidal risk. *Ieee transactions on Biomedical engineering*, 51(9):1530–1540, 2004.

[116] Serguei VS Pakhomov, Susan E Marino, Sarah Banks, and Charles Bernick. Using automatic speech recognition to assess spoken responses to cognitive tests of semantic verbal fluency. *Speech communication*, 75:14–26, 2015.

[117] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[118] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent

Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[119] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[120] Archana Prabu Kumar, Abirami Omprakash, Maheshkumar Kuppusamy, Maruthy KN, Sathiyasekaran BWC, Vijayaraghavan PV, and Padmavathi Ramaswamy. How does cognitive function measured by the reaction time and critical flicker fusion frequency correlate with the academic performance of students? *BMC medical education*, 20:1–12, 2020.

[121] Lawrence R Rabiner, Ronald W Schafer, et al. Introduction to digital speech processing. *Foundations and Trends® in Signal Processing*, 1(1–2):1–194, 2007.

[122] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 23–29 Jul 2023.

[123] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[124] Jack Rae and Ali Razavi. Do transformers need deep long-range memory? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, July 2020. Association for Computational Linguistics.

[125] Brian Roark, Margaret Mitchell, John-Paul Hosom, Kristy Hollingshead, and Jeffrey

Kaye. Spoken language derived measures for detecting mild cognitive impairment. *IEEE transactions on audio, speech, and language processing*, 19(7):2081–2090, 2011.

[126] Jonathan D Rodgers, Kris Tjaden, Lynda Feenaughty, Bianca Weinstock-Guttman, and Ralph HB Benedict. Influence of cognitive function on speech and articulation rate in multiple sclerosis. *Journal of the International Neuropsychological Society: JINS*, 19(2):173, 2013.

[127] Aurko Roy, Mohammad Taghi Saffar, David Grangier, and Ashish Vaswani. Efficient content-based sparse attention with routing transformers, 2020.

[128] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[129] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv e-prints*, pages arXiv–1710, 2017.

[130] Lahiru Samarakoon and Khe Chai Sim. Factorized hidden layer adaptation for deep neural network based acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12):2241–2250, 2016.

[131] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. In *Interspeech*, 2019.

[132] Benjamin G Schultz, Venkata S Aditya Tarigoppula, Gustavo Noffs, Sandra Rojas, Anneke van der Walt, David B Grayden, and Adam P Vogel. Automatic speech recognition in neurodegenerative disease. *International Journal of Speech Technology*, pages 1–9, 2021.

[133] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[134] Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *International Conference on Machine Learning*, 2012.

[135] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

[136] Kevin J Shih, Rafael Valle, Rohan Badlani, Adrian Lancucki, Wei Ping, and Bryan Catanzaro. Rad-tts: Parallel flow-based tts with robust alignment learning and diverse synthesis. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

[137] Prashanth Gurunath Shivakumar and Panayiotis Georgiou. Transfer learning from adult to children for speech recognition: Evaluation. *Analysis and Recommendations*, 2018.

[138] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

[139] Diana Van Lancker Sidtis, Jihee Choi, Amy G. Alken, and John J. Sidtis. Formulaic language in parkinson's disease and alzheimer's disease: Complementary effects of subcortical and cortical dysfunction. *Journal of speech, language, and hearing research : JSLHR*, 58 5:1493–507, 2015.

[140] Khe Chai Sim, Arun Narayanan, Ananya Misra, Anshuman Tripathi, Golan Pundak, Tara N Sainath, Parisa Haghani, Bo Li, and Michiel Bacchiani. Domain adaptation using factorized hidden layer for robust automatic speech recognition. In *Interspeech*, pages 892–896, 2018.

[141] Alex Smola and Bernhard Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[142] Morgan Sonderegger and Joseph Keshet. Automatic measurement of voice onset time using discriminative structured prediction. *The Journal of the Acoustical Society of America*, 132(6):3965–3979, 2012.

[143] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[144] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[145] Jee Eun Sung, Sujin Choi, Bora Eom, Jae Keun Yoo, and Jee Hyang Jeong. Syntactic complexity as a linguistic marker to differentiate mild cognitive impairment from normal aging. *Journal of Speech, Language, and Hearing Research*, 63(5):1416–1429, 2020.

[146] Vanessa Taler, Brendan T Johns, and Michael N Jones. A large-scale semantic analysis of verbal fluency across the aging spectrum: Data from the canadian longitudinal study on aging. *The Journals of Gerontology: Series B*, 2019.

[147] Vanessa Taler and Natalie A Phillips. Language performance in alzheimer's disease and mild cognitive impairment: a comparative review. *Journal of clinical and experimental neuropsychology*, 30(5):501–556, 2008.

[148] Fengyi Tang, Jun Chen, Hiroko H Dodge, and Jiayu Zhou. The joint effects of acoustic and linguistic markers for early identification of mild cognitive impairment. *Frontiers in digital health*, 3, 2021.

[149] R. Tong, L. Wang, and B. Ma. Transfer learning for children's speech recognition. In *2017 International Conference on Asian Language Processing (IALP)*, pages 36–39, 2017.

[150] László Tóth, Ildikó Hoffmann, Gábor Gosztolya, Veronika Vincze, Gréta Szatlóczki, Zoltán Bánréti, Magdolna Pákáski, and János Kálmán. A speech recognition-based solution for the automatic detection of mild cognitive impairment from spontaneous speech. *Current Alzheimer Research*, 15(2):130–138, 2018.

[151] Angela K Troyer, Morris Moscovitch, and Gordon Winocur. Clustering and switching as two components of verbal fluency: evidence from younger and older healthy adults. *Neuropsychology*, 11(1):138, 1997.

[152] Angela K Troyer, Morris Moscovitch, Gordon Winocur, Larry Leach, and Morris Freedman. Clustering and switching on verbal fluency tests in alzheimer's and parkinson's disease. *Journal of the International Neuropsychological Society*, 4(2):137–143, 1998.

[153] Jane Upton. *Mini-Mental State Examination*, pages 1248–1249. Springer New York, New York, NY, 2013.

[154] Annegreet Van Opbroek, M Arfan Ikram, Meike W Vernooij, and Marleen De Bruijne. Transfer learning improves supervised image segmentation across imaging protocols. *IEEE transactions on medical imaging*, 34(5):1018–1030, 2014.

[155] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[156] Sophia Vinogradov, Jennifer Kirkland, John H Poole, Michael Drexler, Beth A Ober, and Gregory K Shenaut. Both processing speed and semantic memory organization predict verbal fluency in schizophrenia. *Schizophrenia Research*, 59(2):269–275, 2003.

[157] Ravichander Vipperla, Steve Renals, and Joe Frankel. Ageing voices: The effect of changes in voice parameters on asr performance. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010:1–10, 2010.

[158] Changhan Wang, Kyunghyun Cho, and Jiatao Gu. Neural machine translation with byte-level subwords. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9154–9160, 2020.

[159] Tianzi Wang, Jiajun Deng, Mengzhe Geng, Zi Ye, Shoukang Hu, Yi Wang, Mingyu Cui, Zengrui Jin, Xunying Liu, and Helen M. Meng. Conformer based elderly speech recognition system for alzheimer's disease detection. In *Interspeech*, 2022.

[160] Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, 2016.

[161] David L Woods, John M Wyma, Timothy J Herron, and E William Yund. Computerized analysis of verbal fluency: Normative data and the effects of repeated testing, simulated malingering, and traumatic brain injury. *PLOS ONE*, 11(12):e0166439, 2016.

[162] Tet Fei Yap, Julien Epps, Eric HC Choi, and Eliathamby Ambikairajah. Glottal features for speech-based cognitive load classification. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5234–5237. IEEE, 2010.

[163] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[164] Kexin Yu, Katherine Wild, N Maritza Dowling, Jeffrey A Kaye, Lisa C Silbert, and Hiroko H Dodge. Emotional characteristics of socially isolated older adults with MCI using tablet administered NIH toolbox: I-CONECT study. *Alzheimers Dement. (Amst.)*, 14(1):e12372, nov 2022.

[165] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE, 2010.

[166] Yu Zhang, Daniel S. Park, Wei Han, James Qin, Anmol Gulati, Joel Shor, Aren Jansen, Yuanzhong Xu, Yanping Huang, Shibo Wang, Zongwei Zhou, Bo Li, Min Ma,

William Chan, Jiahui Yu, Yongqiang Wang, Liangliang Cao, Khe Chai Sim, Bhuvana Ramabhadran, Tara N. Sainath, Françoise Beaufays, Zhifeng Chen, Quoc V. Le, Chung-Cheng Chiu, Ruoming Pang, and Yonghui Wu. Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1519–1532, 2022.

[167] Zi-Hua Zhang, Sanyuan Chen, Long Zhou, Yu Wu, Shuo Ren, Shujie Liu, Zhuoyuan Yao, Xun Gong, Lirong Dai, Jinyu Li, and Furu Wei. Speechlm: Enhanced speech pre-training with unpaired textual data. *ArXiv*, abs/2209.15329, 2022.

[168] Zi-Hua Zhang, Long Zhou, Junyi Ao, Shujie Liu, Lirong Dai, Jinyu Li, and Furu Wei. Speechut: Bridging speech and text with hidden-unit for encoder-decoder based speech-text pre-training. *ArXiv*, abs/2210.03730, 2022.

[169] Ziqiang Zhang and Junyi Ao. The yitrans speech translation system for iwslt 2022 offline shared task. *ArXiv*, abs/2206.05777, 2022.

[170] Li Zhou, Suzanne V Blackley, Leigh Kowalski, Raymond Doan, Warren W Acker, Adam B Landman, Evgeni Kontrient, David Mack, Marie Meteer, David W Bates, et al. Analysis of errors in dictated clinical documents assisted by speech recognition software and professional transcriptionists. *JAMA network open*, 1(3):e180530–e180530, 2018.