國立臺灣大學電機資訊學院電信工程學研究所
博士論文
Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Doctoral Dissertation

語音處理的解離表徵學習、多任務學習及通用模型
Disentangled Representation Learning, Multi-task Learning and Universal Modeling for Speech Processing

陳奕禎
Yi-Chen Chen

指導教授：李宏毅 教授
Advisor: Hung-yi Lee, Ph.D.

中華民國一百一十一年五月
May 2022

doi:10.6342/NTU202302463

誌謝

在這碩班逕博的五年中，得知於人者太多。

首先感謝實驗室大家長李琳山老師，老師的為人為學、做研究的態度、實驗室文化的建立都讓我們受益很多，見樹又見林的哲學不忘時時刻刻銘記在心。

再來感謝我的貴人與指導教授李宏毅老師，從大學時期便引領我進入機器學習的領域，到做專題、進入語音實驗室這個大家庭，一路上讓我學習到老師的謙遜幽默、對學問的熱忱、問題本質的思考、想法的表達，這些一生受用的事情。

藉由宏毅哥的引薦，我也在 Facebook (現為 Meta) AI team 得到珍貴的實習機會得以拓展視野，從青峰哥、肇君、Gil 的身上學習到了業界語音科學家的思考方式及實務技能。也能在 NVIDIA AI Center 的實習過程中，與宏毅哥和正匡一起討論研究並做出成果，從中得到學界與業界合作的經驗。

當然也很感謝語音實驗室的同學們，每個在實驗室一起研究、討論到深夜的時刻都是讓我成長的機會，在 meeting 中也得到了很多新知。一起聊天、吃飯、打球和出遊的時光也讓這段研究旅途增添了不少樂趣。

謝謝彤恩姐，幫我們處理了實驗室的大小事和跟我們聊天，沒有你實驗室就無法這麼順利的運轉下去。

由衷的感謝我的家人，你們總是支持我做我想做的事情，謝謝你們讓我能夠自由地追逐我的夢想。

最後謝謝我的女友彥伶，謝謝你總是陪伴著我、支持著我、鼓勵著我，和我一起成為更好的人。你的家人也讓我覺得在台北有第二個家。未來一起在人生這條漫長的研究路上繼續進步！

doi:10.6342/NTU202302463

摘要

由於深度學習的成功，愈來愈多基於強大深度模型的語音處理應用影響了我們的生活。然而，這些強大的模型總是針對特定任務而特化，而難以泛化用在其他的任務。因此，對於每個任務，我們都必須分別收集、設計、訓練以及調整所有的資料及模型架構。如果有一種通用模型能夠同時學習並進行多種不同的語音處理任務，那有些任務也許可以透過其他任務習得的技能而更加進步，而且透過不同任務得來的資料也可以被加以利用。但是，這種通用模型需要能從語音訊號中汲取不同種資訊（內容或語者），也能處理不同輸入或輸出模態（語音或文字）的多種任務。因此本論文中，我們針對這兩種問題提出方法。我們藉由對抗式訓練，解離並汲取語音訊號中不同種資訊。我們也提出方法能夠利用多任務訓練使單一模型處理多種任務。

doi:10.6342/NTU202302463

Abstract

Owing to the success of deep learning, more and more applications built on powerful deep models for speech processing tasks have influenced our lives. However, these powerful models are always task-specific and have limited capability to generalize to other tasks. Therefore, all the data and model architectures are collected, designed, trained, and tuned separately for each task. If a universal model can simultaneously learn and perform multiple speech processing tasks, some tasks might be improved with the related abilities learned from other tasks, and more data from various tasks might be leveraged. However, such universal model requires capabilities to extract different kinds of information from speech signals (content or speaker) and handle various tasks with different input and output modalities (speech or text). Hence, in this thesis, we propose approaches to address these two problems. We disentangle and extract different kinds of information from speech signals with adversarial training. Then we propose approaches to handle various tasks using one single model with multi-task learning.

doi:10.6342/NTU202302463

Contents

doi:10.6342/NTU202302463

doi:10.6342/NTU202302463

List of Figures

doi:10.6342/NTU202302463

doi:10.6342/NTU202302463

List of Tables

doi:10.6342/NTU202302463

doi:10.6342/NTU202302463

Chapter 1 Introduction

Deep neural networks (DNNs) have achieved huge success in many speech processing tasks ranging from extracting content (e.g., automatic speech recognition (ASR)) or speaker (e.g., automatic speaker verification (ASV)) information from speech signals to generating speech signals (e.g., text-to-speech (TTS) synthesis). Some tasks both require information extracted from speech signals and generate speech signals, such as voice conversion (VC). For different tasks, model networks are usually designed and tuned separately. Although these task-specific models may perform well on the corresponding tasks, they cannot utilize data from or generalize to other tasks. If a universal model can simultaneously learn and perform multiple speech processing tasks, some tasks might be improved with the related abilities learned from other tasks, and more data from various tasks might be leveraged.

However, to make a universal speech processing model possible, we need to make the model able to extract different kinds of information from speech signals (content or speaker), and handle various tasks with different input and output modalities (speech or text). To do so, we further divide the problem into two sub-problems, and propose approaches to address them respectively:

- To extract content or speaker information according to tasks, we encode and disentangle input speech signals into two parts: content representations and speaker representations.
- To handle different speech processing tasks, we train our model with multi-task learning (MTL) optimization.

doi:10.6342/NTU202302463

1.1 Disentangled Representation Learning of Speech

Audio Word2Vec [1] has been proposed for speech representation learning, in which spoken words (signal segments for words without knowing the underlying word it represents) are transformed into vector representations of fixed dimensionality. These vector representations carry the phonetic structures of the spoken words learned from the signals within the spoken words, and have been shown to be useful in spoken term detection, in which the spoken terms are detected simply based on the phonetic structures.

However, speech representations learned by Audio Word2vec carry not only phonetic information but also speaker information. We have to extract one kind of information and exclude the other from speech signals in speech processing tasks because they may disturb each other. For example:

- The word "good" spoken by a person should be considered close to the word "good" spoken by another person with a very different voice in the task of ASR, while the word "good" spoken by a person should not be considered close to the word "bad" spoken by the same person.
- The word "good" spoken by a person should be considered close to the word "bad" spoken by the same person in the task of ASV, while the word "good" spoken by a person should not be considered close to the word "good" spoken by another person with a very different voice.

In Chapter 3, we study the disentangled representation learning using adversarial training. More specifically, in Section 3.1, we explore a two-stage framework to perform phonetic-and-semantic embedding on spoken words considering the context of the spoken

doi:10.6342/NTU202302463

words, with the initial experiments on spoken document retrieval. In Section 3.2, we explore a pretraining framework AIPNet based on adversarial training for accent-invariant representation learning and further finetune this model by connecting the accent-invariant module with an attention-based encoder-decoder model for multi-accent speech recognition. In Section 3.3, we extend the disentangled speech representations learning from the word level to the utterance level by proposing a new segmental audio word2vec in which unsupervised spoken word boundary segmentation and disentangled representation learning are jointly learned and mutually enhanced.

1.2 Multi-task Learning and Universal Modeling of Speech Processing Tasks

Traditionally, a separate neural network is trained for each task. However, although the task-specific models may perform well on the corresponding tasks, they cannot utilize data from or generalize to other tasks. MTL aims to improve such generalization by training a shared model on various tasks and leveraging domain-specific information contained in the training signals of related tasks. In MTL, a network capable of learning shared representations from various tasks brings several advantages:

- Some layers can be shared by several tasks, so the total memory cost is reduced.
- The performance of related tasks could be improved if they share complementary information or act as regularizers for one another.

In the areas of computer vision (CV) and NLP, general models trained by MTL approaches can be evaluated on benchmarks that include various tasks. However, in speech, there has

doi:10.6342/NTU202302463

been little systematic study of MTL for various speech processing tasks.

In Chapter 4, we study multi-task learning and universal modeling of speech processing tasks. More specifically, in Section 4.1, we use a state-of-the-art SSL pretrained shared model and further finetune it with MTL on various discriminative speech processing tasks, and then evaluate the model on a speech multi-task benchmark. In Section 4.2, we design a universal modularized model for not only discriminative but also generative speech processing tasks. In Section 4.3, we propose an ASR model explored with an efficient gradient-based architecture search on multilingual tasks, where we use a unified model for extracting representations for data of different languages.

1.3 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we introduce the related work of this thesis. In Chapter 3, we study the disentangled representation learning using adversarial training. In Chapter 4, we study multi-task learning and universal modeling of speech processing tasks, followed by the conclusion remarks in Chapter 5.

doi:10.6342/NTU202302463

Chapter 2 Related Work

We introduce the related work of this thesis in this chapter.

2.1 Audio Word Embedding

Word embedding or Word2Vec [2, 3, 4, 5] has been widely used in the area of natural language processing [6, 7, 8, 9, 10, 11, 12], in which text words are transformed into vector representations of fixed dimensionality [13, 14, 15]. This is because these vector representations carry plenty of semantic information learned from the context of the considered words in the text training corpus. Similarly, audio Word2Vec has also been proposed in the area of speech signal processing, in which spoken words (signal segments for words without knowing the underlying word it represents) are transformed into vector representations of fixed dimensionality [16, 17, 1, 18, 19, 20, 21, 22, 23, 24, 25]. These vector representations carry the phonetic structures of the spoken words learned from the signals within the spoken words, and have been shown to be useful in spoken term detection, in which the spoken terms are detected simply based on the phonetic structures. Such Audio Word2Vec representations do not carry semantics, because they are learned from individual spoken words only without considering the context.

Audio Word2Vec was recently extended to Segmental Audio Word2Vec [26], in which an utterance can be automatically segmented into a sequence of spoken words [27, 28, 29, 30] and then transformed into a sequence of vectors of fixed dimensionality by Audio Word2Vec, and the spoken word segmentation and Audio Word2Vec can be jointly trained from an audio corpus. In this way the Audio Word2Vec was upgraded from word-level to

doi:10.6342/NTU202302463

utterance-level. This offers the opportunity for Audio Word2Vec to include semantic information in addition to phonetic structures, since the context among spoken words in utterances bring semantic information.

2.2 Audio Segment Embedding

Audio word embedding requires segmented spoken words as input data. However, word level segmentation is another challenging problem. Another line of research on audio representation learning is audio segment embedding, where the audio segment boundaries do not need to match the word boundaries.

Audio segment representation is still an open problem. It is common to use i-vectors to represent utterances in speaker identification [31]. However, i-vectors are not designed to precisely describe the sequential phonetic structure of audio segments. In previous work, embedding approaches were developed primarily in heuristic ways, rather than learned from data. Graph-based embedding approaches are also used to represent audio segments as fixed-length vectors [20, 32]. Retrieval task efficiency is improved by searching audio content using fixed-length vectors instead of using the original acoustic features [20, 32].

Recently, deep learning has been used to encode acoustic information as vectors [19, 33, 18, 16, 22, 34, 24, 35, 36]. This transformation successfully produces vector spaces in which audio segments with similar phonetic structures are located in close proximity. By training a recurrent neural network (RNN) with an audio segment as the input and the corresponding word as the target, the outputs of the hidden layer at the last few time steps can be taken as the representation of the input segment [33, 17]. However, this approach is supervised and therefore necessitates a large amount of labeled training data.

doi:10.6342/NTU202302463

In [18], the authors train a neural network with side information to obtain embeddings that separate same-word and different-word pairs. Since human annotated data is still required, the scenario is weakly supervised. For non-speech audio, some approaches obtain labeled paired data based on the nature of signals [37], but the approaches have not yet been applied to speech.

In unsupervised pattern discovery, segmentation followed by clustering is typical [38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49] Probabilistic Bayesian models are developed to construct a model which learns segmentation and representation (or clustering) jointly [50, 30, 51]. Although Bayesian models yield successful results, they do not scale well to large speech corpora; as such the embedded segmental K-means model has been proposed as an approximation of the Bayesian model [29] – this however does not take advantage of deep learning. Another approach to learn segmentation is using an autoencoder with a sample-based algorithm [52]. First an LSTM is used to model a proposal distribution over sequences of segment boundaries for each utterance. Then m sequences of boundaries from the distribution are sampled to split the utterance into words and the reconstruction loss for each sequence is obtained with an autoencoder. The losses and the distribution are used to compute an importance weight for each sample and breakpoint. A breakpoint is more likely if it appeared in samples with low reconstruction loss. In comparison, Segmental Sequence-to-sequence Autoencoder [26] is a "whole network model" using reinforcement learning, which scales well and can be trained in an end-to-end fashion.

doi:10.6342/NTU202302463

2.3 Modern Self-supervised Representation Learning of Speech and SUPERB Benchmark

Many recently proposed self-supervised learning (SSL) approaches allow us to embed speech signals into representations where the underlying structures in speech signals are leveraging and can be generally useful for downstream tasks.

Modern SSL approaches can be categorized into three types [53]: generative approaches, discriminative approaches, and multi-objective approaches.

- Generative approaches: APC [54], VQ-APC [55], Mockingjay [56], TERA [57], NPC [58], DeCoAR 2.0 [59] use language modeling pretraining approaches on a sequence of acoustic features, generating future or masked frames conditioned on past or unmasked frames.

- Disriminative approaches: CPC [60, 61], wav2vec [62], vq-wav2vec [63], and wav2vec 2.0 [64] adopt contrastive learning to discriminate the correlated positive samples from negative samples with contrastive InfoNCE loss, which maximizes the mutual information between raw data and representations. Instead of using contrastive loss, HuBERT [65] uses a more direct predictive loss by predicting masked tokens via off-line clustering on representations.

- Multi-objective approaches: PASE [66] and PASE+ [67] combine several pretraining objectives together, such as waveform generation, prosody features regression, contrastive InfoMax objectives, and so on. Multiple perturbations are also applied to input speech like reverberation and additive noise.

doi:10.6342/NTU202302463

To fairly evaluate the generalizability of SSL approaches without further heavy downstream task-specific finetuning, the SUPERB benchmark [53] is proposed. The SUPERB benchmark measures the performance of a shared model across a wide range of speech processing tasks without heavy finetuning. Ten tasks are included to investigate four aspects of speech: content, speaker, semantics, and paralinguistics. To evaluate a general model trained with SSL, the pretrained model parameters are frozen, and the fixed representations are extracted and fed into each task-specific prediction head (small downstream model) for training. During the evaluation, the pretrained shared model and trained prediction heads are used on all tasks. In the above scenario, some self-supervised models show outstanding performances on all the ten tasks in SUPERB.

2.4 Adversarial Training

Speech representation learning always embed many kinds of information together in representations, such as phonetic structures or speaker characteristics. One of our goal in this thesis is to disentangle different features in speech signals. If we can properly define the "domain" of inputs, adversarial training allows us to disentangle domain-invariant features from raw inputs.

Gradient reversal training [68] was initially proposed to learn domain-invariant features, which is separated from domain-related features. To be more specific, a gradient reversal layer is placed between the feature extractor and the domain classifier, which leaves the input unchanged during forward propagation and reverses the gradient by multiplying it by a negative scalar during the backpropagation. In this way, the feature distributions over two domains are made similar (as indistinguishable as possible for the domain classifier),

doi:10.6342/NTU202302463

thus resulting in the domain-invariant features.

In Generative adversarial nets (GAN) [69], a generative module is used to generate domain-invariant features from inputs, and a discriminative module is used to discriminate whether the generated features belong to a specific domain. The generative and discriminative modules are iteratively updated: in each iteration, the discriminator is trained to be more discriminative about the domain, while the generator is trained to "fool" the discriminator such that the discriminator cannot discriminate the domain of generated features. Therefore, after the iterative training, the generator is able to generate domain-invariant features.

In our case, we can define the "speaker identities" as "domains". In this way, the speaker-invariant phonetic information can be separated from the speaker characteristics. For a realistic example, adversarial training can also be used in accented speech recognition, which is described in more details in Section 2.5.

2.5 Accented Speech Recognition

Accents are defined as variations in pronunciation within a language and are often peculiar to geographical regions, individuals, social groups, etc. As one of the major sources in speech variability, accents have posed a grand technical challenge to the robustness of ASR systems. Due to the acoustic discrepancy among accents, an ASR system that is trained on the speech data of one accent (e.g., native) often fails to recognize speech of another unseen accent (e.g., non-native). In this thesis, we focus on learning accent-invariant representations through adversarial training, aiming to build a universal ASR system that is generalizable across accents.

doi:10.6342/NTU202302463

There is an extensive literature on multi-accent modeling for speech recognition [70] [71]. The existing approaches can be categorized into two classes in general: accent-independent and accent-dependent. Accent-independent modeling focuses on building a universal model that generalizes well across accents. One popular baseline is to train a model on all the data of different accents [72] [73] [74]. Elfeky *et al.* have attempted to build a unified multi-accent recognizer from a pre-defined unified set of CD states by learning from the ensemble of accent-specific models [72]. Yang *et al.* have proposed to jointly model ASR acoustic model and accent identification classifier through multi-task learning [75]. Accent-dependent approaches either take accent-related information, such as accent embedding or i-vectors, as an complementary input in the modeling or adapt a unified model on accent-specific data [76] [77] [78] [79] [80]. Accent-dependent models usually outperform the unified ones with known accent labels or on an accent-specific dataset, while accent-independent models demonstrate better generalizability on average when accent labels are unavailable during testing.

Generative adversarial nets (GAN) [69] or gradient reverse technique [68] has gained popularity in learning a representation that is invariant to domains or conditions [81] [82] [83] [84]. Serdyuk *et al.* have applied adversarial training to generate noise-invariant representations for speech recognition [82]. Gradient reversal training has recently been used for learning domain-invariant features to alleviate the mismatch between accents during training [83]. Bousmalis *et al.* have proposed to a GAN-based pixel-level transformation from one domain to the other and have shown great improvement over state-of-the-art on a number of domain adaptation tasks [84].

doi:10.6342/NTU202302463

2.6 Multi-task Learning of Speech Processing Tasks

The goal of multi-task learning (MTL) is to leverage useful information contained in multiple related tasks to help improve the generalization performance of all the tasks.

It has been shown that a single deep learning model can jointly learn a number of large-scale tasks from multiple domains [85]. [86] proposes a model to solve ten tasks in natural language processing (NLP). The core idea of the T5 model [87], a unified framework for a variety of text-based language problems, is to treat every text processing problem as a ”text-to-text” problem, i.e., taking text as input and producing new text as output.

In the speech domain, some previous works train a model to solve two tasks or use an auxiliary task to improve the primary task’s performance. [88] studies SE and ASR. [89, 90] study the duality of ASR and TTS. Some papers [91, 92] use SC to improve ASR. [93] studies ASR and SC. [94, 95] shows the effectiveness of SC to help TTS. [96] shows the performance of VC can be improved with text supervision. [97] connects SC and VC. [98] jointly trains TTS and VC. [99] uses TTS and SC to improve VC.

However, in these works, models are designed for some specific speech processing tasks and not applicable for more speech processing tasks. Moreover, the rapid rate of progress and diversity of techniques also make it difficult to compare different algorithms, tease apart the effects of new contributions, and understand the effectiveness of learning multiple speech processing tasks with one model.

Multilingual speech recognition can be treated as a multi-task learning scenario as well, where the recognition of each language is regarded as one task. Many languages have little data available. However, different languages may share some common knowledge such as phonetic or semantics patterns. It has recently been shown that multilingual

doi:10.6342/NTU202302463

ASR [79, 100, 101, 102, 103, 104, 105, 106, 94] can improve ASR performance on many low-resource languages. In the above previous works, the initial parameters or shared encoder learned from many source languages are used to build a better acoustic model for the target language.

2.7 Architecture Search of Deep Neural Networks

A lot of empirical evidence has shown that network architecture matters significantly in fields like image classification (from AlexNet [107] to ResNet [108]) or natural language processing (NLP) (Transformer [109]). Despite the success of these DNNs, the architecture is still hard to design. The popular architectures were usually invented and tuned by experts through a long process of trial and error.

For example, convolutional neural networks (CNN) [107] have been proved to be more effective in image recognition tasks than DNNs with fully-connected layers. CNNs were inspired by biological processes where the connectivity pattern between neurons resembles the organization of the animal visual cortex [110]. However, the birth of such successful model architecture always relies on human wisdom and a flash of insight. Besides, many hyperparameters in CNNs still have to be carefully tuned, such as channel numbers, kernel sizes, strides, padding, pooling and activation functions for each layer. Therefore, it is highly appealing to have an effective algorithm to discover and design architectures of DNNs automatically.

Many researchers have focused on automatic neural architecture search (NAS) algorithms, aiming to optimize not only parameter weights of a fixed-topology neural network architecture, but also the design of architecture itself. Some approaches [111, 112] use

doi:10.6342/NTU202302463

reinforcement learning (RL) to search for building blocks used in CNN. Some other approaches [113] use evolutionary algorithms to find building blocks through mutation and tournament selection. Some recent works also incorporate NAS into their approaches to speech recognition [114] or keyword spotting [115, 116]. Although these approaches have achieved convincing results on many benchmark datasets, a huge amount of computational resources are needed to perform exploration in a search space.

Differentiable ARchiTecture Search (DARTS) [117] uses a gradient-based method for efficient architecture search. Instead of searching over discrete architecture candidates, with a continuous relaxation of architecture representation, the architecture can be jointly optimized with parameter weights directly by gradient descent. On many benchmark datasets of image classification, more recent approaches [118, 119, 120] based on DARTS have discovered model architectures that achieved state-of-the-art results with similar parameter size to other models.

doi:10.6342/NTU202302463

Chapter 3 Disentangled Representation Learning of Speech Signals

In this chapter, we propose approaches based on adversarial training to disentangle phonetic and speaker information from speech [81, 121, 92].

3.1 Phonetic-and-Semantic Embedding of Spoken Words with Applications in Spoken Content Retrieval

Audio Word2vec [1] encode both phonetic structures and speaker characteristics of spoken words but not semantics since they are learned from individual spoken words only without considering the context. In this section, we propose an adversarial training approach to disentangle phonetic structures and speaker characteristics from spoken words. Then we further apply language modeling as in Word2vec [2] on the phonetic representations of spoken words to encode semantics information.

In principle, the semantics and phonetic structures in words inevitably disturb each other. For example, the words "brother" and "sister" are close in semantics but very different in phonetic structure, while the words "brother" and "bother" are close in phonetic structure but very different in semantics. This implies the goal of embedding both phonetic structures and semantics for spoken words is naturally very challenging. Text words can be trained and embedded as vectors carrying plenty of semantics because the phonetic structures are not considered at all. On the other hand, because spoken words are just a different version of representations for text words, it is also natural to believe they do carry some se-

doi:10.6342/NTU202302463

mantic information, except disturbed by phonetic structures plus some other acoustic factors such as speaker characteristics and background noise [122, 123, 124, 125, 126, 127]. So the goal of embedding spoken words to carry both phonetic structures and semantics is possible, although definitely hard.

But a nice feature of such embeddings is that they may include both phonetic structures and semantics [128, 37]. A direct application for such phonetic-and-semantic embedding of spoken words is spoken document retrieval [129, 130, 131, 132, 133]. This task is slightly different from spoken term detection, in the latter case spoken terms are simply detected based on the phonetic structures. Here the goal of the task is to retrieve all spoken documents (sets of consecutive utterances) relevant to the spoken query, which may or may not include the query. For example, for the spoken query of "President Donald Trump", not only those documents including the spoken query should be retrieved based on the phonetic structures, but those documents including semantically related words such as "White House" and "trade policy", but not necessarily "President Donald Trump", should also be retrieved. This is usually referred to as "semantic retrieval", which can be achieved by the phonetic-and-semantic embedding discussed here.

This section proposes a two-stage framework of phonetic-and-semantic embedding for spoken words. Stage 1 performs phonetic embedding but with speaker characteristics disentangled using separate phonetic and speaker encoders and a speaker discriminator. Stage 2 then performs semantic embedding in addition. We further propose to evaluate the phonetic-and-semantic nature of the audio embeddings obtained in Stage 2 by parallelizing with text embeddings [134, 135]. Encouraging results including those for an application task of spoken document retrieval were obtained in the initial experiments

doi:10.6342/NTU202302463

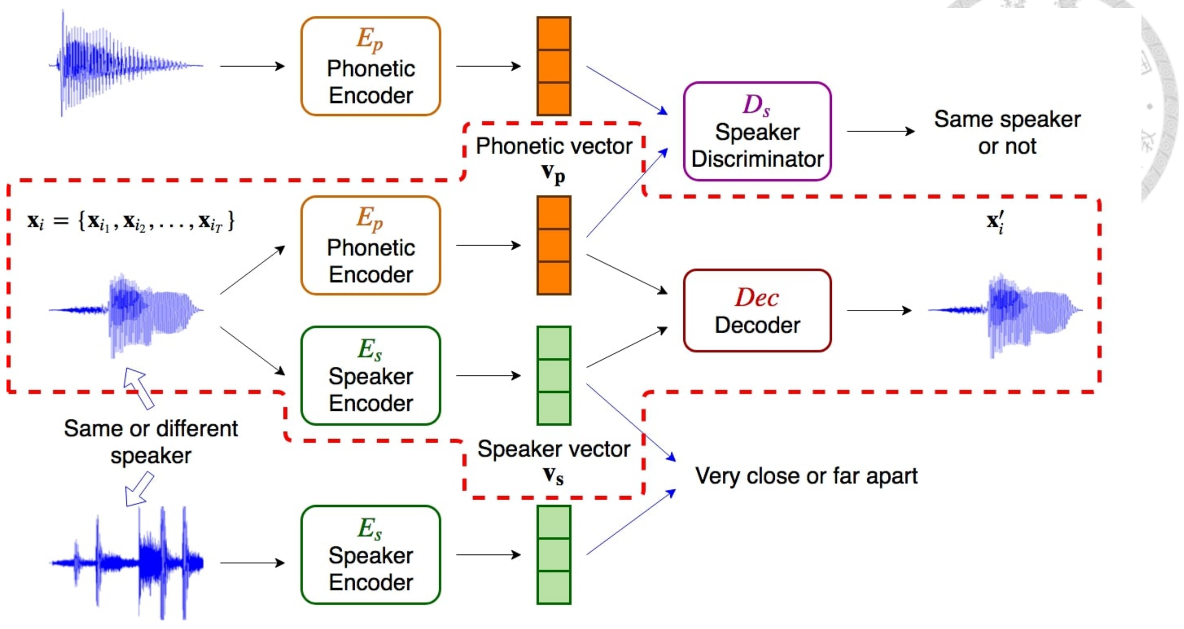


Figure 3.1: Phonetic embedding with speaker characteristics disentangled.

3.1.1 Proposed Approach

The proposed framework of phonetic-and-semantic embedding of spoken words consists of two stages:

Stage 1 - Phonetic embedding with speaker characteristics disentangled.

Stage 2 - Semantic embedding over phonetic embeddings obtained in Stage 1.

In addition, we propose an approach for parallelizing the audio and text embeddings to be used for evaluating the phonetic and semantic information carried by the audio embeddings. These are described in the following subsections respectively.

Stage 1 - Phonetic Embedding with Speaker Characteristics Disentangled

A text word with a given phonetic structure corresponds to infinite number of audio signals with varying acoustic factors such as speaker characteristics, microphone characteristics, background noise, etc. All the latter acoustic factors are jointly referred to as speaker char-

doi:10.6342/NTU202302463

acteristics here for simplicity, which obviously disturbs the goal of phonetic-and-semantic embedding. So Stage 1 is to obtain phonetic embeddings only with speaker characteristics disentangled.

Also, because the training of phonetic-and-semantic embedding is challenging, in the initial effort we slightly simplify the task by assuming all training utterances have been properly segmented into spoken words. Because there exist many approaches for segmenting utterances automatically [26], and automatic segmentation plus phonetic embedding of spoken words has been successfully trained and reported before [26], such an assumption is reasonable here.

We denote the audio corpus as $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{M}$, which consists of M spoken words, each represented as $\mathbf{x}_i = (\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, ..., \mathbf{x}_{i_T})$, where $\mathbf{x}_{i_t}$ is the acoustic feature vector for the t^{th} frame and T is the total number of frames in the spoken word. The goal of Stage 1 is to disentangle the phonetic structure and speaker characteristics in acoustic features, and extract a vector representation for the phonetic structure only.

Autoencoder As shown in the middle of Figure 3.1, a sequence of acoustic features $\mathbf{x}_i = (\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, ..., \mathbf{x}_{i_T})$ is entered to a phonetic encoder E_p and a speaker encoder E_s to obtain a phonetic vector $\mathbf{v_p}$ in orange and a speaker vector $\mathbf{v_s}$ in green. Then the phonetic and speaker vectors $\mathbf{v_p}$, $\mathbf{v_s}$ are used by the decoder Dec to reconstruct the acoustic features $\mathbf{x}'$. This phonetic vector $\mathbf{v_p}$ will be used in the next stage as the phonetic embedding. The two encoders E_p, E_s and the decoder Dec are jointly learned by minimizing the reconstruction loss below:

$$L_r = \sum_i \|\mathbf{x}_i - Dec(E_p(\mathbf{x}_i), E_s(\mathbf{x}_i))\|_2^2. \tag{3.1}$$

doi:10.6342/NTU202302463

It will be clear below how to make E_p and E_s separately encode the phonetic structure and speaker characteristics.

Training Criteria for Speaker Encoder The speaker encoder training requires speaker information for the spoken words. Assume the spoken word $\mathbf{x}_i$ is uttered by speaker s_i. When the speaker information is not available, we can simply assume that the spoken words in the same utterance are produced by the same speaker. As shown in the lower part of Figure 3.1, E_s is learned to minimize the following loss:

$$\begin{aligned} L_s = &\sum_{s_i = s_j} \|\mathbf{v_s}_i - \mathbf{v_s}_j\|_2^2 \\ &+ \sum_{s_i \neq s_j} \max(\lambda - \|\mathbf{v_s}_i - \mathbf{v_s}_j\|_2^2, 0). \end{aligned} \tag{3.2}$$

In other words, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are uttered by the same speaker ($s_i = s_j$), we want their speaker embeddings $\mathbf{v_s}_i$ and $\mathbf{v_s}_j$ to be as close as possible. But if $s_i \neq s_j$, we want the distance between $\mathbf{v_s}_i$ and $\mathbf{v_s}_j$ larger than a threshold λ.

Training Criteria for Phonetic Encoder As shown in the upper right corner of Figure 3.1, a speaker discriminator D_s takes two phonetic vectors $\mathbf{v}_{\mathbf{p}_i}$ and $\mathbf{v}_{\mathbf{p}_j}$ as input and tries to tell if the two vectors come from the same speaker. The learning target of the phonetic encoder E_p is to "fool" this speaker discriminator D_s, keeping it from discriminating the speaker identity correctly. In this way, only the phonetic structure information is learned in the phonetic vector $\mathbf{v_p}$, while only the speaker characteristics is encoded in the speaker vector $\mathbf{v_s}$. The speaker discriminator D_s learns to maximize L_d in (3.3), while the phonetic encoder E_p learns to minimize L_d,

$$L_d = \sum_{s_i = s_j} D_s(\mathbf{v}_{\mathbf{p}_i}, \mathbf{v}_{\mathbf{p}_j}) - \sum_{s_i \neq s_j} D_s(\mathbf{v}_{\mathbf{p}_i}, \mathbf{v}_{\mathbf{p}_j}). \tag{3.3}$$

doi:10.6342/NTU202302463

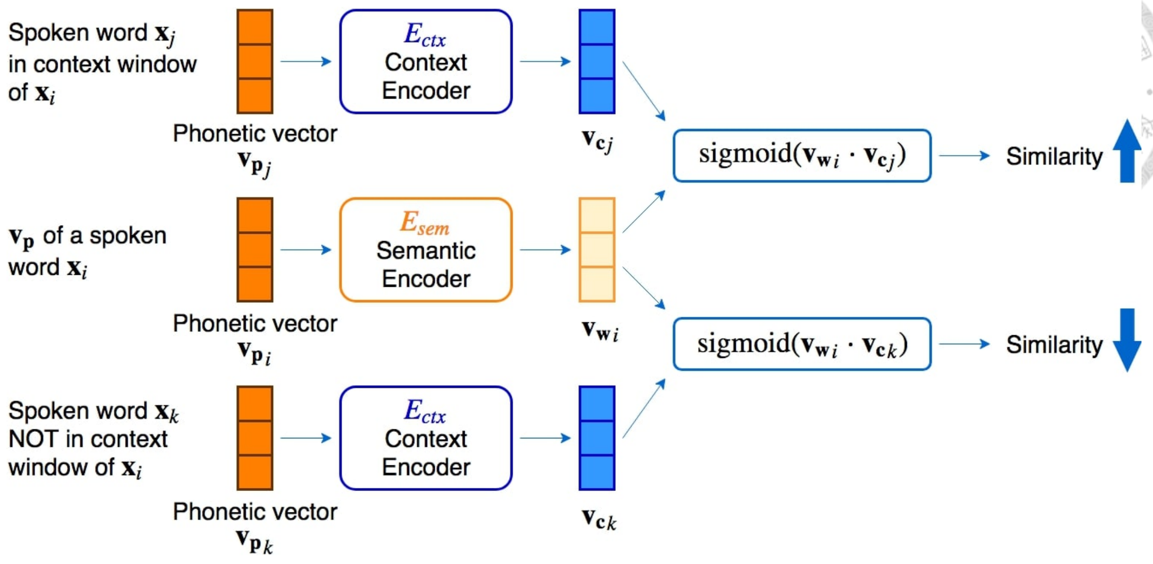

Spoken word $\mathbf{x}_j$ in context window of $\mathbf{x}_i$
Phonetic vector $\mathbf{v}_{\mathbf{p}j}$
E_{ctx} Context Encoder
$\mathbf{v}_{\mathbf{c}j}$
$\mathrm{sigmoid}(\mathbf{v}_{\mathbf{w}i} \cdot \mathbf{v}_{\mathbf{c}j})$
Similarity
$\mathbf{v}_\mathbf{p}$ of a spoken word $\mathbf{x}_i$
Phonetic vector $\mathbf{v}_{\mathbf{p}i}$
E_{sem} Semantic Encoder
$\mathbf{v}_{\mathbf{w}i}$
$\mathrm{sigmoid}(\mathbf{v}_{\mathbf{w}i} \cdot \mathbf{v}_{\mathbf{c}k})$
Similarity
Spoken word $\mathbf{x}_k$ NOT in context window of $\mathbf{x}_i$
Phonetic vector $\mathbf{v}_{\mathbf{p}k}$
E_{ctx} Context Encoder
$\mathbf{v}_{\mathbf{c}k}$

Figure 3.2: Semantic embedding over phonetic embeddings obtained in Stage 1.

where $D_s(\cdot, \cdot)$ is a real number.

Overall Optimization of Stage 1 The optimization procedure of Stage 1 consists of four parts: (1) training E_p, E_s and Dec by minimizing L_r, (2) training E_s by minimizing L_s, (3) training E_p by minimizing L_d, and (4) training D_s by maximizing L_d. Parts (1)(2)(3) are jointly trained together, while iteratively trained with part (4) [136].

Stage 2 - Semantic Embedding over Phonetic Embeddings Obtained in Stage 1

As shown in Figure 3.2, similar to the Word2Vec skip-gram model [2], we use two encoders: semantic encoder E_{sem} and context encoder E_{ctx} to embed the semantics over phonetic embeddings $\mathbf{v_p}$ obtained in Stage 1. On the one hand, given a spoken word $\mathbf{x_i}$, we feed its phonetic vector $\mathbf{v}_{\mathbf{p}_i}$ obtained from Stage 1 into E_{sem} as in the middle of Figure 3.2, producing the semantic embedding (in yellow) of the spoken word $\mathbf{v}_{\mathbf{w}i} = E_{\text{sem}}(\mathbf{v}_{\mathbf{p}_i})$. On the other hand, given the context window size c, which is a hyperparameter, if a spoken word $\mathbf{x_j}$ is in the context window of $\mathbf{x_i}$, then its phonetic vector $\mathbf{v}_{\mathbf{p}_j}$ is a context vector

doi:10.6342/NTU202302463

of $\mathbf{v}_{\mathbf{p}_i}$. For each context vector $\mathbf{v}_{\mathbf{p}_j}$ of $\mathbf{v}_{\mathbf{p}_i}$, we feed it into the context encoder E_{ctx} in the upper part of Figure 3.2, and the output is the context embedding $\mathbf{v}_{\mathbf{c}j} = E_{\text{ctx}}(\mathbf{v}_{\mathbf{p}_j})$.

Given a pair of phonetic vectors $(\mathbf{v}_{\mathbf{p}_i}, \mathbf{v}_{\mathbf{p}_j})$, the training criteria for E_{sem} and E_{ctx} is to maximize the similarity between $\mathbf{v}_{\mathbf{w}i}$ and $\mathbf{v}_{\mathbf{c}j}$ if $\mathbf{v}_{\mathbf{p}_i}$ and $\mathbf{v}_{\mathbf{p}_j}$ are contextual, while minimizing the similarity otherwise. The basic idea is parallel to that of text Word2Vec. Two different spoken words having similar context should have similar semantics. Thus if two different phonetic embeddings corresponding to two different spoken words have very similar context, they should be close to each other after projected by the semantic encoder E_{sem}. The semantic and context encoders E_{sem} and E_{ctx} learn to minimize the semantic loss L_{sem} as follows:

$$\begin{aligned} L_{\text{sem}} = &\sum_{(\mathbf{x_i},\mathbf{x_j}) \text{ in context window}} -\log(\text{sigmoid}(\mathbf{v}_{\mathbf{w}i} \cdot \mathbf{v}_{\mathbf{c}j})) \\ + &\sum_{(\mathbf{x_i},\mathbf{x_k}) \text{ not in context window}} -\log(\text{sigmoid}(-\mathbf{v}_{\mathbf{w}i} \cdot \mathbf{v}_{\mathbf{c}k})). \end{aligned} \tag{3.4}$$

The sigmoid of dot product of $\mathbf{v_w}$ and $\mathbf{v_c}$ is used to evaluate the similarity. With (3.4), if $\mathbf{x_i}$ and $\mathbf{x_j}$ are in the same context window, we want $\mathbf{v}_{\mathbf{w}i}$ and $\mathbf{v}_{\mathbf{c}j}$ to be as similar as possible. We also use the negative sampling technique, in which only some pairs $(\mathbf{x_i}, \mathbf{x_k})$ are randomly sampled as negative examples instead of enumerating all possible negative pairs.

Parallelizing Audio and Text Embeddings for Evaluation Purposes

In this subsection we further propose an approach of parallelizing a set of audio embeddings (for spoken words) with a set of text embeddings (for text words) which will be useful in evaluating the phonetic and semantic information carried by these embeddings.

Assume we have the audio embeddings for a set of spoken words $\mathbf{P_W}$ $=\{\mathbf{p}_{\mathbf{w}1}, ..., \mathbf{p}_{\mathbf{w}i}, ..., \mathbf{p}_{\mathbf{w}M}\}$,

doi:10.6342/NTU202302463

where $\mathbf{p}_{\mathbf{w}_i}$ is the embedding obtained for a spoken word $\mathbf{x}_i$ and M is the total number of distinct spoken words in the audio corpus. On the other hand, assume we have the text embeddings $\mathbf{Q}_{\mathbf{W}} = \{\mathbf{q}_{\mathbf{w}1}, ..., \mathbf{q}_{\mathbf{w}_j}, ..., \mathbf{q}_{\mathbf{w}M}\}$, where $\mathbf{q}_{\mathbf{w}_j}$ is the embedding of the j-th text word for the M distinct text words. Although the distributions of $\mathbf{P}_{\mathbf{W}}$ and $\mathbf{Q}_{\mathbf{W}}$ in their respective spaces are not parallel, that is, a specific dimension in the space for $\mathbf{p}_{\mathbf{w}}$ does not necessarily correspond to a specific dimension in the space for $\mathbf{q}_{\mathbf{w}}$, there should exist some consistent relationship between the two distributions. For example, the relationships among the words {France, Paris, Germany} learned from context should be consistent in some way, regardless of whether they are in text or spoken form. So we try to learn a mapping relation between the two spaces. It will be clear below such a mapping relation can be used to evaluate the phonetic and semantic information carried by the audio embeddings.

Mini-Batch Cycle Iterative Closest Point (MBC-ICP) [135] previously proposed as described below is used here. Given two sets of embeddings as mentioned above, $\mathbf{P}_{\mathbf{W}}$ and $\mathbf{Q}_{\mathbf{W}}$, they are first projected to their respective top K principal components by PCA. Let the projected sets of vectors of $\mathbf{P}_{\mathbf{W}}$ and $\mathbf{Q}_{\mathbf{W}}$ be $\mathbf{A}$ and $\mathbf{B}$ respectively. If $\mathbf{P}_{\mathbf{W}}$ can be mapped to the space of $\mathbf{Q}_{\mathbf{W}}$ by an affine transformation, the distributions of $\mathbf{A}$ and $\mathbf{B}$ would be similar after PCA [135].

Then a pair of transformation matrices, $\mathbf{T}_{\mathbf{ab}}$ and $\mathbf{T}_{\mathbf{ba}}$, is learned, where $\mathbf{T}_{\mathbf{ab}}$ transforms a vector $\mathbf{a}$ in $\mathbf{A}$ to the space of $\mathbf{B}$, that is, $\tilde{\mathbf{b}} = \mathbf{T}_{\mathbf{ab}}\mathbf{a}$, while $\mathbf{T}_{\mathbf{ba}}$ maps a vector $\mathbf{b}$ in $\mathbf{B}$ to the space of $\mathbf{A}$. $\mathbf{T}_{\mathbf{ab}}$ and $\mathbf{T}_{\mathbf{ba}}$ are learned iteratively by the algorithm proposed previously [135].

In our evaluation as mentioned below, labeled pairs of the audio and text embeddings of each word is available, that is, we know $\mathbf{a}_{\mathbf{i}}$ and $\mathbf{b}_{\mathbf{i}}$ for each word $\mathbf{w}_{\mathbf{i}}$. So we can train the

doi:10.6342/NTU202302463

transformation matrices $\mathbf{T_{ab}}$ and $\mathbf{T_{ba}}$ using the gradient descent method to minimize the following objective function:

$$\begin{aligned} L_{trans} = &\sum_i \|\mathbf{b_i} - \mathbf{T_{ab}}\mathbf{a_i}\|_2^2 + \sum_j \|\mathbf{a_j} - \mathbf{T_{ba}}\mathbf{b_j}\|_2^2 \\ &+ \lambda' \sum_i \|\mathbf{a_i} - \mathbf{T_{ba}}\mathbf{T_{ab}}\mathbf{a_i}\|_2^2 \\ &+ \lambda' \sum_j \|\mathbf{b_j} - \mathbf{T_{ab}}\mathbf{T_{ba}}\mathbf{b_j}\|_2^2. \end{aligned} \tag{3.5}$$

where the last two terms in (3.5) are cycle-constraints to ensure that both $\mathbf{a_i}$ and $\mathbf{b_j}$ are almost unchanged after transformed to the other space and back. In this way we say the two sets of embeddings are parallelized.

3.1.2 Experimental Setup

Dataset

We used LibriSpeech [137] as the audio corpus in the experiments, which is a corpus of read speech in English derived from audiobooks. This corpus contains 1000 hours of speech sampled at 16 kHz uttered by 2484 speakers. We used the "clean" and "others" sets with a total of 960 hours, and extracted 39-dim MFCCs as the acoustic features.

Model Implementation

In Stage 1, The phonetic encoder E_p, speaker encoder E_s and decoder Dec were all 2-layer GRUs with hidden layer size 128, 128 and 256, respectively. The speaker discriminator D_s is a fully-connected feedforward network with 2 hidden layers with size 128. The value of λ we used in L_s in (3.2) was set to 0.01.

In Stage 2, the two encoders E_{sem} and E_{ctx} were both 2-hidden-layer fully-connected

doi:10.6342/NTU202302463

feedforward networks with size 256. The size of embedding vectors was set to be 128. The context window size was 5, and the negative sampling number was 5.

For parallelizing the text and audio embeddings, we projected the embeddings to the top 100 principle components, so the affine transformation matrices were 100×100. The mini-batch size was 200, and λ' in (3.5) was set to 0.5.

3.1.3 Experimental Results

Evaluation by Parallelizing Audio and Text Embeddings

Each text word corresponds to many audio realizations in spoken form. So we first took the average of the audio embeddings for all those realizations to be the audio embedding for the spoken word considered. In this way, each word has a unique representation in either audio or text form.

We applied three different versions of audio embedding (AUD) on the top 1000, 3000 and 5000 words with the highest frequencies in LibriSpeech: (i) phonetic embedding only obtained in Stage 1 (AUD-ph); (ii) phonetic-and-semantic embedding obtained by Stages 1 and 2, except the speaker characteristics not disentangled (AUD-(ph$^-$+se)), or L_s, L_d in (3.2), (3.3) not considered; (iii) complete phonetic-and-semantic embedding as proposed in this section including Stages 1 and 2 (AUD-(ph+se)). So this is for ablation study.

On the other hand, we also obtained three different types of text embedding (TXT) on the same set of top 1000, 3000 and 5000 words. Type (a) Phonetic Text embedding (TXT-ph) considered precise phonetic structure but not context or semantics at all. This was achieved by a well-trained sequence-to-sequence autoencoder encoding the precise phoneme sequence of a word into a latent embedding. Type (b) Semantic Text embedding

doi:10.6342/NTU202302463

considered only context or semantics but not phonetic structure at all, and was obtained by a standard skip-gram model using one-hot representations as the input (TXT-(se,1h)). Type (c) Semantic and Phonetic Text embedding (TXT-(se,ph)) considered context or semantics as well as the precise phonetic structure, obtained by a standard skip-gram model but using the Type (a) Phonetic Text embedding (TXT-ph) as the input. So these three types of text embeddings provided the reference embeddings obtained from text and/or phoneme sequences, not disturbed by audio signals at all.

Now we can perform the transformation from the above three versions of audio embeddings (AUD-ph, AUD-(ph$^-$+se), AUD-(ph+se)) to the above three types of text embeddings (TXT-ph, TXT-(se,1h), TXT-(se,ph)) by parallelizing the embeddings. The evaluation metric used for this parallelizing test is the top-k nearest accuracy. If the audio embedding representation $\mathbf{a_i}$ of a word $\mathbf{w_i}$ is transformed to the text embedding $\mathbf{b_j}$ by $\mathbf{T_{ab}}$, and $\mathbf{b_j}$ is among the top-k nearest neighbors of the text embedding representation $\mathbf{b_i}$ of the same word, this transformation for word $\mathbf{w_i}$ is top-k-accurate. The top-k nearest accuracy is then the percentage of the words considered which are top-k-accurate.

The results of top-k nearest accuracies for k=1 and 10 are respectively listed in Tables 3.1 and 3.2, each for 1000, 3000 and 5000 pairs of spoken and text words.

First look at the top part of Table 3.1 for top-1 nearest accuracies for 1000 pairs of audio and text embeddings. Since column (a) (TXT-ph) considered precise phonetic structures but not semantics at all, the relatively high accuracies in column (a) for all three versions of audio embedding (i)(ii)(iii) implied the three versions of audio embedding were all rich of phonetic information. But when the semantics were embedded in (ii)(iii) (AUD-(ph$^-$+se), AUD-(ph+se)), the phonetic structures were inevitably disturbed (0.519, 0.598

doi:10.6342/NTU202302463

Table 3.1: Top-1 nearest accuracies when parallelizing the different versions of audio and text embeddings for different numbers of pairs of spoken and text words.

		(a)TXT-ph	(b)TXT-(se,1h)	(c)TXT-(se,ph)
1000 pairs	(i)AUD-ph	**0.637**	0.124	0.550
	(ii)AUD-(ph$^-$+se)	0.519	0.322	**0.750**
	(iii)AUD-(ph+se)	0.598	0.339	**0.800**
3000 pairs	(i)AUD-ph	**0.465**	0.028	0.279
	(ii)AUD-(ph$^-$+se)	**0.330**	0.032	0.254
	(iii)AUD-(ph+se)	**0.395**	0.033	0.313
5000 pairs	(i)AUD-ph	**0.362**	0.012	0.190
	(ii)AUD-(ph$^-$+se)	**0.263**	0.022	0.173
	(iii)AUD-(ph+se)	**0.315**	0.023	0.212

vs 0.637). On the other hand, column (b) (TXT-(se,1h)) considered only semantics but not phonetic structure at all, the relatively lower accuracies implied the three versions of audio embedding did bring some good extent of semantics, except (i) AUD-ph, but obviously weaker than the phonetic information in column (a). Also, the Stage 2 training in rows (ii)(iii) (AUD-(ph$^-$+se), AUD-(ph+se)) gave higher accuracies than row (i) (AUD-ph) (0.339, 0.332 vs 0.124 in column (b)), which implied the Stage 2 training was successful. However, column (c) (TXT-(se,ph)) is for the text embedding considering both the semantic and phonetic information, so the two versions of phonetic-and-semantic audio embedding for rows (ii)(iii) had very close distributions (0.750, 0.800 in column (c)), or carried good extent of both semantics and phonetic structure. The above are made clearer

doi:10.6342/NTU202302463

Table 3.2: Top-10 nearest accuracies when parallelizing the different versions of audio and text embeddings for different numbers of pairs of spoken and text words.

		(a)TXT-ph	(b)TXT-(se,1h)	(c)TXT-(se,ph)
1000 pairs	(i)AUD-ph	**0.954**	0.355	0.898
	(ii)AUD-(ph^{-}+se)	0.897	0.653	**0.986**
	(iii)AUD-(ph+se)	0.945	0.742	**0.994**
3000 pairs	(i)AUD-ph	**0.854**	0.120	0.654
	(ii)AUD-(ph^{-}+se)	**0.758**	0.146	0.671
	(iii)AUD-(ph+se)	**0.809**	0.166	0.752
5000 pairs	(i)AUD-ph	**0.774**	0.050	0.518
	(ii)AUD-(ph^{-}+se)	**0.658**	0.109	0.544
	(iii)AUD-(ph+se)	**0.717**	0.111	0.607

by the numbers in bold which are the highest for each row, and the numbers in red which are the highest for each column. It is also clear that the speaker characteristics disentanglement is helpful, since row (iii) for AUD-(ph+se) was always better than row (ii) for AUD-(ph^{-}+se).

Similar trends can be observed in the other parts of Table 3.1 for 3000 and 5000 pairs, except the accuracies were lower, probably because for more pairs the parallelizing transformation became more difficult and less accurate. The only difference is that in these parts column (a) for TXT-ph had the highest accuracies, probably because the goal of semantic embedding for rows (ii)(iii) (AUD-(ph^{-}+se), AUD-(ph+se)) was really difficult, and disturbed or even dominated by phonetic structures. Similar trends can be observed in

doi:10.6342/NTU202302463

Table 3.3: Some examples of top-10 nearest neighbors in AUD-(ph+se) (proposed), AUD-ph (with phonetic structure) and TXT-(se,1h) (with semantics). The words in red are the common words of AUD-(ph+se) and AUD-ph, and the words in bold are the common words of AUD-(ph+se) and TXT-(se,1h).

words	**AUD-(ph+se)**	**AUD-ph**	**TXT-(se,1h)**
owned	own, only, unknown, owner, land, armed, **learned**, homes, **known**, alone	owns, armed, owen, arm, own, only, oughtnt, loaned, ode, owing	visited, introduced, lived, related, **learned**, discovered, met, called, think, **known**
didn't	did, sitting, give, **doesn't**, **don't**, given, hadn't, too, bidden, listen	giving, bidden, given, getting, being, even, ridden, didnt, deane, givin	**don't**, can't, wouldn't, **doesn't**, won't, i'm, you're, shouldn't, think, want

Table 3.2 for top-10 accuracies, obviously with higher numbers for top-10 as compared to those for top-1 in Table 3.1.

In Table 3.3, we list some examples of top-10 nearest neighbors in AUD-(ph+se) (proposed), AUD-ph (with phonetic structure) and TXT-(se,1h) (with semantics). The words in red are the common words for AUD-(ph+se) and AUD-ph, and the words in bold are the common words of AUD-(ph+se) and TXT-(se,1h). For example, the word "owned" has two common semantically related words "learned" and "known" in the top-10 nearest neighbors of AUD-(ph+se) and TXT-(se,1h). The word "owned" also has three common phonetically similar words "armed", "own" and "only" in the top-10 nearest neighbors of AUD-(ph+se) and AUD-ph. This is even clearer for the function word "didn't". These clearly illustrate the phonetic-and-semantic nature of AUD-(ph+se).

Results of Spoken Document Retrieval

The goal here is to retrieve not only those spoken documents including the spoken query (e.g. "President Donald Trump") based on the phonetic structures, but those including

doi:10.6342/NTU202302463

words semantically related to the query word (e.g. "White House"). Below we show the effectiveness of the phonetic-and-semantc embedding proposed here in this application.

We used the 960 hours of "clean" and "other" parts of LibriSpeech dataset as the target archive for retrieval, which consisted of 1478 audio books with 5466 chapters. Each chapter included 1 to 204 utterances or 5 to 6529 spoken words. In our experiments, the queries were the keywords in the book titles, and the spoken documents were the chapters. We chose 100 queries out of 100 randomly selected book titles, and our goal was to retrieve query-relevant documents. For each query q, we defined two sets of query-relevant documents: The first set D_1^q consisted of chapters which included the query q. The second set D_2^q consisted of chapters whose content didn't contain q, but these chapters belonged to books whose titles contain q (so we assume these chapters are semantically related to q). Obviously D_1^q and D_2^q were mutually exclusive, and D_2^q were the target for semantic retrieval, but couldn't be retrieved based on the phonetic structures only.

For each query q and each document d, the relevance score of d with respect to q, $s(q, d)$, is defined as follows:

$$s(q, d) = \max_{w \text{ in } d} -\|R(w) - R(q)\|_2, \tag{3.6}$$

where $R(w)$ is the audio embedding of a word w in d. So (3.6) indicates the documents d were ranked by the minimum distance between a word w in d and the query q. We used mean average precision (MAP) as the evaluation metric for the spoken document retrieval test.

We compared the retrieval results with two versions of audio embedding: AUD-(ph+se) and AUD-ph. The results are listed in Table 3.4 for two definitions of groundtruth for the query-relevant documents: the union of D_1 and D_2 and D_2 alone. As can be found

doi:10.6342/NTU202302463

Table 3.4: Spoken document retrieval performance using two different audio embeddings (AUD-(ph+se) and AUD-ph).

groundtruth	**AUD-(ph+se)**	**AUD-ph**
D_1 + D_2	17.8%	15.6%
D_2	2.8%	1.8%

Table 3.5: Some retrieval examples of chapters in D_2 using AUD-(ph+se) show the advantage of semantics information in phonetic-and-semantic embeddings. The word in red in each row indicates the word with the highest similarity to the query in the chapter.

(a) query q	**(b) title of a book** b	**(c) chapter**	**(d) rank**	**(e) the word with the highest similarity to the query**
nations	Myths and Legends of All Nations	Prometheus the Friend of Man	13/5273	...and shall marry the king of that country...
Anne	Anne of Green Gables	Mrs. Rachel Lynde Is Surprised	25/5329	...why the worthy woman finally concluded...
German	In a German Pension	Story 13: A Blaze	22/5232	...through the heavy snow towards the town...
castle	Montezuma's Castle and Other Weird Tales	THE STRANGE POWDER...	3/5141	...what is its history asked doctor Farrington...
baron	Surprising Adventures of Baron Munchausen	Chapter 22	18/5375	...at the palace and having remained in this situation...

from this table, AUD-(ph+se) offered better retrieval performance than AUD-ph in both rows. Note that those chapters in D_2 in the second row of the table did not include the query q, so couldn't be well retrieved using phonetic embedding alone. That is why the phonetic-and-semantic embedding proposed here can help.

In Table 3.5, we list some chapters in D_2 retrieved using AUD-(ph+se) embeddings to illustrate the advantage of the phonetic-and-semantic embeddings. In this table, column (a) is the query q, column (b) is the title of a book b which had chapters in D_2^q, column (c) is a certain chapter chp in b, column (d) is the rank of chp out of all chapters whose content didn't contain q, and column (e) is a part of the content in chp where the word in red is the word in chp with the highest similarity to q. For example, in the first row for the

doi:10.6342/NTU202302463

query “nations”, the chapter “Prometheus the Friend of Man” of the book titled “Myths and Legends of All Nations” is in $D_2^{nations}$. The word “nations” is not in the content of this chapter. However, because the word “king” semantically related to “nations” is in the content, this chapter was ranked the 13th among all chapters whose content didn”’t contain the word “nations”. This clearly verified why the semantics in the phonetic-and-semantic embeddings can remarkably improve the performance of spoken content retrieval.

3.1.4 Conclusion

In this section we propose a framework to embed spoken words into vector representations carrying both the phonetic structure and semantics of the word. This is intrinsically challenging because the phonetic structure and the semantics of spoken words inevitably disturbs each other. But this phonetic-and-semantic embedding nature is desired and attractive, for example in the application task of spoken document retrieval. A parallelizing transformation between the audio and text embeddings is also proposed to evaluate whether such a goal is achieved.

However, the approach in this section require segmented spoken words as input data, while word-level segmentation is another challenging problem in speech. In Section 3.2, we apply the disentanglement approach on frame-level speech signals where word-level segmentation is not required and use it for the accented speech recognition task. In Section 3.3, we study the joint learning of automatic word-level segmentation and disentanglement of speech signals.

doi:10.6342/NTU202302463

3.2 AIPNet: Generative Adversarial Pretraining of Accent-invariant Networks for End-to-end Speech Recognition

This section focuses on learning accent-invariance with the goal of building a unified accent-independent system for end-to-end speech recognition. Pretraining has shown its superiority in many computer vision and NLP tasks [107] [109] [2], while research efforts on accent model pretraining thus far have been limited. We propose a novel pretraining framework `AIPNet` based on GAN for accent-invariant representation learning: **A**ccent **I**nvariant **P**re-training **Net**works. Unlike most of the existing work that unites the modeling of acoustics and accents in a single stage, our approach decouples accent modeling from acoustic modeling and consists of two stages: pretraining and finetuning. In the pretraining stage, `AIPNet` is built through adversarial training to disentangle accent-invariant and accent-specific characteristics from acoustic features. As transcriptions are not needed in pretraining, `AIPNet` allows us to make use of many possible accent resources for which transcriptions are unavailable. In the finetuning stage, we adopt an attention-based encoder-decoder model for sequence-to-sequence speech recognition. Specifically, we plug in the accent-invariant embeddings in `AIPNet` into ASR model for further optimization. Experimental results on 9 English accents show significant WER reduction compared to four popular baselines, indicating the effectiveness of `AIPNet` on accent-invariance modeling. As a general framework for learning domain-invariance, `AIPNet` can be easily generalized to model any variabilities, such as speakers or speech noise, in addition to accents.

doi:10.6342/NTU202302463

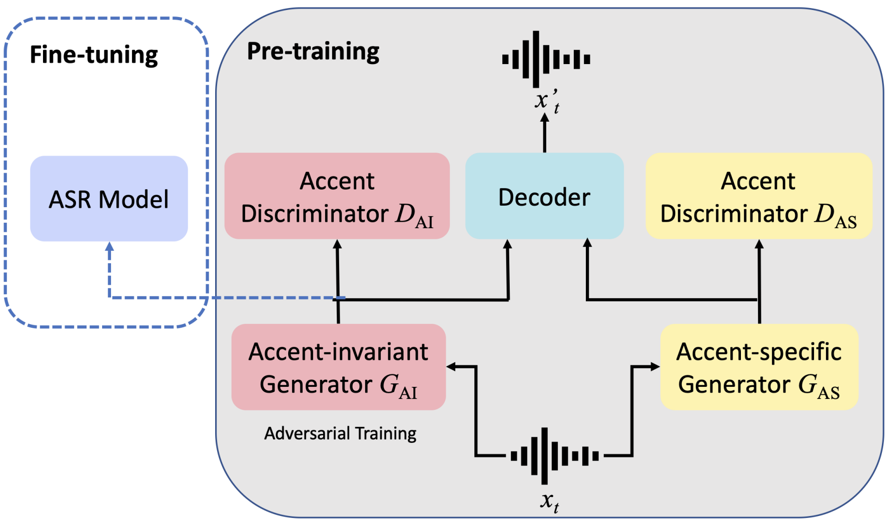

Fine-tuning
Pre-training
ASR Model
x'_t
Accent Discriminator D_{AI}
Decoder
Accent Discriminator D_{AS}
Accent-invariant Generator G_{AI}
Accent-specific Generator G_{AS}
Adversarial Training
x_t

Figure 3.3: The framework of `AIPNet` including both pretraining and finetuning stages.

3.2.1 AIPNet

In this subsection we describe `AIPNet` in details. Our approach consists of two stages: pretraining and finetuning. In the pretraining stage, the model is built through adversarial training with the goal of learning accent-invariant representations. In the finetuning stage, we stack the pretrained model with downstream tasks for further optimization. In this section, we use end-to-end ASR as a downstream application, focusing on improving accent robustness for speech recognition. The framework of `AIPNet` is illustrated in Figure 3.3.

Suppose the input is an utterance $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$, where $\mathbf{x}_t$ represents the feature vector at time step t. The speaker accent corresponding to $\mathbf{x}_t$ is denoted as $a_t \in \{1, 2, ..., C\}$, where C is the number of accents in the training data.

Accent-Invariance Pretraining

The goal of pretraining is to learn accent-invariant representations from accented training data. We define three types of losses for this purpose, including adversarial loss to

doi:10.6342/NTU202302463

disentangle accent-invariant and accent-specific information, reconstruction loss to enforce acoustic characteristics to be preserved in the disentangled representations, as well as consistency regularization to detach linguistic information from accent-specific representations.

Adversarial Loss To learn accent-invariant representations, we define two mappings from speech data: accent-invariant generator $G_{AI}(\mathbf{x}_t)$ and accent-specific generator $G_{AS}(\mathbf{x}_t)$. We also define two discriminators $D_{AI}(G_{AI})$ and $D_{AS}(G_{AS})$ that output probabilities of accents to ensure that G_{AI} and G_{AS} encode the corresponding information. Specifically, we train D_{AI} and D_{AS} to maximize the probability of assigning correct accent labels to samples from G_{AI} and G_{AS} respectively, *i.e.,* minimizing cross-entropy loss L_{CE}^{AI} and L_{CE}^{AS}:

$$\min_{D_{AS},G_{AS}} L_{CE}^{AS} = \sum_{t=1}^{T} -\log P(a_t|G_{AS}(\mathbf{x}_t)), \tag{3.7}$$

$$\min_{D_{AI}} L_{CE}^{AI} = \sum_{t=1}^{T} -\log P(a_t|G_{AI}(\mathbf{x}_t)). \tag{3.8}$$

To decouple accent-related information from G_{AI}, we simultaneously train G_{AI} such that D_{AI} is confused about accent labels of samples from G_{AI}. The objective is to maximize cross-entropy loss L_{CE}^{AI}, equivalent to minimize the negative cross-entropy:

$$\min_{G_{AI}} -L_{CE}^{AI} = \sum_{t=1}^{T} \log P(a_t|G_{AI}(\mathbf{x}_t)). \tag{3.9}$$

Reconstruction Loss The adversarial loss defined between D_{AI} and G_{AI} enforces that accent-specific information is disentangled from G_{AI} but preserved in G_{AS}. To ensure acoustics characteristics are encoded in the representations from both generators, we further define a decoder with autoencoding structure to reconstruct speech feature $\mathbf{x}_t$ as $\mathbf{x}'_t$ from the concatenation of $G_{AI}(\mathbf{x}_t)$ and $G_{AS}(\mathbf{x}_t)$. The decoder is trained by minimizing

doi:10.6342/NTU202302463

the reconstruction error L_R:

$$\min_{decoder, G_{AI}, G_{AS}} L_R = \sum_{t=1}^{T} \| \mathbf{x}'_t - \mathbf{x}_t \|_2^2 \,. \tag{3.10}$$

Consistency Regularization Accent-specific attributes are generally stable within an utterance while linguistic-related acoustics have larger intra-utterance variance across time frames. Inspired by the utterance-level stability of accent-specific attributes, we impose a consistency regularization for $G_{AS}(\mathbf{x}_t)$ such that accent-specific representations from G_{AS} are consistent across time frames within an utterance:

$$\min_{G_{AS}} L_{CR} = \sum_{t=1}^{T-1} \| G_{AS}(\mathbf{x}_{t+1}) - G_{AS}(\mathbf{x}_t) \|_2^2 \,. \tag{3.11}$$

This regularization reinforces the preservation of accent-specific information in G_{AS} meanwhile implicitly encourages linguistic content to be disentangled from G_{AS}. The multi-scale nature of information in speech data has also been applied in voice conversion and speech denoising [123].

Iterative Training Given the minmax two-player game between D_{AI} and G_{AI}, `AIPNet` pretraining is designed of repeating the following two steps in an iterative manner.

- Update the discriminator D_{AI} by minimizing L_D,
- Freeze the discriminator D_{AI} and update the rest of the network by minimizing L_G,

$$L_D = L_{CE}^{AI}, \tag{3.12}$$

$$L_G = -L_{CE}^{AI} + \lambda_1 L_{CE}^{AS} + \lambda_2 L_R + \lambda_3 L_{CR}, \tag{3.13}$$

where λs are hyper-parameters.

doi:10.6342/NTU202302463

Finetuning for End-to-End Speech Recognition

In the finetuning stage, the outputs of G_{AI} which encode accent-invariant linguistic content can be plugged in as inputs of any downstream speech tasks that aim to improve accent robustness, as shown in Figure 3.3. In this section, we focus on multi-accent speech recognition and adopt Listen, attend and spell (`LAS`), a popular attention-based encoder-decoder model [138] for sequence-to-sequence speech recognition. `LAS` consists of an encoder encoding an input sequence into high-level representations as well as an attention-based decoder generating a sequence of labels from the encoded representations. The encoder is typically a unidirectional or bidirectional LSTM and the decoder is a unidirectional LSTM.

The label inventory for `LAS` modeling consists of 200 word pieces and is further augmented with two special symbols <sos> and <eos> indicating the start of a sentence and the end of a sentence respectively. `LAS` models the posterior probability of a label sequence $\mathbf{y}$ given the input feature sequence $G_{AI}(\mathbf{X})$ and the previous label history $\mathbf{y}_{1:j-1}$:

$$P(\mathbf{y}|G_{AI}(\mathbf{X})) = \prod_{j=1} P(y_j|G_{AI}(\mathbf{X}), \mathbf{y}_{1:j-1}). \tag{3.14}$$

Both encoder and decoder can be trained jointly for speech recognition by maximizing the log probability or minimizing L_{ASR}:

$$L_{ASR} = \sum_{j=1} -\log P(y_j|G_{AI}(\mathbf{X}), \mathbf{y}_{1:j-1}). \tag{3.15}$$

There are two ways of finetuning: 1) finetune G_{AI} and `LAS` with L_{ASR}. This requires only transcriptions in the training data; 2) continue with adversarial training with $L'_G = L_G + \lambda_4 L_{ASR}$. This requires both transcriptions and accent labels in the training data. In the experiments, we report results of using both ways.

doi:10.6342/NTU202302463

3.2.2 Experiments

Dataset

The dataset used in experiments contains utterances in a variety of domains, such as weather or music, collected through crowdsourced workers. There are 9 English accents in total in the dataset, including United States (US), Korea (KR), Philippines (PH), Canada (CA), India (IN), France (FR), Britain (GB), Vietnam (VN) and Latin America (LA). The training set contains 4M (3.8K hours) utterances among which 1% is split as validation data. Particularly, there are 1M and 780K utterances in US and LA respectively and about 330K data in each of the remaining accents. The testing set has 10.8K utterances with 1.2K utterances in each accent. In both training and testing sets, we extract acoustic features using 80-dimensional log Mel-filterbank energies that are computed over a 25ms window every 10ms.

Experimental Setup

The architecture of each module in `AIPNet` is a multi-layer LSTM. Specifically, we represent G_{AI}, G_{AS} and decoder using 2 LSTM layers with a hidden size of 768, 256 and 1024 respectively. D_{AI} and D_{AS} are represented by a LSTM layer with softmax outputs. The configuration of `LAS` includes a 4-layer LSTM encoder and a 2-layer LSTM decoder, each with a hidden size of 1024. The hyperparameters $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ are swept within the range $[0.1, 30]$. Our experiments have shown that the final results are generally stable across different hyperparameter settings. For simplicity, we report results with $(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (1, 10, 10, 10)$ in this section. We use batch size of $16,000$ tokens with 32 GPUs for training. We use Adam with learning rate of 5×10^{-4} in pretraining and

doi:10.6342/NTU202302463

2.5×10^{-4} in finetuning, $\beta_1 = 0.9$, $\beta_2 = 0.999$. A dropout rate of 0.1 is applied to all the layers. We pretrain `AIPNet` for 15 epochs and finetune `LAS` for 20 epochs. During inference, speech features are fed into G_{AI} that is absorbed as part of `LAS` encoder and outputs of `LAS` are decoded using beam size of 20 without any external language model.

Baselines

We compare our approach against four popular baselines `B1`-`B4` for multi-accent speech recognition in the experiments. `B1` is an accent-independent model which is trained on the data from all the accents. `B2` and `B3` have shown strong performance on multi-accent speech recognition in [76]. Specifically, we append accent labels at the end of each label sequence and `B2` is trained on the updated sequences from all accents. As accent information is not required at inference, `B2` is accent-independent. When training `B3` which is accent-dependent, we transform accent 1-hot vector into an embedding through a learnable linear matrix and feed the learned embedding into `LAS` encoder. During `B1`-`B3` training, G_{AI} is part of `LAS` encoder containing 6 LSTM layers. `B4` is the most similar to our approach in spirits, aiming to learn accent-invariant features through gradient reversal [83]. The gradient reversal approach keeps modules of G_{AI}, D_{AI} and ASR model in Figure 3.3. Instead of using iterative training in Subsection 3.2.1, we add a gradient reversal layer between G_{AI} and D_{AI} to reverse the backpropagated gradient for G_{AI} training. For more details about `B4`, we refer readers to [83].

Experimental Results

As described in Subsection 3.2.1, `AIPNet` pretraining requires only accent labels in the training data. This approach hence becomes especially useful when there is a large num-

doi:10.6342/NTU202302463

Table 3.6: WER (%) of different approaches in each accent in supervised setting. F1 indicates finetuning with L_{ASR}; F2 indicates finetuning with L'_G; AI indicates accent-independent model; AD indicates accent-dependent model.

Approach			Ave.	US	Non-US	CA	FR	IN	KR	PH	LA	GB	VN
Baselines	B1	AI	8.7	5.7	9.0	6.4	8.4	11.2	**9.9**	7.2	**7.8**	8.0	13.0
	B2	AI	8.8	**5.0**	9.1	6.6	9.3	11.0	10.3	**6.7**	8.1	8.1	12.9
	B3	AD	8.6	5.4	8.9	6.7	8.5	10.9	10.0	6.8	8.6	7.9	**12.0**
	B4	AI	8.8	5.8	9.1	6.1	8.5	11.7	10.7	7.4	8.4	**7.8**	**12.0**
AIPNet	F1	AI	**8.4**	5.6	**8.7**	**6.0**	**8.1**	**9.9**	10.3	6.9	8.0	**7.8**	12.4
	F2	AI	10.1	6.2	10.5	7.9	10.1	12.8	12.1	8.2	9.5	9.4	13.9

ber of accented data without available transcriptions. We design experiments in two settings, *i.e.,* supervised setting where transcriptions are available in all accents and semi-supervised setting where transcriptions are available only in US accent.

Results in Supervised Setting Table 3.6 summarizes the results of different approaches in supervised setting. In our approach, we report results of finetuning G_{AI} and LAS with L_{ASR} using transcriptions (F1), as well as those of finetuning the entire network with L'_G using both transcriptions and accent labels (F2). We can see that finetuning with L_{ASR} (F1) outperforms the baselines by $2.3 \sim 4.5\%$ relative reduction on average WER. Compared to all the baselines, F1 has achieved improvement in CA, FR, GB, and especially IN ($9.1 \sim 15.3\%$ reduction) but has shown a mediocre performance in each of the remaining accents.

Results in Semi-supervised Setting In semi-supervised setting where transcriptions are available only in US accent, we compare the performance between B1 and F1. The results

doi:10.6342/NTU202302463

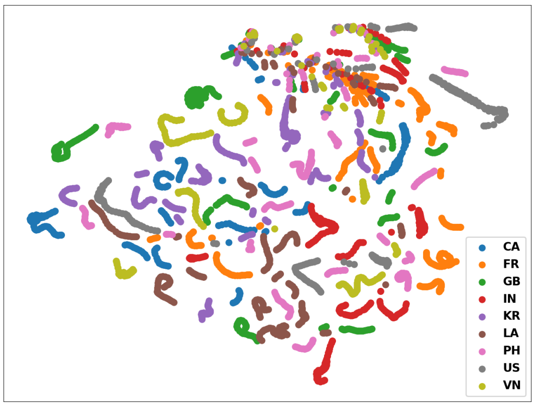
Table 3.7: WER (%) of different approaches in each accent in semi-supervised setting. `F1` indicates finetuning with L_{ASR}; `F2` indicates finetuning with L'_G; PL indicates pseudo labeling; AI indicates accent-independent model; AD indicates accent-dependent model.

Approach				Ave.	US	Non-US	CA	FR	IN	KR	PH	LA	GB	VN
w/o PL	Baseline	`B1`	AI	29.9	10.6	31.8	22.0	33.1	41.0	33.1	28.2	28.7	28.3	40.6
	`AIPNet`	`F1`	AI	27.9	**9.4**	29.8	20.1	30.8	39.0	32.8	25.5	26.4	25.3	39.2
w/ PL	Baselines	`B1`	AI	26.2	10.3	27.8	18.6	28.3	36.1	29.6	24.8	25.1	24.4	35.8
		`B2`	AI	25.9	**9.4**	27.6	19.0	27.7	36.5	29.6	24.2	23.8	25.1	34.9
		`B3`	AD	25.9	9.6	27.5	19.5	28.0	36.4	29.1	23.7	24.2	24.8	35.0
		`B4`	AI	25.0	9.7	26.5	**18.1**	26.7	34.9	28.3	23.7	23.4	23.7	33.6
w/ PL	`AIPNet`	`F1`	AI	25.7	12.1	27.0	19.7	27.4	34.7	28.9	23.0	23.6	24.5	34.6
		`F2`	AI	**24.6**	11.8	**25.9**	19.0	**26.0**	**32.6**	**28.0**	**22.2**	**22.8**	**23.1**	**33.5**

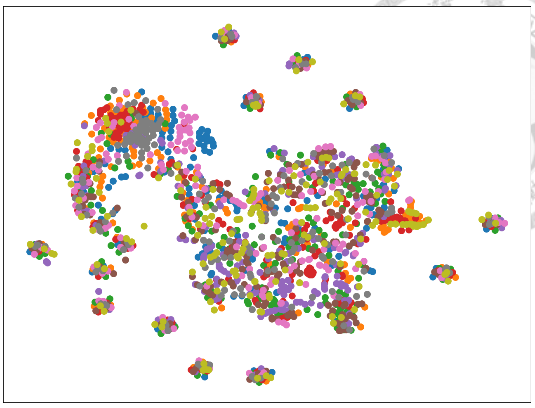
are presented in the first two rows of Table 3.7. As `B2`, `B3`, `B4` and `F2` require the availability of pairs of transcriptions and accent labels for training, the results of these approaches are not available in such scenario. The results have shown that our approach significantly outperforms the baseline model in all accents, achieving $3.4 \sim 11.3\%$ relative WER reduction.

One popular and effective method for semi-supervised learning is to generate target pseudo labels for unlabeled data using an initial model [139]. To achieve better performance, we generate pseudo transcriptions for non-US training data using the US model. As a result, we are able to follow all the experiments in supervised setting. The results with pseudo labeling (PL) are presented in the last six rows of Table 3.7. By comparing the performance between models with and without pseudo labeling, we can observe that pseudo labeling has shown significant gains for all the approaches and almost in each accent, exhibiting its effectiveness on improving generalization performance using unlabeled

doi:10.6342/NTU202302463



(a) G_{AI} embedding from B1.



(b) G_{AI} embedding from F2.

Figure 3.4: t-SNE 2-D plots of G_{AI} embedding from B1 and F2 (w/ PL) in each accent. Each color represents each accent.

data. In the scenario with pseudo labeling, finetuning with L'_G (F2) outperforms the baselines by $1.6 \sim 6.1\%$ relative reduction on average WER and consistently achieves the best performance in all non-US accents except for CA.

Analysis

In this subsection, we analyze the properties of AIPNet to better understand its superiority for multi-accent speech recognition. Without loss of generality, we use B1 and F2 (w/ PL) in semi-supervised setting in the analysis.

Learning accent-invariance To comprehend the effectiveness of AIPNet on learning accent-invariant representations, we extract embedding (outputs) of G_{AI} from B1 and F2 respectively for 300 data samples in each accent. Figure 3.4 shows t-SNE 2-D visualization of G_{AI} embedding from B1 (Figure 3.4a) and F2 (Figure 3.4b) respectively for each accent [140]. As can be seen, G_{AI} outputs from the baseline B1 tend to be clustered in each accent while those from our approach F2 are mixed across different accents. The visualization

doi:10.6342/NTU202302463

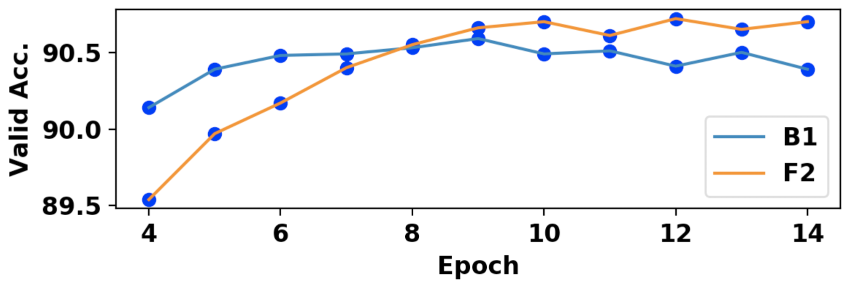


Figure 3.5: Word piece validation accuracy of ASR model in B1 and F2.

demonstrates the validity of the accent-invariant features learned through `AIPNet` and further explains the better generalization performance that our approach has achieved across accents.

Reducing overfitting We further investigate the trend of word piece validation accuracy of the ASR model in `B1` and `F2`, as shown in Figure 3.5. Compared to `B1`, `F2` learns more slowly and reaches a better local optimal. The learning objective of `F2` consists of both L_{ASR} and accent-related regularizers (see Subsection 3.2.1). This observation corroborates the effectiveness of the regularization in our approach on reducing the risk of overfitting. It is worth noting that such benefit from the accent-related regularization in finetuning is not observed in supervised setting (see Table 3.6). One possible reason could be that the sufficient labeled training data in supervised setting empowers the ASR model a strong learning capability that might be even weakened by additional regularizations.

3.2.3 Conclusion

In this section, we proposed `AIPNet`, a GAN-based pretraining network, for learning accent-invariant representations, aiming to build a unified speech recognition system that generalizes well across accents. As transcriptions are not needed in pretraining, `AIPNet` provides

doi:10.6342/NTU202302463

the flexibility of making use of many possible accent resources for which transcriptions are unavailable. Experiments have shown promising results on 9 English accents compared to the baselines, especially in the case when transcriptions are not available in all accents. Experimental results have demonstrated the effectiveness of `AIPNet` on learning accent-invariance.

In Section 3.3, in addition to disentangled representation learning, we further deal with another big challenge in speech, word boundary segmentation.

3.3 Sequence-to-Sequence Autoencoding for Unsupervised Learning of Audio Segmentation and Representation

Human infants acquire languages with little formal teaching; machines, however, must learn from a large amount of annotated data, which makes the development of speech technology for a new language challenging. For typical spoken language understanding, one can simply convert spoken content into word sequences using an off-the-shelf speech recognizer. However, to train a high-quality speech recognition system, huge quantities of annotated audio data are needed. Therefore, for low-resource languages with scarce annotated data, or languages without written forms, sufficiently accurate speech recognition is difficult to achieve. Some previous work [141, 142] focus on speech recognition with mismatched crowdsourcing and probabilistic transcriptions.

Annotating audio data for speech recognition is expensive, but unannotated audio data is relatively easy to collect. If the machine can acquire the word patterns behind speech signals from a large collection of unannotated speech data without speech recognition, it

doi:10.6342/NTU202302463

would be able to learn a new language in a novel linguistic environment with little supervision. Imagine a Hokkien-speaking family buying an intelligent device: although at first the machine does not understand Hokkien, by hearing people speak it, it automatically learns the language. This section is one step toward this dream [143, 144, 145].

In this section, a sequence-to-sequence autoencoder (SA) is used to represent variable-length audio segments using fixed-length vectors [1, 146]. SA consists of an RNN encoder and decoder. The RNN encoder reads an audio segment represented as an acoustic feature sequence and maps it to a vector representation with a fixed length of z; the RNN decoder maps the vector z to another sequence. The RNN encoder and decoder are trained to minimize the reconstruction error of the input acoustic sequence.

Because the representation z extracted by the RNN encoder must reconstruct the input signals, it includes not only phonetic information but also speaker, environment, and channel information in various dimensions. That is, audio segments corresponding to the same terms with the same phonetic structure may have different z if produced by different speakers. Therefore, it is necessary to disentangle the phonetic information in z from other information, so the phonetic information can be further used in downstream applications. As in Section 3.1, we use an adversarial classifier. The classifier learns to distinguish whether two segments were uttered by the same speaker or not based on the representations. To confuse the classifier, the encoder learns to extract speaker-invariant representations.

Word segmentation of speech is critical but challenging for zero-resource speech technology because word boundaries are usually not available for given speech utterances or corpora [143, 30, 147, 148]. Although there are approaches to estimating word bound-

doi:10.6342/NTU202302463

aries [30, 149, 150, 151, 50], we hypothesize that the audio segmentation and SA can be integrated and jointly learned, so that they can enhance each other. This means that the machine learns to segment the utterances into a sequence of spoken words while at the same time transforming these spoken words into a sequence of vectors.

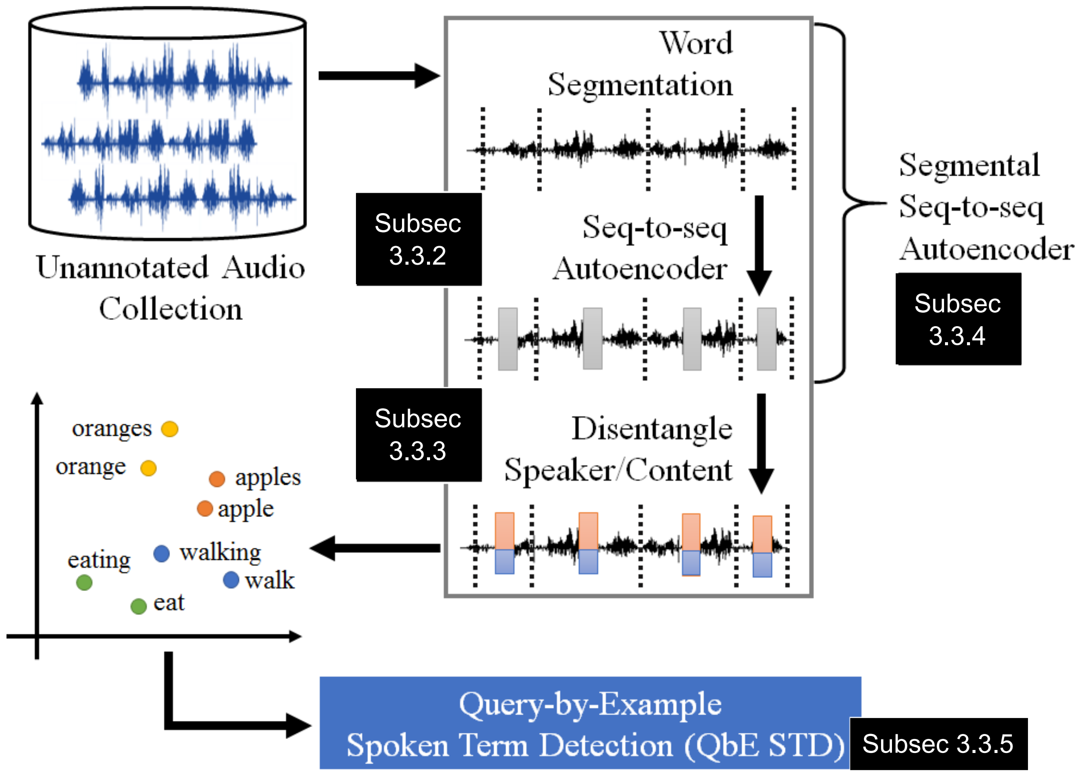
We propose the segmental sequence-to-sequence autoencoder (SSAE) [26], a new model to jointly train the segmenter while extracting the representation. The SSAE contains a segmentation gate jointly learned with SA from an unlabeled corpus in a completely unsupervised way. During training, the SSAE learns to convert the utterances into sequences of embeddings, and then reconstructs the utterances with these embedding sequences. The only thing needed during training is a guideline for the proper number of vectors (or words) within an utterance of a given length, to ensure that the machine segments the utterances into word-level segments. Since the model is not completely differentiable, standard backpropagation is not applicable [152, 153]; thus we use reinforcement learning to train SSAE.

In this section, we employ query-by-example spoken term detection (QbE STD), a real-world application, to evaluate the phonetic structure information of the original utterances contained in these generated word vector sequences. When based on audio word2vec, QbE STD is much more efficient than conventional dynamic time warping (DTW) based approaches, because only the similarities between two single vectors are needed; this is in addition to the significantly better retrieval performance that it yields.

The audio word2vec framework is summarized in Figure 3.6. Given a large collection of annotated audio, it is first segmented into word-level segments. Then the SA model generates an embedding for each audio segment, as described in Subsection 3.3.2. In

doi:10.6342/NTU202302463

Subsection 3.3.3, we also describe how to disentangle speaker and speech content from the embedding. In Subsection 3.3.4, we describe the proposed SSAE model, which jointly learns segmentation and embedding. In Subsection 3.3.5 the embedding is evaluated on QbE STD.



Unannotated Audio Collection
Word Segmentation
Subsec 3.3.2
Seq-to-seq Autoencoder
Segmental Seq-to-seq Autoencoder
Subsec 3.3.4
Subsec 3.3.3
Disentangle Speaker/Content
oranges
orange
apples
apple
eating
walking
walk
eat
Query-by-Example Spoken Term Detection (QbE STD)
Subsec 3.3.5

Figure 3.6: Audio word2vec framework

3.3.1 Audio Representation

The goal for the audio word2vec model is to identify the phonetic patterns in sequences of acoustic features such as MFCCs. Given a word-level audio segment $\mathbf{x} = (x_1, x_2, ..., x_T)$ where x_t is the acoustic feature at time t, and T is the length, audio word2vec transforms the features into a fixed-length vector $\mathbf{z} \in \mathbb{R}^d$ with dimension d. In this subsection, we assume the word boundaries are ready available. In the next subsection we describe how

doi:10.6342/NTU202302463

to jointly learn segmentation and representation.

3.3.2 Sequence-to-sequence Autoencoder

Recurrent neural networks (RNNs) have shown great success in many NLP tasks with their ability to capture sequential information. The hidden neurons form a directed cycle and perform the same task for every element in a sequence. Given a sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$, the RNN updates its hidden state h_t according to the current input x_t and the previous h_{t-1}. The hidden state h_t acts as an internal memory at time t that enables the network to capture dynamic temporal information, and also allows the network to process sequences of variable length.

The RNN encoder-decoder architecture [23, 13] consists of an RNN encoder and an RNN decoder. The encoder reads the input sequence $\mathbf{x} = (x_1, x_2, ..., x_{T_1})$ sequentially, and the hidden state h_t of the RNN is updated accordingly. After the last symbol x_{T_1} is processed, the hidden state h_{T_1} is interpreted as the learned representation of the whole input sequence. Then, taking h_{T_1} as input, the RNN decoder generates the output sequence $\mathbf{y} = (y_1, y_2, ..., y_{T_2})$ sequentially, where T_1 and T_2 can be different, or the length of $\mathbf{x}$ and $\mathbf{y}$ can be different. This RNN encoder-decoder framework is able to handle variable-length input and output.

Figure 3.7 depicts the structure of the sequence-to-sequence autoencoder (SA), which integrates the RNN encoder-decoder framework with an autoencoder for the unsupervised learning of audio segment representations. The SA consists of an RNN encoder (the left part of Figure 3.7) and decoder (the right part). Given an audio segment represented as an acoustic feature sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$ of any length T, the RNN encoder reads

doi:10.6342/NTU202302463

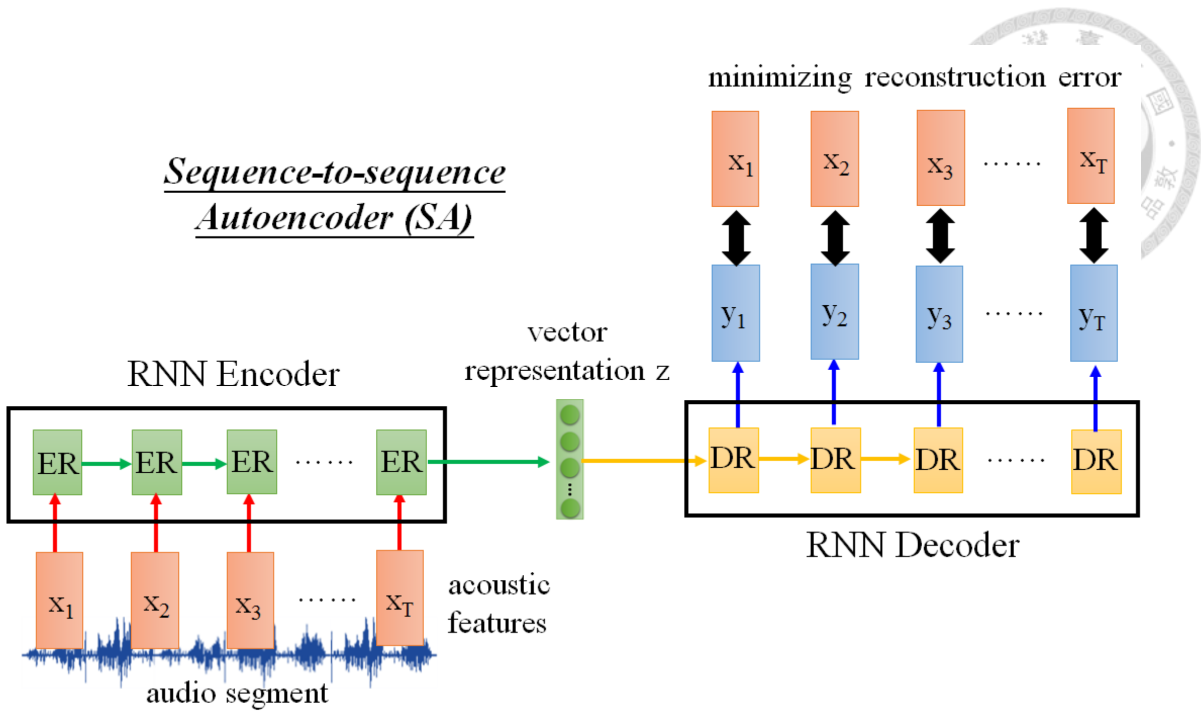


Figure 3.7: Sequence-to-sequence autoencoder (SA), consisting an RNN encoder (ER) and an RNN decoder (DR). The encoder reads an audio segment represented as an acoustic feature sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$ and maps it to a fixed-length vector representation with dimension $\mathbf{z}$; the decoder maps the vector z to another sequence $\mathbf{y} = (y_1, y_2, ..., y_T)$. The RNN encoder and decoder are jointly trained such that the output sequence $\mathbf{y}$ is as close to the input sequence $\mathbf{x}$ as possible.

each acoustic feature x_t sequentially and the hidden state h_t is updated accordingly. After the last acoustic feature x_T has been read and processed, the hidden state h_T of the encoder is taken as the learned representation z of the input sequence (the vector in the middle of Figure 3.7).

The RNN decoder takes h_T as the initial state, and generates a sequence $\mathbf{y}$. Based on autoencoder principles [154, 155], the target of the output sequence $\mathbf{y} = (y_1, y_2, ..., y_T)$ is the input sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$. In other words, the RNN encoder and decoder

doi:10.6342/NTU202302463

are jointly trained by minimizing the reconstruction error $\mathcal{L}_{mse}$,

$$\mathcal{L}_{mse} = \sum_{\mathbf{x}} \sum_{t=1}^{T_{\mathbf{x}}} \|x_t - y_t\|^2, \tag{3.16}$$

where $\mathcal{L}_{mse}$ is the sum over all the audio segments **x** in the data collection, and $T_{\mathbf{x}}$ represents the length of the segment **x**. Because the input sequence is taken as the learning target, the training process requires no labeled data. The fixed-length vector representation z is thus a meaningful representation for the input audio segment x because the whole input sequence x can be reconstructed from z with the RNN decoder. Although in Figure 3.7 both the RNN encoder and decoder have only one hidden layer, this does not preclude the use of multiple layers.

In Figure 3.7, after generating output y_1, instead of taking y_1 as the input of the next time step, a zero vector is used as input to generate y_2, and so on, in contrast to the typical encoder-decoder architecture. This use of historyless decoding is critical here. We found that if a typical decoder is used (that is, RNN takes y_1 as input to generate y_2, and so on), despite the resultant low reconstruction error, the SA-learned vector representations do not include useful information. This is because a strong decoder focuses less on including more information in the vector representation. Historyless decoding yields a weakened decoder because the input of the decoder is removed, which forces the model to rely more on the vector representation. Historyless decoding is also used in some NLP applications [156, 157, 158].

3.3.3 Feature Disentanglement

Because the representation z extracted by the RNN encoder in Figure 3.7 must reconstruct the input signals, it includes not only phonetic information but also speaker, environment,

doi:10.6342/NTU202302463

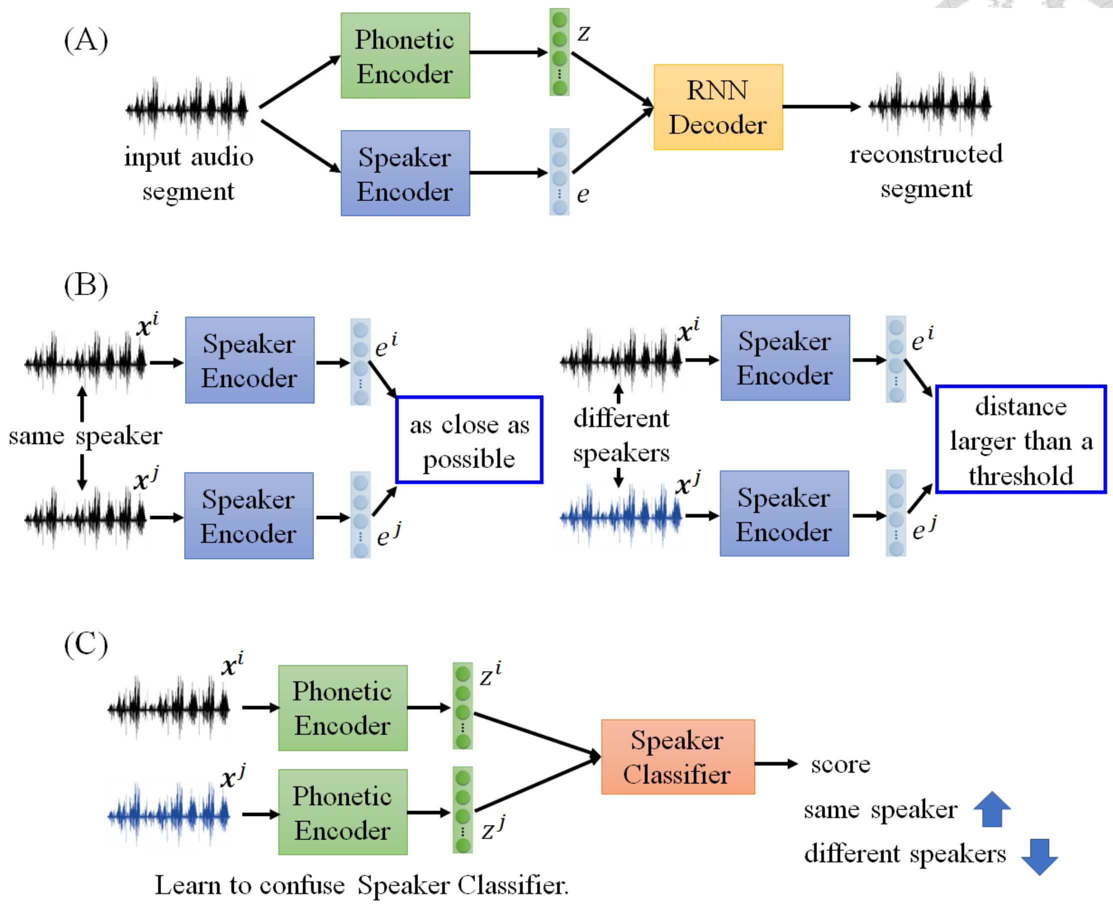


Figure 3.8: Feature disentanglement. (A) Adding speaker encoder for reconstruction. (B) Additional training criteria for the speaker encoder. (C) Speaker classifier learns to distinguish whether two z's are from the same speaker or not; phonetic encoder attempts to confuse the classifier with z.

and channel information in various dimensions. Therefore, it is necessary to disentangle the phonetic information from other information.

As shown in Figure 3.8, the idea of disentanglement is the same as Section 3.1, where we add an additional speaker encoder and use adversarial training to disentangle phonetic and speaker information. The original encoder in Figure 3.7 is the phonetic encoder, which takes the audio segment **x** as input and outputs embedding vector z. The speaker encoder

doi:10.6342/NTU202302463

has the same RNN architecture as the phonetic encoder. Its output embedding is denoted as e. The input of the RNN decoder is the concatenation of vectors z and e. The phonetic encoder, speaker encoder, and RNN decoder are jointly learned to minimize the reconstruction error $\mathcal{L}_{mse}$ in (3.16).

Ensuring that z contains phonetic information and e contains speaker information, we require additional training criteria as in Section 3.1. The speaker encoder learns to minimize the distance between the e of audio segments uttered by the same speaker, and enlarge the distance between the e of different speakers past a threshold. This assumes that speaker labels are available. If speaker information is not available, the speaker encoder can still be learned by assuming that segments from the same utterance are produced by the same speaker. Although we only consider the speaker information here, it is possible to use the same approach to consider other information such as the channel.

As in Section 3.1, we also adopt an adversarial speaker classifier to discriminate if an input pair of two phonetic vectors come from the same speaker. The phonetic encoder tries its utmost to generate z vectors that confuse the speaker classifier. If it successfully achieves this after training, it produces a z that contains no speaker information, and an e that contains all the speaker information. The complete procedure for feature disentanglement is shown in Algorithm 1.

3.3.4 Jointly Learning Segmentation and Representation

Segmental Sequence-to-Sequence Autoencoder (SSAE)

The proposed structure for SSAE is depicted in Figure 3.9, in which the segmentation gate is inserted into the SA. The input of SSAE is an utterance $\mathbf{x}' = \{x_1, x_2, ..., x_{T'}\}$, where $\mathbf{x}_t$

doi:10.6342/NTU202302463

Algorithm 1 Feature Disentanglement

```
Input: Audio segment collection $\mathcal{X}$, total update iterations $T_{train}$, Speaker classifier update iterations $T_{dis}$
Output: Phonetic encoder parameters $\theta_p$
 1: Initialize phonetic encoder parameters $\theta_p$
 2: Initialize speaker encoder parameters $\theta_s$
 3: Initialize RNN decoder parameters $\theta_d$
 4: Initialize speaker classifier parameters $\theta_c$
 5: for $t = 1$ to $T_{train}$ do
 6:     Sample $M$ audio segments $\mathcal{X}_r$ from $\mathcal{X}$
 7:     Sample $M$ audio segment pairs $\mathcal{X}_p$ from $\mathcal{X}$ in which the paired segments are from the same speakers
 8:     Sample $M$ audio segment pairs $\mathcal{X}_n$ from $\mathcal{X}$ in which the paired segments are from different speakers
 9:     % Train phonetic classifier
10:     Compute phonetic classifier loss $\mathcal{L}_c$ based on the $2M$ segment pairs $\{\mathcal{X}_p, \mathcal{X}_n\}$
            $$\mathcal{L}_c = \sum_{(\mathbf{x}^i,\mathbf{x}^j)\in\mathcal{X}_p} C(z^i, z^j) - \sum_{(\mathbf{x}^i,\mathbf{x}^j)\in\mathcal{X}_n} C(z^i, z^j), \quad (3.17)$$
        where $C(z^i, z^j)$ is the speaker classifier output score given the phonetic embedding of the segment pair ($\mathbf{x}^i$, $\mathbf{x}^j$) with parameters $\theta_c$
11:     % Phonetic classifier parameters updated with additional iterations as in typical GAN framework
12:     for $t' = 1$ to $T_{dis}$ do
13:         $\theta_c \leftarrow \theta_c - \eta \bigtriangledown \mathcal{L}_c + gradient\ penalty$
14:     end for
15:     % Minimize reconstruction error
16:     Compute reconstruction loss $\mathcal{L}_{mse}$ in (3.16) over segments in $\mathcal{X}_r$
17:     % Extra training criteria for speaker encoder
18:     Compute loss $\mathcal{L}_s$
            $$\mathcal{L}_s = \sum_{\mathbf{x}^i,\mathbf{x}^j\in\mathcal{X}_p} \|e^i - e^j\|^2 + \sum_{\mathbf{x}^i,\mathbf{x}^j\in\mathcal{X}_n} max(\lambda_d - \|e^i - e^j\|^2, 0), \quad (3.18)$$
        where $e^i$ and $e^j$ are output of speaker encoder given $\mathbf{x}^i$ and $\mathbf{x}^j$.
19:     % Compute the total loss $\mathcal{L}_{tot}$ for updating the whole model
            $$\mathcal{L}_{tot} = \mathcal{L}_{mse} + \mathcal{L}_s - \mathcal{L}_c \quad (3.19)$$
20:     $\theta_p \leftarrow \theta_p - \eta \bigtriangledown \mathcal{L}_{tot}$
21:     $\theta_s \leftarrow \theta_s - \eta \bigtriangledown \mathcal{L}_{tot}$
22:     $\theta_d \leftarrow \theta_d - \eta \bigtriangledown \mathcal{L}_{tot}$
23: end for
```

doi:10.6342/NTU202302463

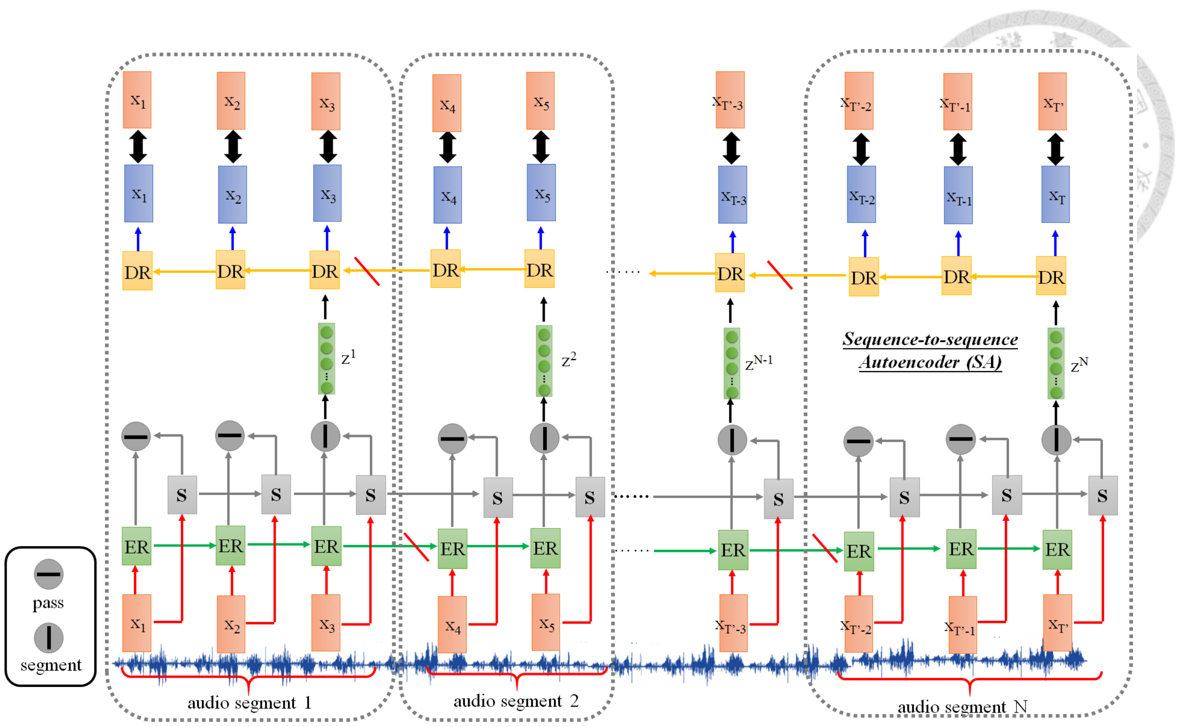


Figure 3.9: Segmental sequence-to-sequence autoencoder (SSAE). In addition to the RNN encoder (ER blocks) and RNN decoder (DR blocks), a segmentation gate (**S** blocks) is included in the model to estimate word boundaries.

represents the t-th acoustic feature, and T' is the length of the utterance. In general, an utterance $\mathbf{x}'$ is much longer than an audio segment x with length T in Subsection 3.3.1, that is, $T' \gg T$. Given an input utterance, the model learns to determine the word boundaries in the utterance, and generates N audio segments. The model then produces the embeddings for the N generated audio segments, $\mathbf{z} = \{z_1, z_2, ..., z_N\}$, where z_n is the n-th embedding and $N \leq T'$. As with the conventional autoencoder, the proposed SSAE consists of an RNN encoder and an RNN decoder. The encoder includes an extra segmentation gate, controlled by another RNN (shown in Figure 3.9 as the sequence of blocks labeled **S**). The segmentation gate can be considered an "agent" in the parlance of typical reinforcement learning. At each time t, the segmentation gate agent performs an action a_t, according

doi:10.6342/NTU202302463

to a given input feature s_t (or *state* per reinforcement learning). The action a_t is either "segment" or "pass". If "segment", then $\mathbf{x}_t$ is regarded as a word boundary.

For the segmentation gate, the input state at time t, s_t, is defined as the concatenation of the input acoustic feature x_t, the gate activation signal (GAS) g_t, and the previous taken action a_{t-1},

$$s_t = \left[x_t||g_t||a_{t-1}\right]. \tag{3.20}$$

The GAS feature g_t is an unsupervised feature for segmentation. This feature is extracted from the values of the update gates of the GRU of another pretrained RNN autoencoder, which is trained simply to minimize the reconstruction loss. We use GAS as extra input formation here because it has been verified that the temporal structure of such signals is correlated with the phoneme boundaries [149]. Here the same set of audio data is used to train the SSAE and learn GAS features. The policy π_t at time t is modeled by the output h_t of the layers of the segmentation gate RNN (**S** blocks in Figure 3.9) followed by a linear transform (W^π,b^π) and a softmax nonlinearity:

$$h_t = \mathit{RNN}(s_1, s_2, ..., s_t), \tag{3.21}$$

$$\pi_t = \mathit{softmax}(W^\pi h_t + b^\pi). \tag{3.22}$$

This π_t gives two probabilities respectively for "segment" and "pass". An action a_t is then sampled from this distribution during training to encourage exploration. During testing, the action with the highest probability is taken.

When a_t is "segment", the time t is viewed as a word boundary, after which the segmentation gate passes the output of the RNN encoder. If this is the n-th time the action "segment" is taken, and last time action "segment" is taken at the t'-th time step, yielding the n-th audio segment, when t' and t refer to the beginning and ending time for the n-th

doi:10.6342/NTU202302463

audio segment. The output of the RNN encoder at time step t is the vector representation of the n-th audio segment z_n:

$$z_n = Encoder(x_{t'}, x_{t'+1}, ..., x_t). \tag{3.23}$$

After "segment" is taken, the internal state of the RNN encoder is reset to its initial value, so z_n is generated based only on the acoustic features of the n-th audio segment.

Then the input utterance **x** is reconstructed with the embedding sequence **z** = $\{z_1, z_2, ..., z_N\}$. Given an embedding z_n for the n-th input segment in (3.23) above, the RNN decoder generates $\{y_{t'}, y_{t'+1}, ..., y_t\}$ to reconstruct the input acoustic features $\{x_{t'}, x_{t'+1}, ..., x_t\}$. When the RNN decoder begins decoding each audio segment, its state is also reset.

In Figure 3.9, each audio segment in the boxes with dotted lines can be viewed as performing the sequence-to-sequence training from Figure 3.7 individually. Note that the sequence-to-sequence training in Figure 3.9 reconstructs the target sequence in reverse order, in contrast to that shown in Figure 3.7. In preliminary experiments, we found that the order of reconstruction does not significantly influence the performance of the representation. We use the reverse order here because it can be implemented more efficiently.

SSAE Training

Although all the parameters in the SSAE model can be trained simultaneously, we actually train our model using an iterative process consisting of two phases. In the first phase, the RNN encoder and decoder parameters are updated, while in the second phase, only the segmentation gate parameters are updated. The two phases are performed iteratively.

First Phase - RNN Encoder and Decoder In the first phase, we train only the RNN encoder and decoder to minimize reconstruction error while fixing the parameters of the

doi:10.6342/NTU202302463

segmentation gate. Because the segments are already provided by the segmentation gate, the first phase training is parallel to training a typical SA as in Subsection 3.3.2. That is, the encoder and decoder learn to minimize the reconstruction error $\mathcal{L}_{mse}$ in (3.16). Each time in phase one, the encoder and decoder are learned from random initialized parameters, instead of starting off with the parameters learned in the previous iteration – this was found to offer better training stability.

Second Phase – Segmentation Gate In the second phase, we update the parameters of the segmentation gate while fixing the parameters of the encoder and decoder. Although the encoder and decoder are not updated in this phase, they are involved in computing the reward for training the segmentation gates by reinforcement learning.

The segmentation gate is trained using reinforcement learning. After the gate performs the segmentation for each utterance, it receives a reward r and a reward baseline r_b for updating the parameters. r and r_b are defined later. We can express the expected reward for the gate under policy π as $J(\theta) = \mathbf{E}_{\pi}[r]$, where θ is the parameter set. To maximize the expected reward $J(\theta)$, policy gradient [159] is used to update the parameters of the segmentation gate using the parameter update formulation below.

$$\nabla_\theta J(\theta) = \mathbf{E}_{a \sim \pi}[\nabla_\theta (r - r_b) \sum_{t=1}^{T'} log \pi_t^{(\theta)}(a_t)], \tag{3.24}$$

where $\pi_t^{(\theta)}(a_t)$ is the probability for the action a_t taken per (3.22).

The reconstruction error is an effective indicator of whether the segmentation boundaries are good, since the embeddings are generated based on the segmentation. We hypothesize that good boundaries, for example those close to word boundaries, result in smaller reconstruction errors because the audio segments for words appear more frequently in the

doi:10.6342/NTU202302463

corpus and thus the embeddings are trained better with lower reconstruction errors. Therefore, one proper choice for the first term in the reward function may be r_{mse}, which is the negative reconstruction error, $r_{mse} = -\mathcal{L}_{mse}$.

At the same time, it is important to have a guideline for the proper number of segments N in an utterance of a given length T'. Without this guideline, the segmentation gate generates as many segments as possible in order to minimize the reconstruction error. Therefore, we design the reward such that the smaller number of segments N normalized by the utterance length T', the higher the reward:

$$r_{num} = -\frac{N}{T'}, \tag{3.25}$$

where N and T' are respectively the numbers of segments and frames for the utterance as in Figure 3.9.

The total reward r is obtained by choosing the minimum between r_{mse} and r_{num}:

$$r = min(r_{mse}, \lambda r_{num}) \tag{3.26}$$

where λ is a hyperparameter to be tuned for a reasonable guideline to estimate the proper number of segments for an utterance of length T. λ is determined to make the values of r_{mse} roughly equivalent to λr_{num}. r_{mse} and r_{num} are unknown before the model training, but it is possible to estimate their average values. Here we assume the average length of spoken words is known as the prior knowledge (this is the only language specific prior knowledge we used), so the average value of r_{num} can be roughly estimated. r_{mse} is estimated by randomly segmenting the utterances first and then training a sequence-to-sequence autoencoder. Interpolating r_{mse} and r_{num} as total reward r is also possible, but in our preliminary experiments, we found that the minimum function yielded better results than interpolation.

doi:10.6342/NTU202302463

For the reward baseline r_b, we further use an utterance-wise reward baseline to remove the bias between utterances. For each utterance, M different sets of segment boundaries are sampled by the segmentation gate, each of which is used to evaluate a reward r_m with (3.26). The reward baseline r_b for the utterance is then their average:

$$r_b = \frac{1}{M} \sum_{m=1}^{M} r_m. \tag{3.27}$$

3.3.5 Example Application: Unsupervised Query-by-Example Spoken Term Detection

Here we consider unsupervised query-by-example spoken term detection (QbE STD) as an example application to evaluate the quality of the embeddings. The task of unsupervised QbE STD here is to verify the existence of the input spoken query in an utterance or audio file without performing speech recognition [32]. With SSAE in Subsection 3.3.4, this is achieved as illustrated in Figure 3.10. Given the acoustic feature sequences of a spoken query and a spoken document, SSAE represents these sequences as embeddings, **q** = { q_1, q_2, ..., q_{n_q}} for the query and **d** = { d_1, d_2, ..., d_{n_d} } for the document. Here **q** and **d** are sequence z obtained by SSAE in Subsection 3.3.4 with a spoken query or a spoken document as input, respectively. With the embeddings, simple subsequence matching is used to evaluate the relevance score $S(\mathbf{q}, \mathbf{d})$ between **q** and **d**. First, we compute S_n for each position in **d**.

$$S_n = \prod_{m=1}^{n_q} Sim(q_m, d_{m+n-1}). \tag{3.28}$$

Cosine similarity can be used in the similarity measure in (3.28). As shown in the right part of Figure 3.10, S_1 = $sim(q_1, d_1) \cdot sim(q_2, d_2)$, S_2 = $sim(q_1, d_2) \cdot sim(q_2, d_3)$ and so on. The relevance score $S(\mathbf{q}, \mathbf{d})$ between the query **q** and document **d** is then the sum of

doi:10.6342/NTU202302463

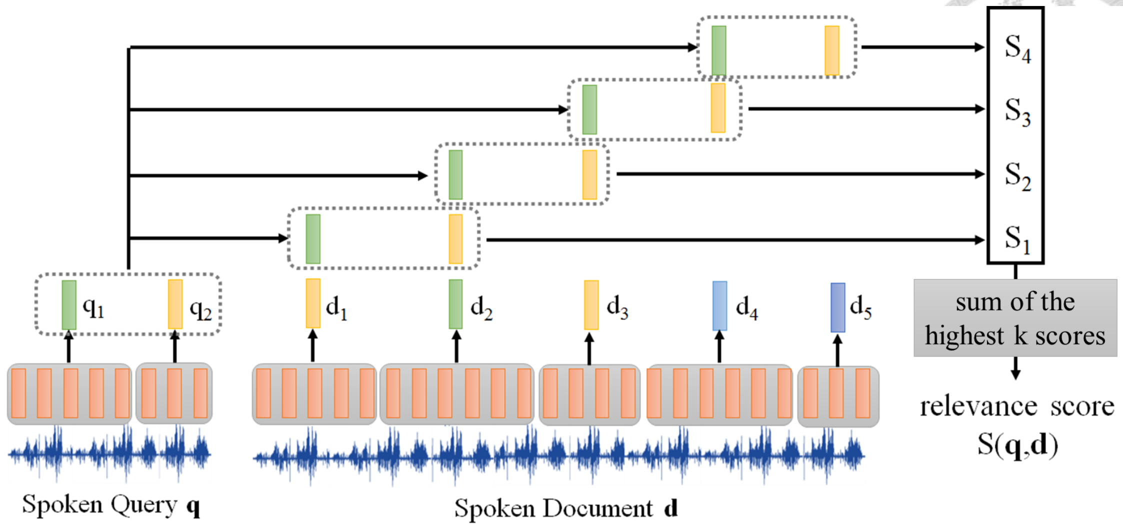


Figure 3.10: Unsupervised query-by-example spoken term detection (QbE STD) with SSAE.

the highest k scores among S_n obtained in (3.28).

3.3.6 Experiment: Audio Representation

Experimental Setup

In this subsection, we used Librispeech [137] as the speech corpus. The dataset was segmented according to the word boundaries obtained by forced alignment to the reference transcriptions. We used the 100 hours clean speech audio data set for model training. For evaluation, another 3000 utterances were used. Thirty-nine-dimension MFCCs were used as the acoustic features. The phonetic encoder was a 2-layer GRU with a 256-node hidden layer; the speaker encoder uses the same architecture. In the case without disentanglement, the RNN decoder was also a 2-layer GRU with a 256-node hidden layer. However, in the disentanglement model, because the RNN decoder takes as input the concatenation of the phonetic and speaker encoders, we set its size to 512. The speaker classifier was a

doi:10.6342/NTU202302463

fully-connected feedforward network with two 256-node hidden layers. The models were trained with the Adam optimizer with a batch size of 64. Algorithm 1's T_{dis} was set to 3.

Experimental Results

We trained the SA models on the training set to encode the segments in the evaluation set, which were never seen during training, and then computed the cosine similarity between each segment pair. We computed the average cosine similarity of the vector representations for each pair of audio segments in the testing set, and compared it with the phoneme sequence edit distance (PSED). Shown in Figure 3.11 are the average and variance (the length of the black line on each bar) of the cosine similarity for groups of pairs clustered by the PSED (PSED $= 0,1,2,3$ and > 3) between the two words.

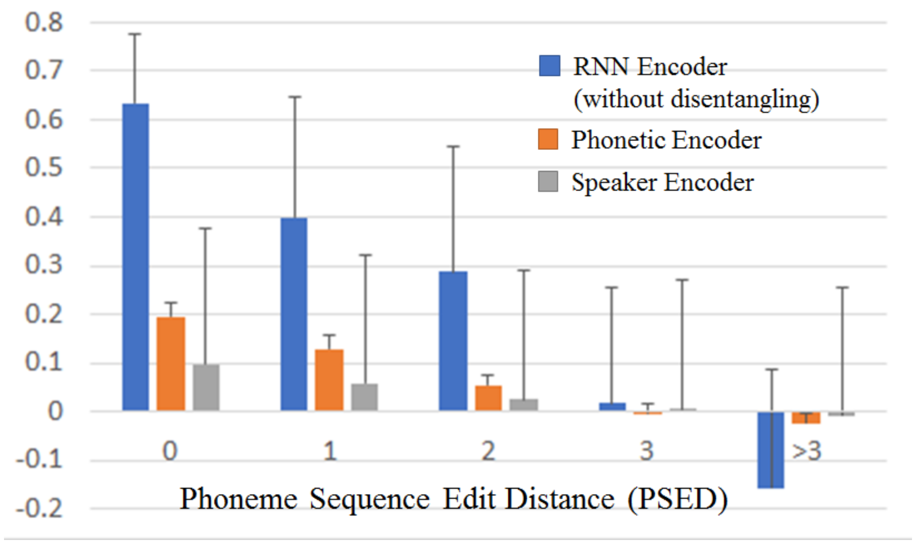


Figure 3.11: Average cosine similarity and variance (length of black line on each bar) between vector representations for all segment pairs in the evaluation set, clustered by phoneme sequence edit distance (PSED).

In Figure 3.11, the cosine similarities of the RNN encoder and phonetic encoder de-

doi:10.6342/NTU202302463

crease as the edit distances increase; that is, the vector representations for words with similar pronunciations are in close proximity to each other. This means that both the RNN encoder and the phonetic encoder indeed encode the sequential phonetic structures into fixed-length vectors. Clearly, the speaker encoder output includes little phonetic information because speaker encoder similarities are almost independent of the PSED. We also find from the means of similarities that the RNN encoder clearly distinguishes word segments with different phonemes even without disentangling features. For example, the similarity for word segments whose phonemes are exactly the same (PSED$= 0$) is 0.63, while the similarity for word segments with one different phoneme (PSED$= 1$) is only 0.40. However, their similarities have very large variances. For example, the variances of the group with one different phoneme is 0.24, which leads to ambiguity between different groups. This is reasonable because it is well-known that even with exactly identical phoneme sequences, acoustic realizations can differ greatly for different speakers. For the phonetic encoder, the mean similarities between different groups are not as remarkable as for the RNN encoder; however, the variances in each group are much smaller (0.018–0.029). This shows that disentangling features separate the values of the similarities of different groups.

In addition, we performed an ablation study to verify the effectiveness of different components in Figure 3.8 to help disentangle the features. Similar to the setting used before [144], we used ABX metric to evaluate the performance. In all triplet minimal pairs, only words with three phonemes were considered. A and X corresponded to the same text word, and A and B only differed in the central phoneme. For the within-speaker task, A, B and X belonged to the same speaker (e.g. $A = beg_{T1}, B = bag_{T1}, X = beg'_{T1}$); for the across-speaker task, A and B belonged to the same speaker, and X to a different

doi:10.6342/NTU202302463

Loss	Within Speaker	Across Speaker
Same as Alg. 1	**12.5**	**19.2**
No $\mathcal{L}_c$	14.0	19.8
No $\mathcal{L}_s$	13.1	20.2
No $\mathcal{L}_c$ & $\mathcal{L}_s$	15.2	20.0

Table 3.8: Within- and across-speaker ABX scores for the learned vector representations. Here an ablation study was performed by removing some loss terms in Algorithm 1, or some parts of Figure 3.8.

one (e.g. $A = beg_{T1}, B = bag_{T1}, X = beg_{T2}$). Then we calculated the cosine similarities of the (A, X) pair and (B, X) pair to obtain the discriminability. We converted the results into error rates listed in Table 3.8. The first row in the table shows the result of the original Algorithm 1, and the results in other rows reflect the degradation of performance because some loss terms in Algorithm 1 or components of Figure 3.8 were removed. The results clearly show that each of the components in Figure 3.8 is helpful for disentanglement of features.

In the following experiments, we further compare the performance of the RNN encoder without feature disentanglement and the phonetic encoder on QbE STD.

Visualization

In this subsection, we visualize the representations of the audio segments in the evaluation set for further analysis.

Figures 3.12 and 3.13 show the difference between the outputs of the phonetic and

doi:10.6342/NTU202302463

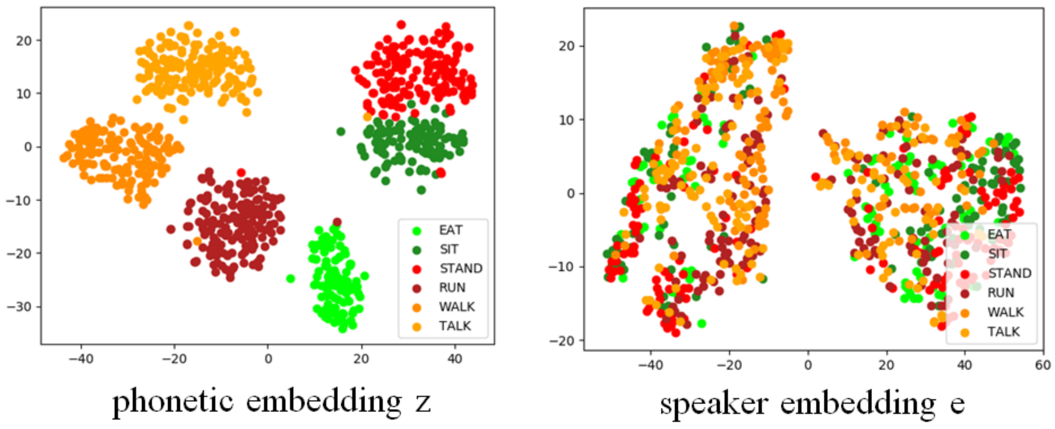


Figure 3.12: Output of phonetic encoder and speaker encoder for six different words.

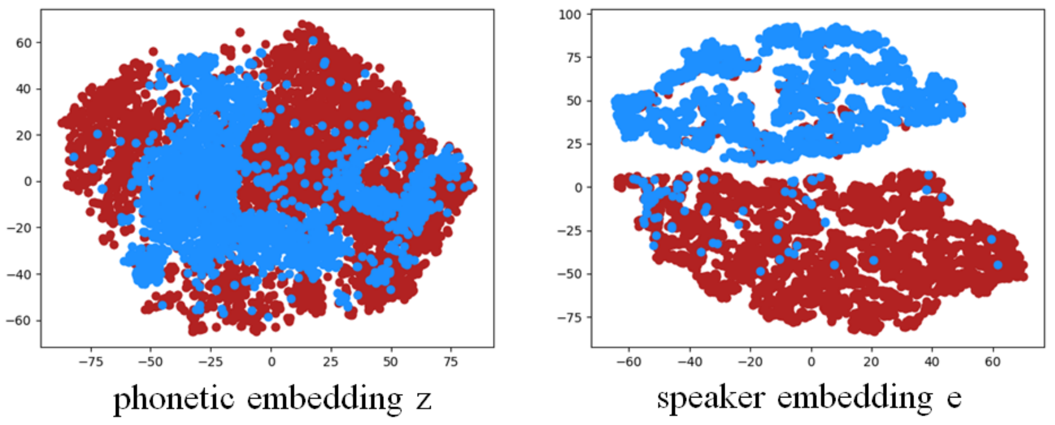


Figure 3.13: Output of phonetic encoder and speaker encoder for two different speakers.

speaker encoders. In Figure 3.12, we show the representations of the audio segments of six different words ("eat", "sit", "stand", "run", "walk", and "talk") from a number of male and female speakers. Each point in Figure 3.12 corresponds to a representation of an audio segment; different words are represented using different colors. The points in the left and right parts of Figure 3.12 are the representations of the same sets of segments but with different encoders. The left part is the output of the phonetic encoder z, while the right part is the output of the speaker encoder e. The representations were reduced to two

doi:10.6342/NTU202302463

dimensions using PCA. It is clear that the phonetic representations distinguish different words, while the speaker representations from the six words are mixed together. We also find that the speaker representations are clustered into two groups corresponding to males and females. The setup of Figure 3.13 is parallel to that of Figure 3.12; here we show the representations of the audio segments from two speakers. The segment representations of the two speakers correspond to the red and blue points. The phonetic representations do not distinguish the audio segments of the two speakers because their utterances show no remarkable differences. The audio segments of the two speakers, however, show very different speaker representations.

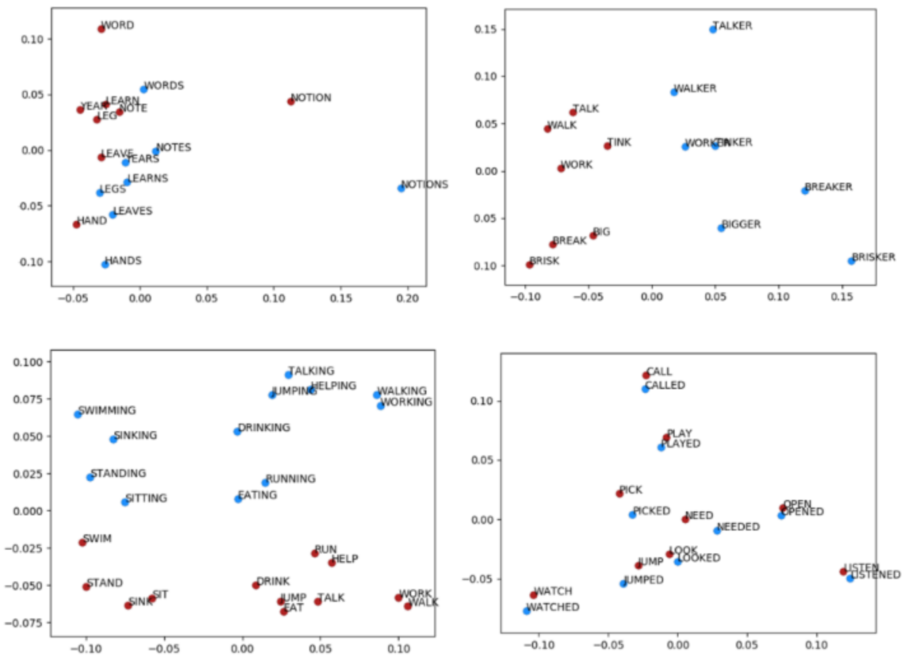


Figure 3.14: Averaged representations for four sets of words

For another test, we selected four sets of words that differ only in the last few phonemes. We averaged the phonetic representations z of the audio segments corresponding to the same word, and reduced the dimensionality of the averaged presentations to 2 using PCA. The averaged representation of word w is denoted as $V(w)$. From the results, shown in Fig-

doi:10.6342/NTU202302463

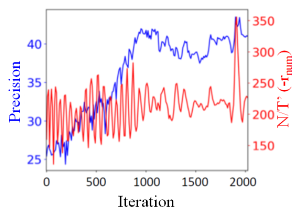
Loss	+ing	+ed	+s	+er
Same as Alg. 1	**0.09**	**0.26**	**0.13**	**0.09**
No $\mathcal{L}_c$ & $\mathcal{L}_s$	0.08	0.19	0.09	0.08

Table 3.9: Mean reciprocal rank (MRR) scores for retrieval results of vector representations of words plus four kinds of suffixes.

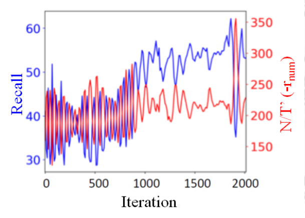
ure 3.14, we see that the representations z constitute very good descriptions for the sequential phonemic structures of the acoustic segments. For example, in the leftmost figure of Figure 3.14, we observe that $V(SIT) - V(SITTING) \approx V(STAND) - V(STANDING)$. Several similar examples are found in Figure 3.14.

For quantitative analysis, we conducted an experiment to evaluate whether the difference vector between the phonetic representation of a certain word w and that of the word plus a suffix, such as $w - ing$, would be consistent regardless of what w was. More specifically, for a certain pair ($V(w_1)$, $V(w_2)$, we calculated $V(w_2) + V(w_1 - ing) - V(w_1)$, and used mean reciprocal rank (MRR) (harmonic mean of the retrieval ranks) to serve as the retrieval evaluation measure of $V(w_2 - ing)$. The retrieval results of four sets of words plus suffixes ($w - ing$, $w - ed$, $w - s$ and $w - er$) are listed in Table 3.9. Here we also compared the performance of representations with or without disentanglement. It can be observed that the difference vectors mentioned above were consistent to an extent, and again disentanglement of features improved the retrieval results.

doi:10.6342/NTU202302463

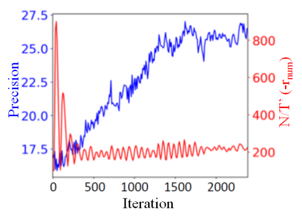


(A) Precision

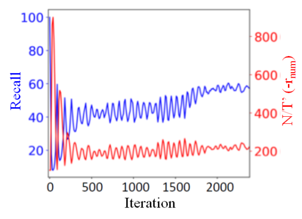


(B) Recall

Figure 3.15: SSAE learning curves on TIMIT validation set. Red curves are for N/T' ($-r_{num}$ in (3.25)), where N is the number of segments, and T' is the number of acoustic features. Blue curves are (A) precision and (B) recall.



(A) Precision



(B) Recall

Figure 3.16: SSAE learning curves on Czech validation set.

3.3.7 Experiment: Segmentation

Experimental Setup

We conducted segmentation experiments on TIMIT and GlobalPhone [160], specifically Czech, French, and German. For TIMIT the ground truth word boundaries were provided, while for GlobalPhone we used the forced-aligned word boundaries. In this subsection, both the RNN encoder and decoder of the SSAE consist of one hidden layer with 100 LSTM units. Feature disentanglement does not apply in the experiments of this subsection. That is, here we do not have a speaker encoder and speaker classifier. The segmentation gate consists of two 256-node LSTM layers. All parameters were trained with

doi:10.6342/NTU202302463

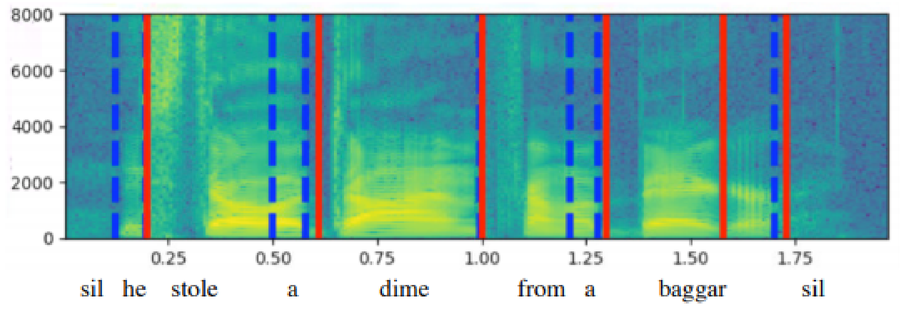


Figure 3.17: An example of segmentation by SSAE. The blue dashed lines were the oracle boundaries and the red lines were produced by SSAE.

Adam [161]. We set M in (3.27) for estimating the reward baseline r_b to be 5, and $\lambda = 5$ in (3.26). The word boundaries were initialized randomly. The proximal policy optimization algorithm [162] was used in the policy gradient. The tolerance window for word segmentation evaluation was taken as 40 ms. The acoustic features used were 39-dim MFCCs with utterance-wise CMVN.

Experimental Results

Figure 3.15 shows the SSAE learning curves on the TIMIT validation set. Figure 3.16 is the results for the Czech validation set; we do not show the French and German results because their trends mirror that of Figure 3.16. We see that SSAE gradually learns to segment utterances into spoken words because both the precision and recall (blue curves in Figure 3.15 and Figure 3.16 respectively) increase during training. The reward r_{num} in (3.25) (red curves) fluctuates initially during training and tends to converge at the end.

Table 3.10 shows the spoken word segmentation performance of the proposed SSAE in terms of precision, recall, and F1 score. We compared the SSAE results with three baselines: random segments, gate activation signals (GAS) [149], and hierarchical agglomer-

doi:10.6342/NTU202302463

ative clustering (HAC) [151, 163]. We observe that SSAE significantly outperforms the other baselines on all languages other than German GAS, to which it is comparable. An example of segmentation by SSAE is shown in Figure 3.17.

Method	TIMIT			Czech	French	German
	Precision	Recall	F1	F1		
Random	24.60	41.08	30.77	22.56	32.66	25.41
HAC	26.84	46.21	33.96	30.84	33.75	27.09
GAS	33.22	52.39	40.66	29.53	31.11	32.89
SSAE	37.06	51.55	43.12	37.78	48.14	31.69

Table 3.10: Spoken word segmentation performance compared to different methods.

3.3.8 Experiment: QbyE STD

In this subsection, we evaluate the performance of audio word2vec on QbyE STD. We use mean average precision (MAP) as the evaluation measure.

The first set of experiment is conducted on English (TIMIT), Czech, French and German. The testing set utterances were used as spoken documents [148]. We randomly selected as the query words five words for each language containing a variety of phonemes; from the training set we used several occurrences of each of these phoneme-rich words as spoken queries. For English, Czech, French, and German, we used 29, 21, 25, and 23 spoken queries for evaluation, respectively. The number of the highest scores among S_n obtained in (3.28), k, in Subsection 3.3.5 and Figure 3.10 was set to be 1.

We compared the quality of the SSAE embeddings with other kinds of audio word2vec

doi:10.6342/NTU202302463

			Embeddings (different seg.)			
Lang.	Ran.	DTW	GAS	HAC	SSAE	Oracle
TIMIT	0.74	12.02	8.29	0.91	23.27	30.28
Czech	0.38	16.59	0.68	1.13	19.41	22.56
French	0.27	11.72	0.40	0.92	21.70	29.66
German	0.18	6.07	0.27	0.26	13.82	21.52

Table 3.11: Spoken term detection performance in mean average precision (MAP) for proposed SSAE as compared to audio word2vec embeddings trained with spoken words segmented with other methods for different languages. Random baseline (Ran.) assigns a random score to each query-document pair. Standard frame-based DTW is the primary baseline; oracle segmentation is the upper bound.

embeddings trained with the segments generated using the different segmentation methods. Although the spoken queries are single words, the models do not know this. The spoken queries are also segmented into segments using different segmentation approaches. The results are listed in Table 3.11. The performance for embeddings trained with the ground truth word boundaries (oracle) in the last column serves as the upper bound. For the random baseline in the first column, a random score was assigned to each query-document pair. In the second column we also report the performance of standard frame-based dynamic time warping (DTW) [148] as a primary baseline. From the table we observe that the oracle method outperforms all other methods on all corpora. SSAE outperforms the DTW baseline because DTW cannot identify spoken words if the speaker or gender characteristics are very different; such varying signal characteristics are better absorbed in the

doi:10.6342/NTU202302463

audio word2vec training. These experimental results confirm that the SSAE embeddings do carry the sequential phonetic structure information from the utterances, leading to the better STD performance. In most cases, the performance of the GAS and HAC embeddings are not too far from random. It appears that audio word2vec does not train well if the performance of spoken word segmentation does not exceed some minimum level. That is, spoken word segmentation boundaries made the biggest impact on STD performance. We also note that although the GAS segmentation performance was slightly better than that of SSAE for German, SSAE clearly outperformed GAS on German QbyE STD.

Training Set	Testing Set	$k = 1$		$k = 40$	
		$NoDis$	PE	$NoDis$	PE
Oracle	SSAE	16.41	17.24	17.27	21.79
SSAE	SSAE	15.54	17.09	17.60	19.60

Table 3.12: MAP performance of query-by-example spoken term detection (QbE STD). $NoDis$ and PE are the results without feature disentanglement and using the phonetic encoder respectively. The segmentation of training and testing sets can be either oracle or by SSAE. Different k for the search algorithm in Subsection 3.3.5 and Figure 3.10 are tested.

Then we conducted experiments with more spoken queries based on Librispeech. The audio word2vec models were trained on the 100 hour clean data set. The spoken archive to be retrieved is the clean testing data set. The chapters are considered as the unit to be retrieved. We have 361 spoken queries also from Librispeech, but not included in the training set or retrieved utterances. All the spoken queries correspond to a single word in

doi:10.6342/NTU202302463

the experiments.

The experimental results are shown in Table 3.12. $NoDis$ and PE are the results without feature disentanglement and using the phonetic encoder respectively. Oracle means we segmented the audio using the word boundaries obtained by forced alignment with the reference transcriptions. SSAE in Table 3.12 means the segments were obtained by SSAE. Different k ($k = 1$ or 40) for the search algorithm in Subsection 3.3.5 and Figure 3.10 are tested. Clearly, the output of the phonetic encoder outperformed the features without disentanglement because it reduces the speaker dependence (PE v.s. $NoDis$) .

3.3.9 Concluding Remarks

In this section, we extend the research of audio word2vec. We use domain adversarial training to automatically learn encoders that encode different information. The experimental results show that this thus disentangles the phonetic and speaker information. We further propose an SSAE trained with reinforcement learning, in which word-level segmentation and segment representation are jointly learned.

Audio embedding has many possible applications beyond STD. For example, audio embedding can be considered as better audio representation for speech recognition. It is possible to use a large amount of unlabeled audio to learn the embeddings to improve low-resource speech recognition. The learned embeddings can also be used in the applications related to spoken language understanding of low-resource language like spoken question answering, spoken content summarization, and speech translation. These spoken language understanding systems can take the learned audio embedding as input, instead of the transcriptions of spoken content.

doi:10.6342/NTU202302463

Chapter 4 Multi-task Learning and Universal Modeling of Speech Processing Tasks

In this chapter, we study multi-task learning and universal modeling of various speech processing tasks [164, 165, 166].

4.1 Speech Representation Learning Through Self-supervised Pretraining And Multi-task Finetuning

Recently many SSL approaches have been proposed for pretraining models for speech processing tasks [54, 56, 57, 59, 60, 62, 63, 64, 65, 66, 67]. After a shared model is pretrained with SSL to extract general representations, it can then be specialized on downstream tasks with task-specific head models and simple finetuning. This method achieves state-of-the-art performance in many applications.

To fairly evaluate the generalizability of SSL approaches without further heavy downstream task-specific finetuning, the SUPERB benchmark [53] is proposed to measure the performance of a shared model across a wide range of discriminative speech processing tasks without heavy finetuning.

Supervised multi-task learning (MTL) is to train a shared model on various downstream tasks [167, 168, 169, 170, 171]. This section wants to investigate if MTL on various downstream tasks can further improve the representations from SSL. In CV [172, 173] and NLP [86, 174], general models trained by MTL approaches can be evaluated on benchmarks that include various tasks. However, in speech, there has not been a systematic

doi:10.6342/NTU202302463

study of general representation learning models trained by MTL of various speech processing tasks.

In this section, we investigate two MTL training scenarios and also one task transfer learning scenario. For the two MTL scenarios, we select a state-of-the-art SSL pretrained shared model in the SUPERB benchmark as the starting point for MTL. Then we train the shared model with MTL in two different scenarios:

- All-task MTL Finetuning: Finetune the SSL pretrained shared model with all tasks in SUPERB. It serves as a strong baseline for SSL approaches and the following scenarios.
- Leave-one-out MTL Finetuning: Finetune the SSL pretrained shared model with all but one tasks in SUPERB. We can observe the influence of removing one task on the learned representations and their performance on the other tasks.

To further examine if the representations learned with supervised MTL can generalize to an unseen new task, we have an additional Task Transfer Learning scenario.

- We take a shared model from the Leave-one-out MTL scenario and freeze its parameters. We extract the representations with this shared model for training the prediction head of the remaining task that is not involved in MTL finetuning.

Through the different training scenarios above, we perform a preliminary study of the generalizability of representation learning by MTL of various discriminative speech processing tasks on a standard benchmark.

doi:10.6342/NTU202302463

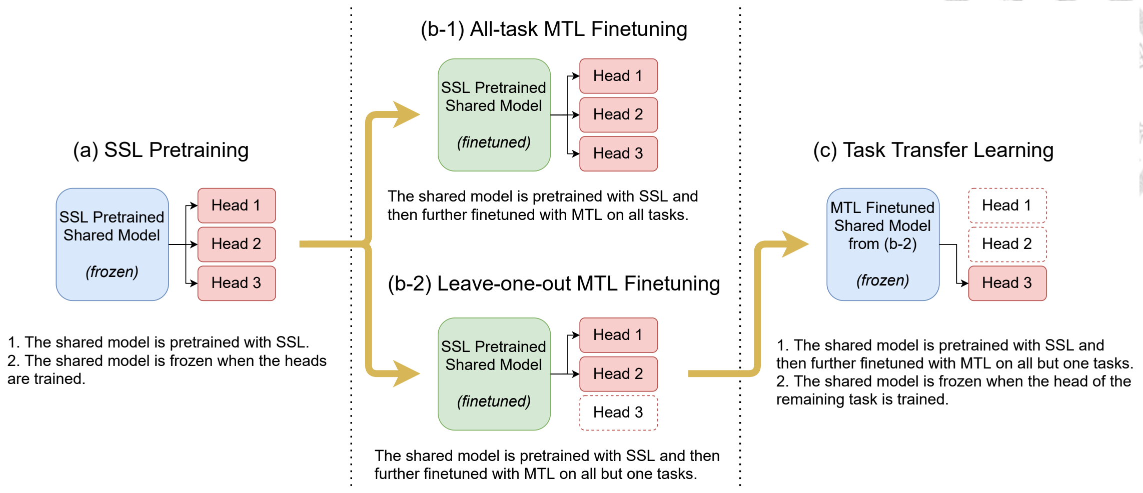

(a) SSL Pretraining
SSL Pretrained Shared Model (frozen)
Head 1
Head 2
Head 3
1. The shared model is pretrained with SSL.
2. The shared model is frozen when the heads are trained.
(b-1) All-task MTL Finetuning
SSL Pretrained Shared Model (finetuned)
Head 1
Head 2
Head 3
The shared model is pretrained with SSL and then further finetuned with MTL on all tasks.
(b-2) Leave-one-out MTL Finetuning
SSL Pretrained Shared Model (finetuned)
Head 1
Head 2
Head 3
The shared model is pretrained with SSL and then further finetuned with MTL on all but one tasks.
(c) Task Transfer Learning
MTL Finetuned Shared Model from (b-2) (frozen)
Head 1
Head 2
Head 3
1. The shared model is pretrained with SSL and then further finetuned with MTL on all but one tasks.
2. The shared model is frozen when the head of the remaining task is trained.

Figure 4.1: Four different training scenarios. Scenarios (b-1) and (b-2) require the pretrained shared model from (a). Scenario (c) requires the finetuned shared model from (b-2). The parameters of the shared model in scenarios (a) and (c) are frozen.

4.1.1 Training Scenarios

In this subsection, we describe four related training scenarios in the following subsections: SSL Pretraining, All-task MTL Finetuning, Leave-one-out MTL Finetuning, and Task Transfer Learning. The two MTL finetuning scenarios require the SSL pretrained shared model, and the Task Transfer Learning scenario requires the shared model from the Leave-one-out MTL Finetuning scenario.

SSL Pretraining

Many SSL approaches are evaluated and compared on the ten tasks in SUPERB [53]. For each SSL approach, we first pretrain a model with SSL objectives. Then we use this pretrained model as the shared model to extract representations for all downstream tasks. The parameters of the pretrained model are frozen. Then we train each task-specific prediction head (small downstream model) with the fixed representations, as illustrated in Figure 4.1

doi:10.6342/NTU202302463

(a).

All-task MTL Finetuning

We take the shared model pretrained with SSL as the starting point. Then we further finetune the shared model with MTL by jointly training it with downstream task-specific heads of all tasks in SUPERB, as illustrated in Figure 4.1 (b-1). In this way, the shared model can be updated by the gradients of all tasks to fit the respective objectives of each task. Therefore, the representations extracted from the shared model can perform well on the tasks involved in MTL. It serves as a strong baseline for SSL approaches and the following scenarios. To further examine the generalizability of representation learning by supervised MTL, we have two additional training scenarios below.

Leave-one-out MTL Finetuning

Similarly, we take the shared model pretrained with SSL as the starting point. Then we further finetune the shared model with MTL by jointly training it with downstream task-specific heads of **all but one** tasks in SUPERB, as illustrated in Figure 4.1 (b-2). Compared to the finetuned shared model with All-task MTL Finetuning, we can observe the influence of removing one task on the learned representations and their performance on the other tasks.

Task Transfer Learning

We take the finetuned shared model with Leave-one-out MTL Finetuning as the pretrained shared model in this scenario. Then we freeze the shared model, and only train the downstream head model of the remaining task that is not used in MTL finetuning, as illustrated

doi:10.6342/NTU202302463

in Figure 4.1 (c).

Comparing the representations in this scenario with those learned with only SSL approaches, we can observe the generalizability of the representations learned with MTL on a new task compared to SSL only. On the other hand, in comparison with the representations learned with All-task MTL Finetuning, we can observe how the performance of a task is influenced if this task is not involved in the MTL finetuning.

4.1.2 Experimental Setup

Tasks In SUPERB

Ten tasks in SUPERB can be used to investigate four aspects of speech: **content** (Phoneme Recognition (PR), Automatic Speech Recognition (ASR), Keyword Spotting (KS), and Query by Example Spoken Term Detection (QbE)), **speaker** (Speaker Identification (SID), Automatic Speaker Verification (ASV), and Speaker Diarization (SD)), **semantics** (Intent Classification (IC) and Slot Filling (SF)), and **paralinguistics** (Emotion Recognition (ER)). Since no downstream model training is required in QbE, we only perform MTL experiments and compare the results on the other nine tasks.

- **PR** converts an utterance into a sequence of phonemes. Alignment modeling is included in the PR task to avoid the potential inaccurate forced alignment. The evaluation metric is the phone error rate (PER).
- **ASR** transcribes an utterance into a sequence of words. While PR analyzes the performance of modeling phonetics, ASR reflects the performance of recognizing more common text units in a real-world scenario. The evaluation metric is the word error rate (WER).

doi:10.6342/NTU202302463

- **KS** identifies preregistered keywords in an utterance by classifying the utterance into a predefined set of words. The task is important for on-device speech processing and requires low response time. The evaluation metric is the accuracy (ACC).

- **SID** classifies the speaker identity of an utterance in a multi-class classification setting, where the set of speakers are the same for both training and testing. The evaluation metric is the accuracy (ACC).

- **ASV** verifies whether the speakers of a pair of utterances match in a binary classification setting. Different from SID, the speakers in the testing set may not appear in the training set. Therefore, ASV is more challenging than SID. The evaluation metric is the equal error rate (EER).

- **SD** segments an utterance and classifies the segments into speaker identities, i.e., *who is speaking when*. Multiple speakers can speak simultaneously. Rich and various speaker characteristics should be encoded in the extracted representations for each frame to represent mixtures of signals. The evaluation metric is the diarization error rate (DER).

- **IC** classifies an utterance into predefined classes of speaker intents. The evaluation metric is the accuracy (ACC).

- **SF** converts an utterance into a sequence of semantic slot-type classes. For example, *FromLocation* can be a slot-type for a spoken word *Taipei*, which is known as a slot-value. Both slot-types and slot-values are essential for an SLU system. Therefore, we use two evaluation metrics for slot-types and slot-values respectively: the slot-type F1 score (F1) and the slot-value character error rate (CER).

doi:10.6342/NTU202302463

- **ER** predicts an emotion class for each utterance. The evaluation metric is accuracy (ACC).

As for the datasets and splits used for each task, we follow the original settings in SUPERB (PR [137], ASR [137], KS [175], SID [176], ASV [176], SD [177], IC [178], SF [179], and ER [180]).

The SSL Pretraining Approach In Experiments

Many SSL approaches are evaluated and compared on the tasks in SUPERB. Among them, HuBERT [65] achieves the overall best performance. Therefore, we select HuBERT as the SSL pretraining approach across all of our experiments.

HuBERT utilizes an offline clustering algorithm on hidden representations to provide aligned target labels for a BERT-like [109] prediction. The clustered labels at the masked locations serve as the prediction targets. We use a weighted sum of hidden representations of all layers in the HuBERT model as the representations for downstream heads, as in SUPERB.

Model Architecture and Implementation Details

Since MTL requires more computational resources than single-task training, we adopt HuBERT Base rather than Large in SUPERB as our shared model architecture. For task-specific head architectures, we simply follow the settings in SUPERB. We use a batch size of 2 for two MTL finetuning training scenarios, and a batch size of 8 for downstream head training in the Task Transfer Learning scenario. Each model is trained with an Adam optimizer with a linearly warmup learning rate from 0 to 1e-5 for the first 5000 steps and then

doi:10.6342/NTU202302463

Table 4.1: Experimental Results of training scenarios described in Subsection 4.1.1. We have the numbers in Scenario (b-2) be bold or underlined if they are better or worse than (b-1) respectively.

Scenario	Tasks for MTL	ASR	PR	SF		SD	ER	IC	KS	ASV	SID
	Finetuning	**WER↓**	**PER↓**	**F1↑**	**CER↓**	**DER↓**	**ACC↑**	**ACC↑**	**ACC↑**	**EER↓**	**ACC↑**
(a) SSL	N/A	6.42	5.41	88.53	25.20	5.88	64.24	98.34	96.30	5.11	81.42
(b-1) SSL+MTL	all	6.22	3.61	87.56	26.76	4.93	67.28	99.60	97.34	6.76	90.86
(b-2): SSL+MTL	all but ASR	X	3.63	87.28	27.11	**4.89**	65.07	**99.63**	**97.57**	7.78	90.69
	all but PR	6.79	X	86.94	27.66	**4.81**	66.73	**99.66**	**97.44**	7.94	**91.16**
	all but SF	**6.10**	**3.39**	X	X	**4.73**	65.71	99.58	97.18	7.61	90.70
	all but SD	6.28	**3.54**	**87.94**	**26.31**	X	66.73	**99.63**	97.11	7.49	90.79
	all but ER	**6.17**	**3.40**	87.45	26.90	**4.77**	X	99.55	97.27	7.19	90.51
	all but IC	**6.13**	**3.34**	**87.65**	26.94	**4.78**	66.08	X	97.27	**6.74**	90.55
	all but KS	**6.17**	**3.55**	**87.83**	26.88	**4.91**	66.27	**99.71**	X	7.86	90.67
	all but ASV	**5.90**	**2.79**	**87.88**	**26.52**	**3.61**	64.88	99.58	**97.44**	X	85.06
	all but SID	**5.95**	**3.25**	87.33	27.39	**4.50**	**68.66**	99.55	97.27	9.00	X
(c) Task Transfer	N/A	6.27	5.79	88.14	26.24	5.80	64.24	97.42	96.33	7.55	62.05

a linearly decaying learning rate to 0 for 195000 steps.

4.1.3 Experimental Results

All experimental results are presented in Table 4.1. An up-/down-arrow beside an evaluation metric means that better performance results in a higher/lower number of that metric. The results are grouped according to four different training scenarios corresponding to the subsections in Subsection 4.1.1 respectively.

doi:10.6342/NTU202302463

The Performance Of All-task MTL Finetuning

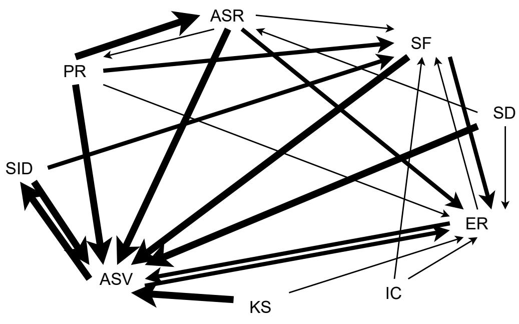
From the comparison between Scenario (a) and Scenario (b-1), we observe that the performance of the shared model finetuned with All-task MTL is better in the tasks ASR, PR, SD, ER, IC, KS, SID, and worse only in the tasks SF and ASV than SSL pretraining. It indicates that MTL is a strong baseline for SSL pretraining or other representation learning approaches.

One thing worth noting is that all tasks except for ASV do not suffer from overfitting in terms of the validation scores during training. However, the model is prone to overfit on ASV. In Scenario (a), the model checkpoint of a downstream head can be determined by the best validation score during training for each task respectively. Yet in Scenario (b-1), since the shared model is jointly trained with the downstream heads of all tasks, it is hard to select the model checkpoint based on the validation scores of all tasks. In this section, we simply select the last model checkpoint after 200,000 training steps for all tasks. It may be a reason for the worse performance of MTL finetuning on ASV. We leave exploring a better method to select the model checkpoint with MTL in future work.

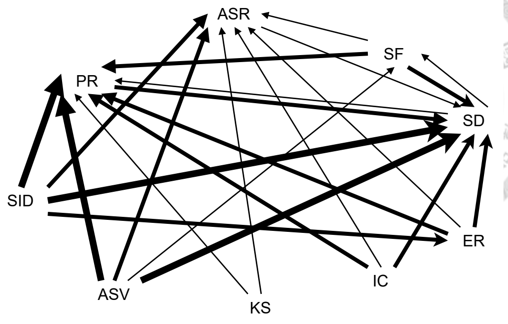
The Influence Of Removing One Task In MTL Finetuning

The rows in Scenario (b-2) show the results of finetuning the shared model with Leave-one-out MTL. To compare these results with Scenario (b-1) more clearly, we have the numbers in Scenario (b-2) be bold or underlined if they are better or worse than (b-1) respectively. Furthermore, we calculate the relative score increases/decreases of Scenario (b-2) compared to Scenario (b-1), and plot two relation graphs in Figure 4.2 accordingly. For SF, we use CER to plot the edges because the relative changes are small in terms of

doi:10.6342/NTU202302463



The improvement relations of tasks.



The hurt relations of tasks.

Figure 4.2: Two relation graphs of tasks. If a task A performs worse/better after removing a task B in MTL, we connect an edge from B to A in the improvement/hurt graphs, indicating B can improves/hurts the performance of A. The width of an edge is thin/medium/thick if the relative score change is in the range [0.5%, 2%), [2%, 8%), or larger than 8%. If the the relative score change is less than 0.5%, we consider it negligible and no edge is connected.

F1. If a task A improves/regresses after removing another task B in MTL, it means that B can hurt/help A in MTL.

- For **ASR**, PR is an important auxiliary task for ASR, and SD helps a little. ASR performs better after removing any of the other tasks.
- For **PR**, ASR helps in MTL while the other tasks hurt the performance of PR. We can observe that content recognition tasks such as ASR and PR are hurt the most by speaker recognition tasks such as ASV and SID.
- For **SF**, ASR, PR, ER, and SID help in MTL. IC and KS help a little in terms of F1 but hurt in terms of CER. SD and ASV hurt SF.
- For **SD**, all of the other tasks hurt SD. SD is also hurt the most by speaker recognition tasks such as ASV and SID. Although SD, ASV and SID are all related to speaker

doi:10.6342/NTU202302463

characteristics, SID and ASV focus on the utterance-level embedding, while SD aims to distinguish frame-level speaker characteristics. Therefore, the fine-grained information needed by SD may be lost when jointly trained with ASV or SID.

- For **ER**, all of the other tasks except for SID help ER.
- For **IC** and **KS**, the influences of the other tasks are negligible.
- For **ASV**, all of the other tasks except for IC help ASV. As discussed previously, ASV suffers from severe overfitting. Therefore, jointly learning ASV with other tasks can mitigate this issue, especially with SID.
- For **SID**, all of the other tasks except for PR help SID. ASV especially helps a lot while the others help a little.

From another perspective, the results may provide a different insight in addition to the evaluation of supervised MTL as a representation learning. If we focus on a certain primary task, we may select proper auxiliary tasks to assist the primary task based on these MTL experimental results. For example, if we want to have a better performance on ER, we can jointly train ER with all of the other tasks except for SID.

The transitive relations of tasks in MTL need to be further verified. For example, suppose we know jointly training a task A with another task B can improve the performance of A; and we also know jointly training A with another task C can improve the performance of A. Yet, we cannot conclude that jointly training A with B and C simultaneously can improve A's performance. We leave more in-depth research of MTL and its optimization on speech processing tasks in future work.

doi:10.6342/NTU202302463

The Performance Of Task Transfer Learning

To further examine the generalizability of representation learning by MTL, we present the results of the Task Transfer Learning scenario in Table 4.1 (c). Compared to Table 4.1 (a), all of the tasks except for ASR and KS in Scenario (c) perform worse. It indicates SSL pretraining is still a more generalizable representation learning approach for a new downstream task. On the other hand, compared to Table 4.1 (b-1), all of the tasks except for SF in Scenario (c) perform worse. It indicates whether a task is involved in MTL is crucial to the performance of this task.

To obtain better generalizability, it is worth trying to train the shared model with both SSL and MTL simultaneously as semi-supervised MTL representation learning. We leave this exploration in future work.

4.1.4 Conclusion And Discussion

In this section, we investigate different training scenarios of supervised MTL as a speech representation learning approach along with SSL pretraining on a benchmark with various discriminative speech processing tasks. We analyze the generalizability of representations learned with supervised MTL empirically.

This section is only a preliminary study of MTL with various discriminative speech processing tasks. The performance of MTL is dependent on many factors, such as the amount of data, task relationships, noise and so on. These factors should be isolated and investigated with more theoretical analyses and empirical experiments in the future.

In the next section, we further propose a universal modularized model for not only discriminative but also generative speech processing tasks.

doi:10.6342/NTU202302463

4.2 SpeechNet: A Universal Modularized Model for Speech Processing Tasks

In the previous section, we explore multi-task learning of discriminative speech processing tasks, where input speech is encoded into representations by a shared model for various discriminative downstream tasks. In this section, we propose a more flexible universal model for various discriminative and generative speech processing tasks, where the task input or output can be either speech or text.

There is a wide variety of speech processing tasks, for example, automatic speech recognition (ASR), speech enhancement (SE), speaker classification (SC), text-to-speech (TTS) synthesis, and voice conversion (VC), etc. These tasks involve different capabilities related to speech processing ranging from extracting content information from speech signals to generating speech signals. In literature, model networks are usually designed and tuned separately for different tasks, and each aims to expert a specific ability for processing speech. However, when we only focus on one task, we may ignore some useful abilities that can be shared across tasks to make the tasks better. Human can learn different speech tasks and transfer the knowledge of different abilities between tasks. Can we train a universal model that can learn all the different speech processing abilities jointly in one model?

In this section [165], we propose SpeechNet, a universal model for various speech processing tasks. Inspired by T5 [87], we treat all speech processing tasks as the format: a task that takes speech/text input and produces speech/text output. In SpeechNet, there are basic modules to handle different modalities, as illustrated in Figure 4.3 and introduced

doi:10.6342/NTU202302463

below:

- Speech input: We use Prosody Encoder, Speaker Encoder and Content Encoder to extract prosody, speaker and content embeddings from speech.
- Speech output: We use Audio Decoder to synthesize audio.
- Text input: We use Text Encoder to map the input text to the content embedding space (which is the same output space of Content Encoder).
- Text output: We use Text Decoder to produce text according to content embedding.

Most of the speech processing tasks can be done by concatenating the modules above, making multi-task learning (MTL) for a wide variety of speech processing tasks possible. It has been shown that the universal models trained to solve multiple tasks can benefit from multi-task learning (MTL) [181, 182, 183], improving the generalizability and performance of models in NLP and computer vision. During MTL, the tasks share the same modules, and the gradients computed from different objective functions of these tasks are accumulated to update the shared modules.

This section shows that SpeechNet can simultaneously learn five common and important speech processing tasks: ASR, SE, SC, TTS, and VC. We conduct experiments with commonly used datasets for the five tasks. Based on the results of MTL, we know which combinations of speech tasks are effective. SpeechNet is modularized and flexible for incorporating more modules, tasks, and training criteria in the future.

doi:10.6342/NTU202302463

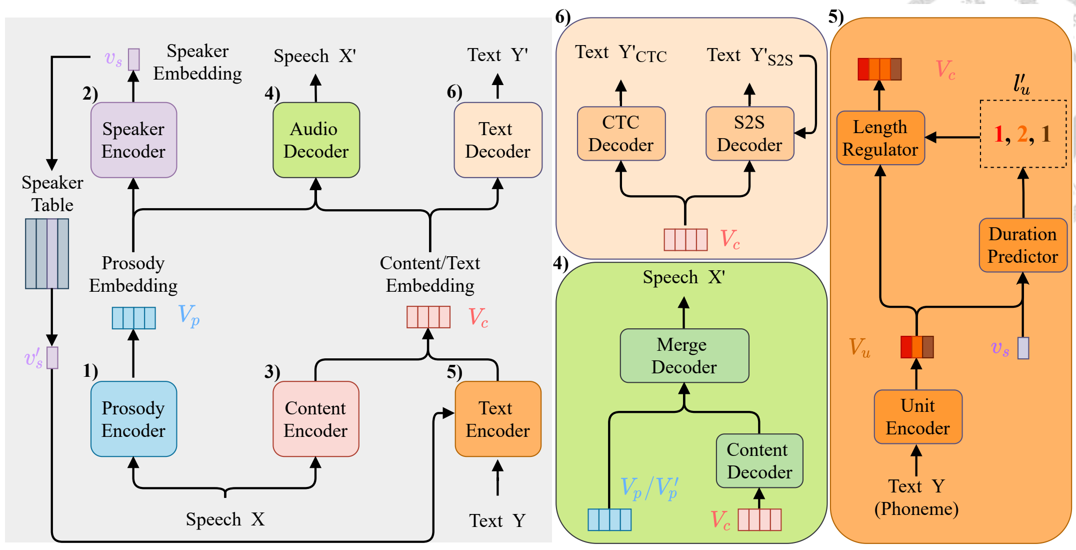


Figure 4.3: The architecture of SpeechNet is on the left. The six core modules in SpeechNet are Prosody Encoder, Speaker Encoder, Content Encoder, Audio Decoder, Text Encoder, and Text Decoder. More details about Audio Decoder, Text Encoder and Text Decoder are shown in a corresponding block on the right, with the same index and the same background color as that on the left of the figure.

4.2.1 SpeechNet: a universal modularized model for speech processing tasks

Any speech processing task can be treated as taking speech or text as input or output. SpeechNet contains six core modules for speech and text, respectively, to handle different modalities. In this subsection, the six modules are introduced. How to concatenate the modules to do the five tasks used in this section is described. At the end of this subsection, we discuss two problems of TTS and VC in this framework and further propose a modified version of SpeechNet by adding one additional module, Prosody Predictor.

doi:10.6342/NTU202302463

Modules in SpeechNet

Here we discuss the detailed formulation of each module presented in Figure 4.3 respectively, while describing the model architecture details in Subsection 4.2.2.

Prosody Encoder: E_P When speech $\mathbf{X} = \{\mathbf{x_1}, ..., \mathbf{x_T}\}$ with frame length T serves as the input, it can be passed through Prosody Encoder E_P to obtain a frame-level prosody embedding vector sequence

$$\mathbf{V_p} = E_P(\mathbf{X}), \tag{4.1}$$

Here we want to encode all the speaker and prosody characteristics in speech into $\mathbf{V_p}$.

Speaker Encoder: E_S The prosody embedding vectors can be further passed through Speaker Encoder E_S to obtain an utterance-level speaker embedding vector

$$\mathbf{v_s} = E_S(\mathbf{V_p}), \tag{4.2}$$

which represents the speaker characteristics in speech.

Content Encoder: E_C The speech input can also be passed through Content Encoder E_C to get a sequence of frame-level content embedding vectors

$$\mathbf{V_c} = \{\mathbf{v_{c_1}}, ..., \mathbf{v_{c_{T'}}}\} = E_C(\mathbf{X}), \tag{4.3}$$

which contain content information in speech. T' is the length of content embedding vectors, which can be equal to the original frame length of speech T or be shorter for more compact embeddings.

doi:10.6342/NTU202302463

Audio Decoder: D_A When the output is speech, Audio Decoder D_A takes a sequence of prosody embeddings $\mathbf{V_p}$ and a sequence of content embeddings $\mathbf{V_c}$ as input and output the desired speech. Specifically, the content embedding is firstly transformed by Content Decoder D_C, concatenated with the prosody embeddings, and then passed into Merge Decoder D_M to output the speech

$$\mathbf{X}' = D_A(\mathbf{V_p}, \mathbf{V_c}) = D_M([\mathbf{V_p}; D_C(\mathbf{V_c})]). \tag{4.4}$$

Text Encoder: E_T When text $\mathbf{Y} = \{y_1, ..., y_L\}$ with length L serves as input, it is firstly encoded into unit token vectors

$$\mathbf{V_u} = \{\mathbf{v_{u_1}}, ..., \mathbf{v_{u_L}}\} = E_U(\mathbf{Y}), \tag{4.5}$$

through Unit Encoder E_U.

The number of input text tokens is usually much smaller than the frame length of output speech. To match the content embedding encoded from text and speech, we need to predict the frame length of each unit token, and replicate the unit token vectors accordingly to obtain the frame-level content embedding vectors with the same length from speech. It is called the length regulation technique originally proposed in TTS [184]. Specifically, we use Duration Predictor DP to predict the frame lengths of each text units according to the speaker characteristics

$$\mathbf{l'_u} = \{l'_{u_1}, ..., l'_{u_L}\} = DP([\mathbf{V_u}; \mathbf{v_s}]), \tag{4.6}$$

and replicates unit token vectors according to frame lengths through Length Regulator LR to obtain frame-level content embedding vectors

$$\mathbf{V_c} = \{\mathbf{v_{c_1}}, ..., \mathbf{v_{c_T}}\} = LR(\mathbf{V_u}, \mathbf{l'_u}), \text{where } T = \Sigma_{i=1}^{L} l'_{u_i}. \tag{4.7}$$

doi:10.6342/NTU202302463

For example, if the text input sequence and corresponding unit token vectors are $\{a, b, c\}$, $\{\mathbf{v_a}, \mathbf{v_b}, \mathbf{v_c}\}$ respectively and the predicted frame lengths $\{1, 2, 1\}$, the content embedding vectors are $\mathbf{V_c} = \{\mathbf{v_a}, \mathbf{v_b}, \mathbf{v_b}, \mathbf{v_c}\}$.

It is worth noting that although the speaker embedding vector $\mathbf{v_s}$ is used during the generation of content embedding vectors, it is only used for duration prediction according to speaker characteristics and replication of text vectors. Therefore, each content embedding vector does not contain speaker information. Overall, the generation of content embedding vectors through Text Encoder E_T can be described as

$$\mathbf{V_c} = E_T(\mathbf{Y}, \mathbf{v_s}) = LR(E_U(\mathbf{Y}), DP([E_U(\mathbf{Y}); \mathbf{v_s}])). \tag{4.8}$$

Text Decoder: D_T When the output is text, Text Decoder D_T takes the content embedding vectors as input and output the text sequence. In recent state-of-the-art sequence-to-sequence ASR models, two text sequences are decoded as output by two decoders: sequence-to-sequence (S2S) and connectionist temporal classification (CTC) decoders D_{S2S} and D_{CTC}. During the inference, we can simply select the text with the better decoder during training.

$$[\mathbf{Y'_{CTC}}; \mathbf{Y'_{S2S}}] = D_T(\mathbf{V_c}) = [D_{CTC}(\mathbf{V_c}); D_{S2S}(\mathbf{V_c})]. \tag{4.9}$$

Five tasks for MTL

In this subsection, we describe how to combine the modules in SpeechNet into different speech tasks, which are also depicted in Figure 4.4.

doi:10.6342/NTU202302463

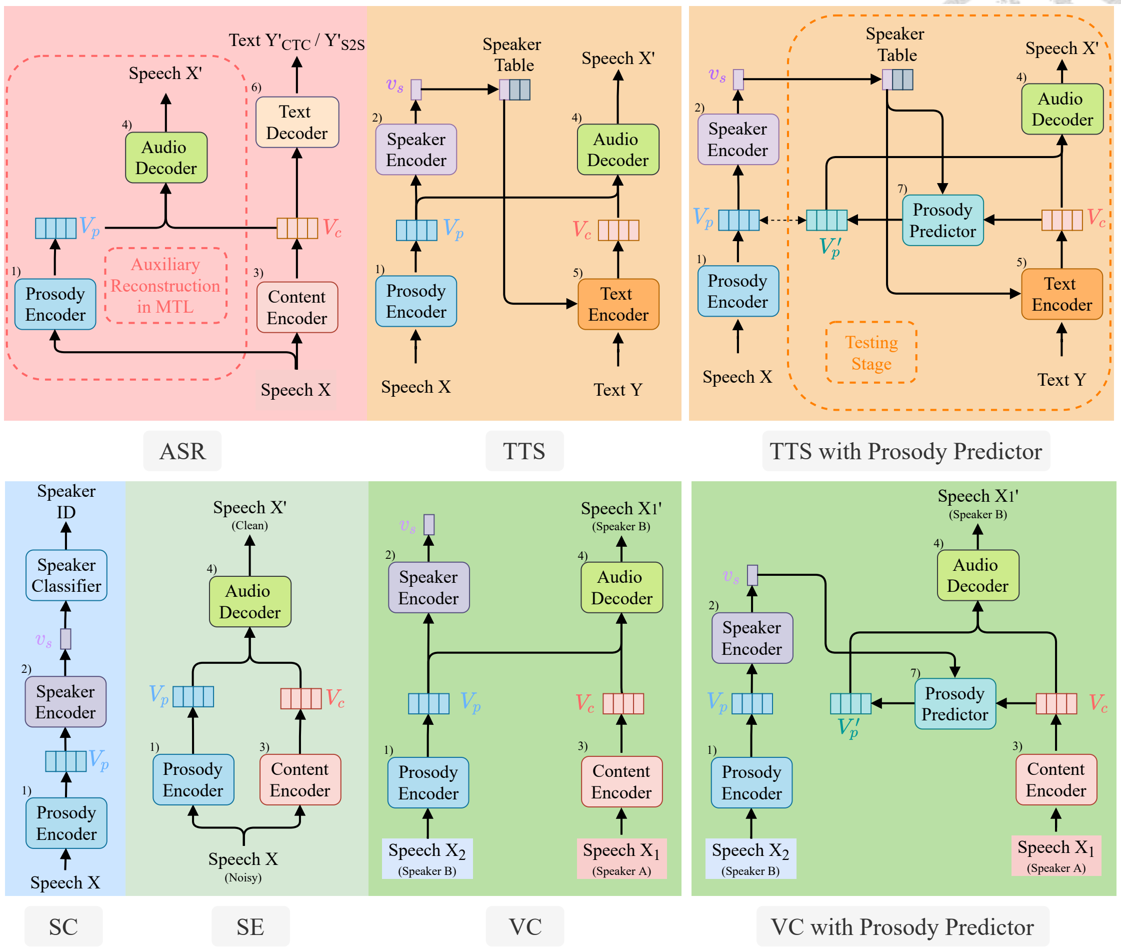


Figure 4.4: This figure shows how to combine the modules in SpeechNet into five different speech tasks. Each module block in this figure shares the same color and index in Figure 4.3.

Automatic Speech Recognition (ASR) In ASR, the input is speech $\mathbf{X}$ and the output is the corresponding transcription text $\mathbf{Y}'_{\mathbf{CTC}}$ and $\mathbf{Y}'_{\mathbf{S2S}}$:

$$[\mathbf{Y}'_{\mathbf{CTC}}; \mathbf{Y}'_{\mathbf{S2S}}] = D_T(E_C(\mathbf{X})). \tag{4.10}$$

The objective function is similar to those used in previous works [185, 186], which is a weighted sum of a S2S loss and a CTC loss:

$$L_{ASR} = -\alpha_{ASR} \log P_{CTC}(\mathbf{Y}'_{\mathbf{CTC}}|\mathbf{X}) - (1 - \alpha_{ASR}) \log P_{S2S}(\mathbf{Y}'_{\mathbf{S2S}}|\mathbf{X}), \tag{4.11}$$

doi:10.6342/NTU202302463

where P_{S2S} and P_{CTC} are the S2S and CTC frame-wise posterior distributions of $\mathbf{Y'_{S2S}}$ and $\mathbf{Y'_{CTC}}$ given corresponding source $\mathbf{X}$ respectively, and α_{ASR} is a scalar hyperparameter.

In MTL, an auxiliary reconstruction objective function is also applied for aligning content vector space with other tasks:

$$L_{recon} = \parallel D_A(E_P(\mathbf{X}), E_C(\mathbf{X})) - \mathbf{X} \parallel^2 . \tag{4.12}$$

The final loss for ASR is

$$L_{ASR\_total} = L_{ASR} + L_{recon}. \tag{4.13}$$

Speech Enhancement (SE) In SE, the model takes noisy speech as input and outputs clean speech. Here we encode the input noisy speech $\mathbf{X_{noisy}}$ into two parts, prosody embeddings and content embeddings, and decode back the denoised speech $\mathbf{X}'$:

$$\mathbf{X}' = D_A(E_P(\mathbf{X_{noisy}}), E_C(\mathbf{X_{noisy}})). \tag{4.14}$$

The objective function is the mean absolute error (MAE) between the predicted and clean speech, $\mathbf{X}'$ and $\mathbf{X_{clean}}$, for more sensitivity to noise than mean square error (MSE):

$$L_{SE} = |\mathbf{X}' - \mathbf{X_{clean}}|. \tag{4.15}$$

Speaker Classification (SC) In SC, the model takes speech as input and outputs the speaker identity. Here we encode the input speech $\mathbf{X}$ into a speaker embedding vector, and use a speaker classifier C_S to recognize the speaker S':

$$S' = C_S(E_S(E_P(\mathbf{X}))). \tag{4.16}$$

The objective function is the cross entropy loss with regard to speaker labels:

$$L_{SC} = -\log P(S'|\mathbf{X}). \tag{4.17}$$

doi:10.6342/NTU202302463

Text-to-speech Synthesis (TTS) In TTS, we want to output a speech $\mathbf{X}'$ according to a text sequence $\mathbf{Y}$ conditioned on speaker characteristics. Specifically, we maintain a trainable speaker embedding table, where every speaker in the training data corresponds to exactly one embedding vector in the table. During training, we hope the speaker embedding output by Speaker Encoder to be as close as possible to the embedding in the table. Therefore we have a MSE speaker loss:

$$L_{speaker} = \parallel E_S(E_P(\mathbf{X})) - \mathbf{v_s'} \parallel, \tag{4.18}$$

where $\mathbf{v_s'}$ is the corresponding speaker embedding of $\mathbf{X}$ in the table.

Then the overall TTS process becomes:

$$\mathbf{X}' = D_A(E_P(\mathbf{X}), E_T(\mathbf{Y}, \mathbf{v_s'})). \tag{4.19}$$

The objective function in training is the sum of (a) the mean square error (MSE) between the predicted and target speech $\mathbf{X}'$ and $\mathbf{X}$, (b) the MAE between the logarithms of predicted and original frame durations of text units, $\log(\mathbf{l_u'})$ and $\log(\mathbf{l_u})$ and (c) $L_{speaker}$:

$$L_{TTS} = \parallel \mathbf{X}' - \mathbf{X} \parallel^2 + |\log(\mathbf{l_u'}) - \log(\mathbf{l_u})| + L_{speaker}. \tag{4.20}$$

There are two problems with this setting. Firstly, during training, there is no additional constraint, so Prosody Encoder and Audio Decoder alone may become an autoencoder, and the content embeddings can be ignored. Secondly, during inference, since the target speech is not available as input of Prosody Encoder, the input speech $\mathbf{X}$ has to be any other speech sentence uttered by the same speaker. However, the prosody of a speech is closely related to the content and duration of the speech. Because the prosodies of input speech of Prosody Encoder and target speech do not match, the generated speech cannot be produced well. We will address these two issues later.

doi:10.6342/NTU202302463

Voice Conversion (VC) In VC, we try to convert the voice of an audio clip from one speaker to another while preserving the content. Specifically, we input two speech utterances $\mathbf{X_1}$ and $\mathbf{X_2}$ with the same content but different speakers, and output the converted speech utterance $\mathbf{X'_{12}}$ with the content of $\mathbf{X_1}$ and speaker characteristics of $\mathbf{X_2}$

$$\mathbf{X'_{12}} = D_A(E_P(\mathbf{X_2}), E_C(\mathbf{X_1})). \tag{4.21}$$

Besides, to make the training more stable and easier, we also train the network to reconstruct the original utterances $\mathbf{X_1}$ and $\mathbf{X_2}$:

$$\mathbf{X'_1} = D_A(E_P(\mathbf{X_1}), E_C(\mathbf{X_1})). \tag{4.22}$$

$$\mathbf{X'_2} = D_A(E_P(\mathbf{X_2}), E_C(\mathbf{X_2})). \tag{4.23}$$

The objective function is the sum of MSE losses of conversion and reconstruction:

$$L_{VC} = \| \mathbf{X'_{12}} - \mathbf{X_2} \|^2 + \| \mathbf{X'_1} - \mathbf{X_1} \|^2 + \| \mathbf{X'_2} - \mathbf{X_2} \|^2 . \tag{4.24}$$

During inference of this setting, since the target converted speech is not available as input of Prosody Encoder, the input speech $\mathbf{X_2}$ has to be any other speech sentence uttered by the same speaker. Similar to TTS, because the prosodies of input speech of Prosody Encoder and target converted speech do not match, the generated speech cannot be produced well. We are now going to address this issue.

Adding one more module for TTS and VC: Prosody Predictor

We point out the issues of TTS and VC in the previous two paragraphs. Here we address them by proposing a modified version of SpeechNet by simply adding one additional module, Prosody Predictor, to generate the estimated prosody embeddings of target speech

doi:10.6342/NTU202302463

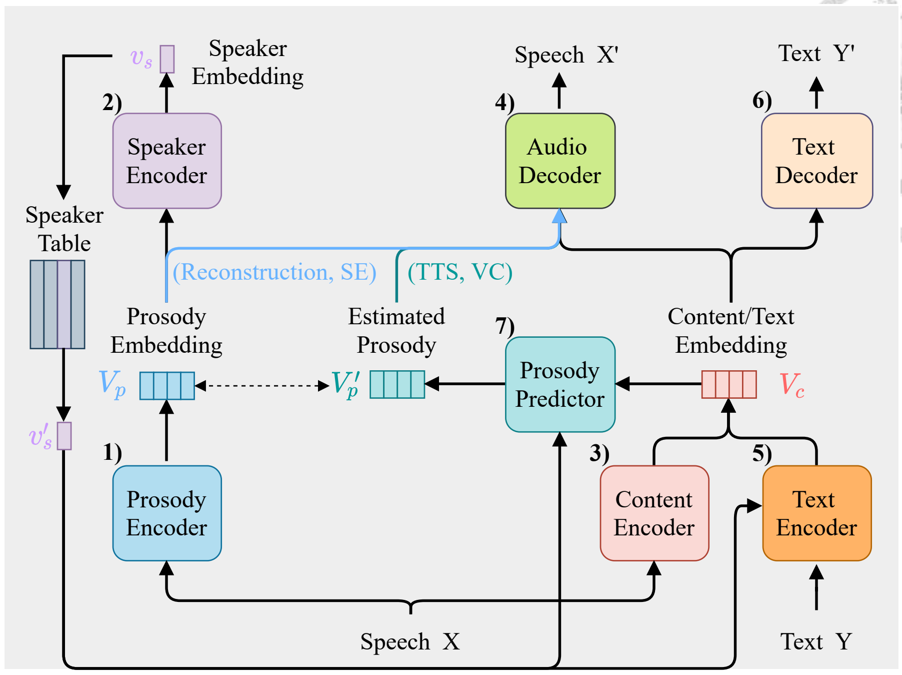


Figure 4.5: The modified architecture of SpeechNet by adding Prosody Predictor.

according to content and speaker embeddings

$$\mathbf{V_p'} = P_P(\mathbf{V_c}, \mathbf{v_s}). \tag{4.25}$$

During training, there is one additional loss to make the estimated prosody $\mathbf{V_p'}$ as close as possible to the original target prosody $\mathbf{V_p}$ generated by Prosody Encoder:

$$\text{(In TTS)}\ \ L_{prosody\_TTS} = \| \ E_P(\mathbf{X}) - P_P(E_T(\mathbf{Y}, \mathbf{v_s'}), \mathbf{v_s'}) \ \|^2 . \tag{4.26}$$

$$\text{(In VC)}\ \ L_{prosody\_VC} = \| \ E_P(\mathbf{X_{12}}) - P_P(E_C(\mathbf{X_1}), E_S(E_P(\mathbf{X_2}))) \ \|^2 . \tag{4.27}$$

Then the Audio Decoder takes $\mathbf{V_p'}$ as input in TTS and VC. The modified version of SpeechNet is shown in Figure 4.5.

With Prosody Predictor, during inference, the generation of speech does not need to rely on the prosody from the speech input. The overall TTS process becomes:

$$\mathbf{X'} = D_A(P_P(E_T(\mathbf{Y}, \mathbf{v_s'}), \mathbf{v_s'}), E_T(\mathbf{Y}, \mathbf{v_s'})). \tag{4.28}$$

doi:10.6342/NTU202302463

And the overall VC process becomes:

$$\mathbf{X}'_{\mathbf{12}} = D_A(P_P(E_C(\mathbf{X_1}), E_S(E_P(\mathbf{X_2}))), E_C(\mathbf{X_1})). \tag{4.29}$$

For the experiments in Subsection 4.2.5, we use SpeechNet with Prosody Predictor. We also present experiments with SpeechNet without Prosody Predictor.

4.2.2 Experimental setup

This subsection introduces the model architecture, input/output formats of data, datasets, and evaluation metrics used in this section.

Model architecture

Transformers [187] have achieved state-of-the-art performances on many NLP tasks [109, 174]. More recently, many tasks in the speech domain started to use transformer-based models, such as ASR [188, 189, 185], SE [190, 191, 192], SC [193, 194, 195], TTS [196, 197, 198] and VC [199, 200, 201].

In this section, we adopt Conformer [188] layers as the architectures for Prosody Encoder, Content Encoder, Audio Decoder and Prosody Predictor. The convolution blocks in Conformer make it empirically better than Transformer for speech data. Speaker Encoder is a self-attention pooling model [202, 203]. For Unit Encoder, Duration Predictor and Length Regulator in Text Encoder, we use similar Transformer-based architectures as those in FastSpeech 2 [196]. In the original FastSpeech 2, since it performs single-speaker TTS, no additional speaker embedding is required. In our SpeechNet, we concatenate the unit token vectors and speaker embedding vector as the input of Duration Predictor for the length regulation. Finally, the ordinary Transformer layers are adopted for S2S Decoder

doi:10.6342/NTU202302463

in Text Decoder, and CTC Decoder in Text Decoder is a single-layer fully-connected network. We insert 1D-convolution blocks right before Content Encoder for down-sampling. It is commonly used in ASR to obtain more compact content information [188, 189, 185]. Accordingly, we apply a similar down-sampler on content embedding vectors produced by Text Encoder. We also use 1D-convolution blocks right after Prosody Predictor and Content Decoder in Audio Decoder for up-sampling.

4.2.3 Implementation details

The batch size of each task is 16. We adopt AdamW optimizer [204] with learning rate 3.0e-4, epsilon 1.0e-12 and betas [0.9, 0.999]. The dropout rate is set to 0.1 except for the dropout rate 0.5 in Duration Predictor in Text Encoder. All the model parameters have decaying factor 0.01 except for those with names containing ”bias”, ”norm-ff.weight”, ”norm-mha.weight”, ”norm-conv.weight” and ”norm-final.weight”. The numbers of Conformer layers in each module are: Content Encoder 6, Speaker Encoder 3, Content Decoder in Audio Decoder 3, Merge Decoder in Audio Decoder 3, Unit Encoder in Text Encoder 4 and S2S Decoder in Text Decoder 4. The CNN down- and up-samplers have 2 1d-convolution blocks with sample rate 4. The α for ASR in (4.11) is 0.3. The σ’s for the multi-task objective loss are initialized as 1.

The hidden dimension size is 256, and the linear unit size is 1024. The head number is 4 except for the head number 2 in Unit Encoder in Text Encoder. We use a linear learning rate warmup with 10000 steps and a linear decay with 100000 steps.

All the experiments are conducted on NVIDIA V100 GPUs. Each trial requires 2 GPUs with memory size 32GB.

doi:10.6342/NTU202302463

Datasets and evaluation metrics

We use the commonly adopted dataset and evaluation metric for each task.

For ASR, LibriSpeech [137] is a corpus of read English speech from audiobooks of the LibriVox project. We perform ASR on the "train-clean-100" set for training, which contains 100-hour speech uttered by 251 speakers, and on the "test-clean" set for evaluation. For easier comparison in our experiments, we use greedy decoding and do not use beam-search decoding and additional language model rescoring during the inference. We measure the performance of ASR by the word error rate (WER).

For SE, Nonspeech [205] is a widely used noise dataset containing 100 types of noises. We choose LibriSpeech [137] as clean speech and randomly augment with noises from Nonspeech with various SNRs to create paired data for both training and testing. During the training stage, the "train-clean-100" set is augmented with SNR $\in \{3, 6, 9\}$, while in the testing stage, the "test-clean" set is augmented with $SNR \in \{-8, -6, -4, -2, 0, 2, 4, 6, 8\}$, providing a more severe condition to test whether the model can generalize. For evaluation, we report SiSDR, PESQ[206], and STOI [207]. The first one measures the scale-invariant signal-to-distortion ratio, and the others align well with human's perspective. We select the checkpoint of the best SiSDR on validation set for testing.

For SC, VoxCeleb1 [208] contains speech uttered by 1,251 celebrities extracted from videos uploaded to YouTube. We take 100 speakers from the official training and testing sets in this section. The speaker classification accuracy is used as the evaluation metric of SC.

For TTS, LibriTTS [209] is a multi-speaker English corpus of read English speech from the audiobooks of the LibriVox project. Utterances with significant background noise

doi:10.6342/NTU202302463

are excluded in LibriTTS. Montreal Forced Aligner [210] is used to obtain the durations of phonemes. We randomly select 200 utterances from the "train-clean-100" set for testing and the remaining ones for training. The MSE between the output and groundtruth mel-spectrograms serves as the evaluation metric of TTS.

For VC, CMU Arctic [211] is a corpus which consists of speech utterances by 18 speakers recorded under studio conditions. Every speaker utters the same set of sentences. In this section, 1133 pairs of data are selected as testing data, and the remaining are for training. Each training instance is a pair of speech utterances by different speakers. The MSE between the output and groundtruth mel-spectrograms serves as the evaluation metric of VC.

For further evaluation of TTS and VC, we apply a linear transformation to recover linear-scale spectrograms and the Griffin-Lim vocoder [212] to convert spectrograms back to wav files for listening.

For all speech, we use 16kHz sample rate and extract the 80-dim mel-spectrogram features with 25 ms window size and 10 ms hop size. Then we add the first- and second-order derivatives and apply the cepstral mean and variance normalization (CMVN) commonly used for ASR and SC in previous works. The text input units for TTS are phonemes, and the output text units for ASR are subword units by the Byte Pair Encoding (BPE) [213], so the cross entropy loss is computed on the subword units.

The 100 speakers in VoxCeleb1 selected for SC are from id10001 to id10099. The testing sets in CMU Arctic are aew and slt.

doi:10.6342/NTU202302463

4.2.4 Optimization Strategies

During MTL, some tasks may share some modules, and the gradients computed from different objective functions of these tasks are accumulated to update the shared modules. However, there are two problems: (1) how to balance these objective functions with different types and scales, and (2) how to deal with conflicting gradients of parameters between different tasks. We experiment with two popular MTL optimization strategies, tackling these two problems respectively.

Loss balancing for MTL

We adopt an automatic loss balancing technique (which we denote as "AutoLoss" in the experiments) based on the task-dependent data-independent uncertainty measurement of each task [167]. It has been shown effective to capture the relative confidence between tasks and learn loss weights for tasks.

The overall objective function of n objective functions can be defined as:

$$\Sigma_{i=1}^{n} L_i' = \Sigma_{i=1}^{n} (\frac{1}{\sigma_i^2} L_i + \log \sigma_i), \tag{4.30}$$

where L_i' is the scaled version of the original loss L_i with the introduction of learnable scalar variables σ's.

Eliminating gradient conflicts in MTL

PCGrad [170] is a gradient manipulation approach to handle conflicting gradients on the same set of parameters. Specifically, for parameters in every layer of the model, we perform PCGrad: If the gradients between two tasks have negative cosine similarity, the gradient

doi:10.6342/NTU202302463

of one task is projected onto the normal plane of the gradient of the other task. In this way, the conflicting component of the gradient no longer exists.

4.2.5 Experiments

We conduct single-task, two-task, and five-task learning experiments with the five tasks described in Subsection 4.2.1. Moreover, We experiment with two popular optimization strategies for MTL, "AutoLoss" and "PCGrad", which are described in Subsection 4.2.4, for all of the two-task learning experiments. We also have the ablation study of two optimization strategies for five-task learning.

Single-task and two-task learning results

The single-task and two-task experiment results are shown in Table 4.2. Each column shows the evaluation performance with a specific metric of a task. The first row is the name of the evaluation task, and the second row is the evaluation metric. The down-/up-arrow beside the evaluation metric means the better performance results in lower/higher numbers of that metric. The diagonal cells with pink shadow are the single-task results. The off-diagonal cells represent the results of joint training with a specific auxiliary task[1]. The best number on each metric is highlighted with bold font. If the numbers of multi-task are better than the single-task ones, the numbers are underlined.

We also plot an "improvement graph" based on two-task learning results, as shown in Figure 4.6, to illustrate the beneficial relationship between all task pairs. For example, if the model trained with ASR and SE improves the ASR WER compared to single-task ASR, we connect a directed edge from SE to ASR to denote the former benefits the latter.

[1]The columns are for the evaluated tasks, while the rows for the auxiliary tasks

doi:10.6342/NTU202302463

Table 4.2: The results of single-task (pink cells) and two-task learning of five tasks.

	ASR	**SE**			**SC**	**TTS**	**VC**
Auxiliary	**WER↓**	**PESQ↑**	**SISDR↑**	**STOI↑**	**ACC↑**	**MSE↓**	**MSE↓**
ASR	0.329	**2.46**	5.62	**0.880**	0.746	3.06	5.93
SE	0.320	2.44	**5.90**	0.877	0.820	3.08	6.06
SC	**0.307**	2.15	4.02	0.850	0.860	2.98	6.04
TTS	0.322	2.29	4.96	0.865	**0.879**	2.94	6.02
VC	0.316	2.02	4.80	0.847	0.703	3.57	5.95

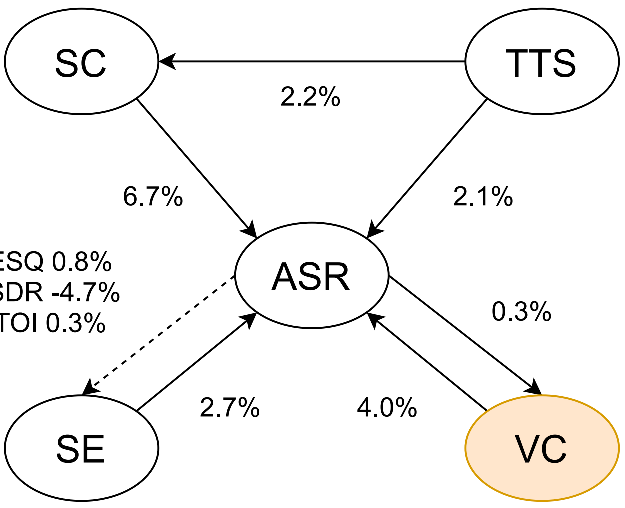


Figure 4.6: The improvement graph based on two-task learning results. We show the relative improvement beside the edge. The dashed line represents that PESQ and STOI are improved but SISDR is not.

We can observe ASR can be improved by all of the other tasks in two-task learning from the results. It indicates the related information provided by the other tasks can help Content Encoder to generate content embeddings for Text Decoder to produce text transcriptions. SE is improved slightly by ASR in terms of the PESQ and STOI metrics. The

doi:10.6342/NTU202302463

Table 4.3: The ablation study of two optimization strategies with five-task learning of five tasks.

	ASR	SE			SC	TTS	VC
Optim Strategy	**WER↓**	**PESQ↑**	**SISDR↑**	**STOI↑**	**ACC↑**	**MSE↓**	**MSE↓**
AutoLoss + PCGrad	0.511	1.99	3.71	0.837	0.451	3.36	6.01
AutoLoss	0.600	2.04	3.68	0.833	0.101	3.26	<u>5.88</u>
PCGrad	0.839	2.00	3.91	0.838	0.466	3.19	5.96
No Strategy	0.538	2.12	3.82	0.838	0.044	3.18	**<u>5.86</u>**

performance of SC is improved with the aid of TTS. VC is also slightly improved by ASR.

Five-task learning results and ablation study of optimization strategies

The ablation study of two optimization strategies with five-task learning is shown in Table 4.3. The optimization of five-task learning is much more difficult than single-task or two-task learning, as we can see, all of the performances degrade except for VC. Specifically, without PCGrad strategy, i.e. "AutoLoss" and "No Strategy" in the Table, SC cannot even be learned effectively. However, VC can be improved in these cases compared to single-task learning. With only "PCGrad" strategy, ASR performs the worst. The model can learn all of the tasks only when using both AutoLoss and PCGrad strategies. But the results are still worse than single-task learning. This ablation study shows that optimization strategies influence the training of MTL significantly. The two popular MTL optimization strategies in our experiments cannot help five-task learning outperform single-task learning.

doi:10.6342/NTU202302463

Table 4.4: The results of single-task learning of TTS and VC with SpeechNet without Prosody Predictor.

TTS	VC
MSE↓	**MSE↓**
23.21	16.07

Experiments with SpeechNet without Prosody Predictor

We show the single-task learning results of TTS and VC using SpeechNet without Prosody Predictor. The results are shown in Table 4.4. We can observe that the testing MSEs cannot be decreased because the prosodies of input speech of Prosody Encoder and target speech do not match. It validates our motivation and the necessity to generate the estimated prosody of target speech based on content and speaker embeddings.

Analysis and discussion

In this section, we exhibit five important speech processing tasks that can be learned with SpeechNet and conduct experiments of single-task, two-task, and five-task learning with two popular MTL optimization strategies. However, many important research directions can be further extended from SpeechNet. (1) As shown in the last subsection, suitable optimization strategies for MTL of speech processing tasks are important and desirable. SpeechNet can be a testbed for developing MTL optimization strategies on speech processing tasks. (2) Besides MTL, other training schemes involving multiple tasks can be investigated in the future, such as transfer learning or meta learning. Different sizes of data are also worth investigating. (3) SpeechNet is flexible and easy to modify or add

doi:10.6342/NTU202302463

new modules. Therefore many other speech or text processing tasks can be joined for research. For example, in our experiment results, ASR benefits from MTL the most. It may indicate other speech processing tasks that take speech input and produce text output can also benefit from MTL with SpeechNet, such as speech translation or multilingual speech recognition.

4.2.6 Conclusion

In this section, we propose a universal modularized model for speech processing tasks. We select five common and important tasks for multi-task learning experiments.

So far we have investigated the optimization of model parameters under a fixed network architecture with MTL of speech processing tasks. In the next section, we further investigate nueral architecture search during MTL training.

4.3 DARTS-ASR: Differentiable Architecture Search for Multilingual Speech Recognition and Adaptation

In previous sections, we have investigated the optimization of model parameters under a fixed network architecture with MTL of speech processing tasks. In this section, we further investigate nueral architecture search during MTL training. Inspired by DARTS, we propose an ASR approach with efficient gradient-based architecture search, **DARTS-ASR**. In order to examine the generalizability of DARTS-ASR, we apply our approach not only on many languages to perform monolingual ASR, but also on a multilingual ASR setting, where the architecture and parameter weights are pretrained on some source languages,

doi:10.6342/NTU202302463

and then adapted on the target language. It has recently been shown that multilingual ASR [79, 100, 101, 102, 103, 104, 105, 106, 94] can improve ASR performance on many low-resource languages. In the above previous works, the initial parameters or shared encoder learned from many source languages are used to build a better acoustic model for the target language. Different from previous works, DARTS-ASR further learns better network architecture from the source languages.

Following the previous works [101, 103, 106, 94], we conducted experiments on the multilingual dataset, IARPA BABEL [214]. The experiment results show that our approach outperformed the baseline fixed-topology architecture by 10.2% and 10.0% relative reduction on character error rates (CER) under monolingual and multilingual ASR settings respectively. Furthermore, we perform some analysis on the searched architectures by DARTS-ASR.

4.3.1 Proposed Approach: DARTS-ASR

In previous works of ASR, network architectures were manually designed with human experience, and parameter weights can only be optimized under the fixed topology. Although those networks work well in previous works, they are very likely not the optimal architectures for ASR. In this section, we propose **DARTS-ASR**, where the network architecture can be automatically learned jointly with parameter weights.

Search Space and Continuous Relaxation of Architecture Representation

To search for the network architecture, we first define the search space. As shown in Figure 4.7, the search space is a directed acyclic graph consisting of K nodes $\{n_0, n_1, ..., n_K\}$,

doi:10.6342/NTU202302463

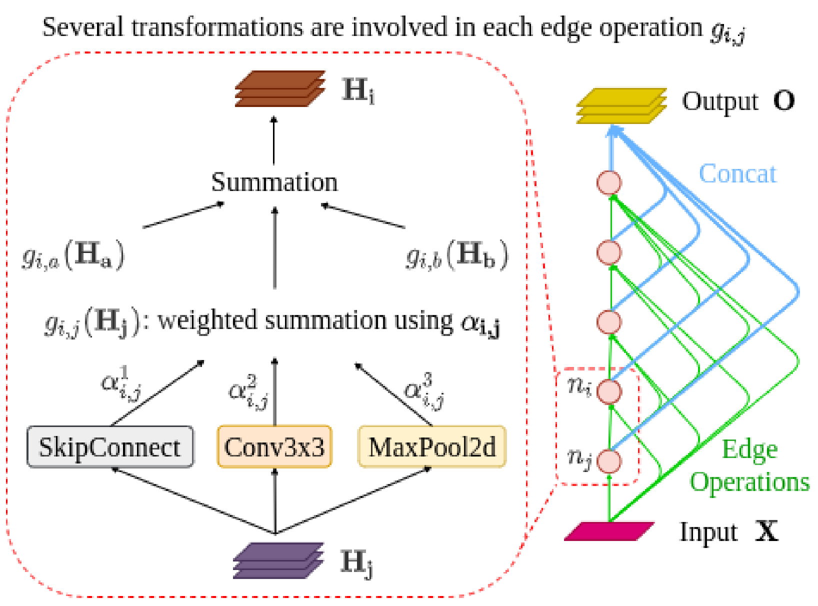


Figure 4.7: Differentiable ARchiTecture Search (DARTS) for ASR.

where n_0 is the input feature $\mathbf{X}$ and the other nodes represent latent features $\mathbf{H_1}, \mathbf{H_2}, ..., \mathbf{H_K}$. In the scenario of ASR, the input feature $\mathbf{X}$ is a segment of acoustic features such as Mel-filterbanks, and latent features $\mathbf{H_i}$ have the shape like CNN feature maps. For each node n_i, there are i directed input edges $\{e_{i,0}, e_{i,1}..., e_{i,i-1}\}$, where each edge $e_{i,j}$ transforms $\mathbf{H_j}$ with some operation $g_{i,j}$. The feature H_i of each node n_i is the summation of the operations of all its previous nodes as below.

$$\mathbf{H_i} = \Sigma_{j<i} \; g_{i,j}(\mathbf{H_j}), \tag{4.31}$$

$$where \;\; g_{i,j}(\mathbf{H_j}) = \Sigma_{f\in\mathbb{F}} \frac{\exp(\alpha^f_{i,j})}{\Sigma_{f'\in\mathbb{F}} \exp(\alpha^{f'}_{i,j})} f(\mathbf{H_j}). \tag{4.32}$$

The operation $g_{i,j}$ is the weighted sum of a set of transformations $\mathbb{F}$. Each transformation acts as a typical network layer like 3x3Conv, MaxPool2d or skip connection. Some of the transformations have parameter weights to be learned (for example, 3x3Conv), while some of them do not (for example, MaxPool2d, skip connection). The transformation weights in an operation are parameterized by a vector $\alpha_{\mathbf{i,j}}$ of dimension $|\mathbb{F}|$. The final output of

doi:10.6342/NTU202302463

searched architecture is the concatenation of all the latent features:

$$\mathbf{O} = Concat(\mathbf{H_1}, \mathbf{H_2}, ..., \mathbf{H_K}). \tag{4.33}$$

These variables $\alpha_{\mathbf{i,j}}$ is jointly trained with parameter weights directly by gradient descent. If the weights $\alpha_{\mathbf{i,j}}$ are sparse, (4.32) can be regraded as the selection of transformations used to connect node n_i and n_j, so $\alpha_{\mathbf{i,j}}$ can be considered as controlling the network architecture. Therefore, architecture search can be performed through learning the continuous variables $\{\alpha_{i,j}\}$. With continuous relaxation of architecture representation by variables $\{\alpha_{i,j}\}$, the transformation components and connections of the model can be softly designed by gradient descent optimization.

Multilingual Pretraining and Adaptation

To examine the generalizability of DARTS-ASR, we apply DARTS-ASR on not only monolingual but also multilingual ASR to check if it works on ASR of different languages. For monolingual ASR, each language data is separately trained with respective training data, and the model is not shared across languages. For multilingual ASR, some source languages are used for pretraining and some target languages for adaptation. For each source language in pretraining, the input is encoded by the shared model, and then fed into the language-specific head of the corresponding language to output the prediction sequence. During adaptation of target languages, the pretrained shared model is used for finetuning, but the head is trained from scratch.

We apply three types of finetuning approaches:

- Adapt only param.: the continuous variables $\{\alpha_{i,j}\}$ from pretraining are fixed, and only parameter weights in the transformations are trained. That is, the network ar-

doi:10.6342/NTU202302463

chitecture is learned from the source languages, and with the learned architecture, its network parameters are learned from the target language.

- Adapt arch.+param.: the continuous variables $\{\alpha_{i,j}\}$ keep being trained with parameter weights in the transformations. That is, both the network architecture and network parameters learned from source languages are further fined-tuned on the target language.
- Adapt pruned arch.+param.: the architecture learned from the source languages is pruned by removing some transformations with low $\{\alpha_{i,j}^{f}\}$ values. Then the pruned $\{\alpha_{i,j}\}$ keeps being trained with remaining parameter weights.

4.3.2 Experiments

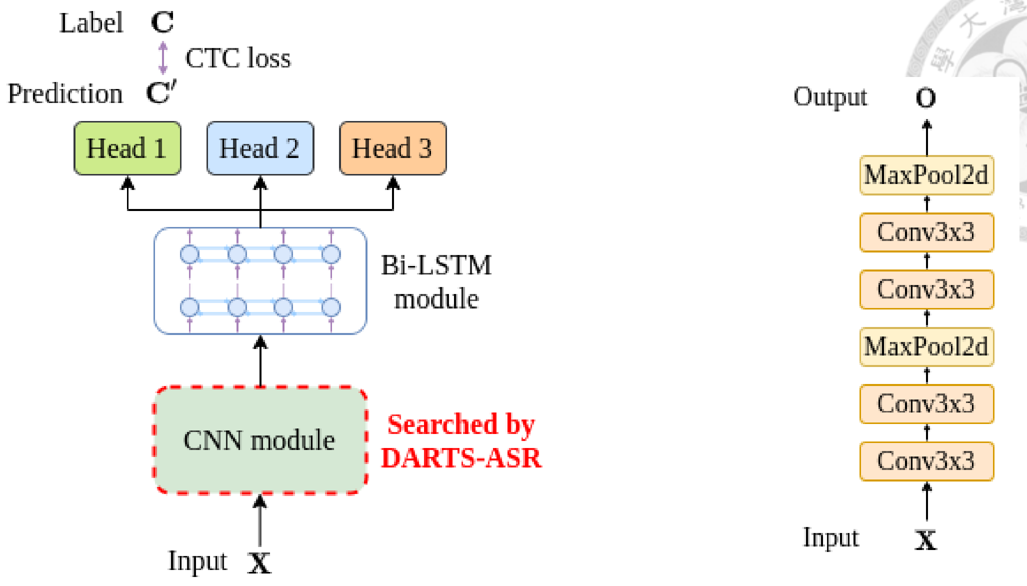
Data and Features

We conducted experiments on the Full Language Pack from the multilingual dataset, IARPA BABEL [214]. Three source languages were selected for multilingual pretraining: Bengali (Bn), Tagalog (Tl) and Zulu (Zu), and four target languages for adaptation: Vietnamese (Vi), Swahili (Sw), Tamil (Ta) and Kurmanji (Ku). We followed the ESPnet recipe [215] for data preprocessing and final score evaluation. The acoustic features are 80-dimensional Mel-filterbanks that are computed over a 25ms window every 10ms, plus 3-dimensional pitch features.

Implementation Details

Following the previous works [106, 94], we used a CNN-BiLSTM-Head structure as the multilingual ASR model, as shown in Figure 4.8(a), and adopted Connectionist Temporal

doi:10.6342/NTU202302463



(a) The framework of ASR model.

(b) CNN module as VGG.

Figure 4.8: Multilingual ASR model with CTC.

Classification (CTC) [216] loss as the objective function. The baseline model architecture followed the previous work [94], where the CNN module was a 6-layer VGG block as shown in Figure 4.8(b), and the BiLSTM module was a 3-layer bidirectional LSTM network with 360 cells in each direction. We experimented with the channel number of convolutions in VGG as 128 or 512, and the results of these two settings in the following subsection were named as VGG-Small and VGG-Large. The head used for each language was a linear matrix with softmax activation.

In this section, we applied DARTS-ASR on the CNN module to search for a better architecture for extracting useful features from input. To match the depth and the parameter size of VGG-Large, the number of nodes K in the search space of DARTS-ASR, as mentioned in Subsection 4.3.1, was set to 5, and the channel number of convolutions were 32. The transformation candidates in $\mathbb{F}$ were {3x3 convolution, 5x5 convolution, 3x3 dilated convolution, 5x5 dilated convolution, 3x3 average pooling, 3x3 max pooling, skip

doi:10.6342/NTU202302463

connection}.

In addition to standard convolution blocks and pooling, we also added dilated convolutions and skip connection into the transformation candidate set. Dilated convolutions have generally improved the performance of semantic segmentation, as reported in a previous work [217]. The improvement comes from the fact that dilated convolutions expand the receptive field without loss of resolution or coverage. Although convolutions with strides larger than one and pooling are similar concepts, both reduce the resolution. Skip connection forwards the input to the next layer with an identity function and has been proved to avoid the problem of vanishing gradients. It has become very popular in recent CNN models such as DenseNet [218] or ResNet [108]. Therefore, these two types of transformations were also chosen as candidates during architecture search.

All transformations were of stride one (if applicable), and the convolved feature maps were padded to preserve their spatial resolution. All convolutions were followed by ReLU activation and batch normalization [219]. The operation parametrization vectors $\alpha_{\mathbf{i,j}}$ described in Subsection 4.3.1 were initialized as zero vectors to ensure equal amount of attention over all possible transformations, so parameter weights in every candidate transformation could receive sufficient gradients to learn at the beginning. Adam [161] (lr=0.0001, betas=[0.5, 0.999], decay=0.001) was used as the optimizer for operation parametrization vectors $\alpha_{\mathbf{i,j}}$, and SGD (lr=0.01, momentum=0.9, decay=0.0003) was used as the optimizer for parameter weights. The learning rate was reduced by a factor of 0.2 if no improvement for 3 epochs. All of the training processes were terminated after the validation loss had converged. The performances on the test sets were evaluated with greedy search decoding and 5-gram language model re-scoring.

doi:10.6342/NTU202302463

Results

Table 4.5: CER (%) results of monolingual ASR using different CNN modules.

Language	CNN Module			
	VGG-Small	VGG-Large	DARTS-ASR	
			Full	Only Conv3x3
Vietnamese	46.0	48.3	**40.9**	45.7
Swahili	39.6	38.3	**35.9**	36.8
Tamil	57.9	60.1	**48.0**	51.6
Kurmanji	57.2	56.8	**55.5**	56.5

Monolingual ASR For monolingual ASR on four languages, we evaluated diiferent kinds of CNN modules, VGG-Small and VGG-Large, as listed in Table 4.5. The results of DARTS-ASR using all the seven kinds of transformations mentioned in the last subsection are listed in the third column. We can observe DARTS-ASR significantly outperformed both VGG-Small and VGG-Large, showing that the connection pattern of nodes in DARTS-ASR contributed a lot to the huge performance boosting. It is worth noting that even though the parameter size of VGG-Large was four times as many as VGG-Small, the CERs of Vietnamese and Tamil became worse due to overfitting and the CERs of Swahili and Kurmanji improved only a little. In comparison, the parameter size of DARTS-ASR was also much larger than VGG-Small. However, DARTS-ASR outperformed VGG-Small by 10.2% relative reduction on average CER. It indicates the role of architecture for training DNN is very important.

To further understand the importance of the connection pattern and transformation candidates between nodes, in addition to the search space described before, we constructed

doi:10.6342/NTU202302463

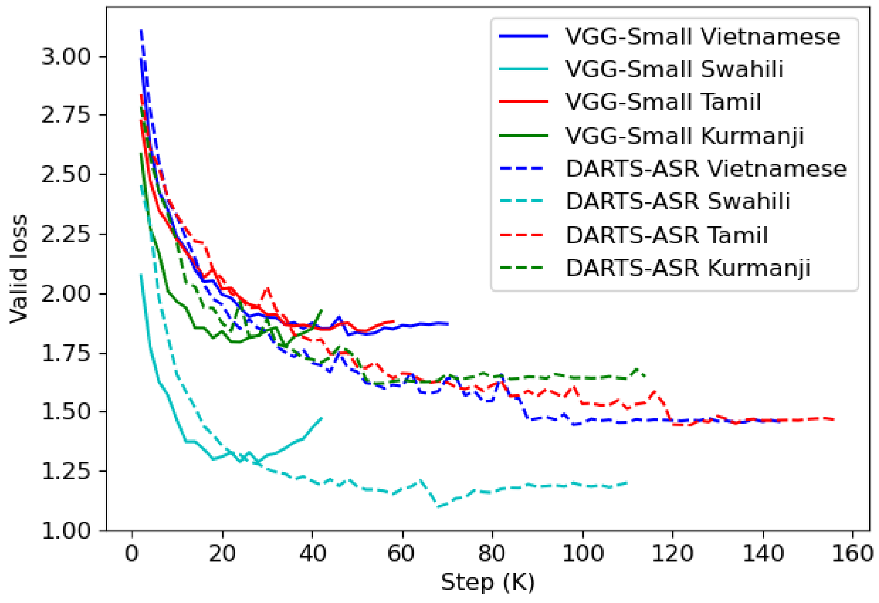


Figure 4.9: Validation loss vs training step with VGG-Small or DARTS-ASR for monolingual ASR on four languages.

another search space for DARTS-ASR: instead of having seven transformation candidates in the search space as described before, there was only {3x3 convolution} in the search space. The channel number of the convolution was set to 256 to match the parameter size of the original search space. The results with only 3x3 convolution are listed in the fourth column. DARTS-ASR outperformed VGG models even with limited search space. It indicates the connection pattern of DARTS-ASR alone contributed a lot to performance improvement. Furthermore, the performance of the full search space outperformed the {3x3 convolution} search space. It proves that diversity of transformation candidates can provide the model an opportunity to find a better architecture.

In Figure 4.9, the validation losses of VGG-Small and DARTS-ASR on different languages are presented. The solid lines are the results of VGG-Small and the dashed lines are those of DARTS-ASR. Different colors stand for different languages. From the lines, we can observe the convergence of VGG-Small was generally faster than DARTS-ASR.

doi:10.6342/NTU202302463

But DARTS-ASR could reach much lower validation losses in the end. The training of VGG-Small suffered from serious overfitting, causing the losses to increase again after some training steps. In comparison, the validation losses of DARTS-ASR could decrease more steadily.

Table 4.6: CER (%) results of multilingual ASR using DARTS-ASR under different finetuning approaches.

Language	Finetuning of DARTS-ASR		
	Adapt only param.	Adapt arch.+param.	Adapt pruned arch.+param.
Vietnamese	**40.9**	**40.9**	41.1
Swahili	33.2	**32.3**	35.3
Tamil	46.4	**45.9**	47.5
Kurmanji	53.6	53.5	**53.2**

Table 4.7: CER (%) results of multilingual ASR using different CNN modules.

Language	CNN Module		
	VGG-Small	VGG-Large	DARTS-ASR
Vietnamese	45.3	43.2	**40.9**
Swahili	36.3	36.1	**32.3**
Tamil	55.7	55.0	**45.9**
Kurmanji	54.5	55.1	**53.5**

Multilingual ASR For multilingual ASR, the model was first pretrained on three source languages, and then adapted on the same four different target languages as in the mono-

doi:10.6342/NTU202302463

lingual ASR experiments, respectively.

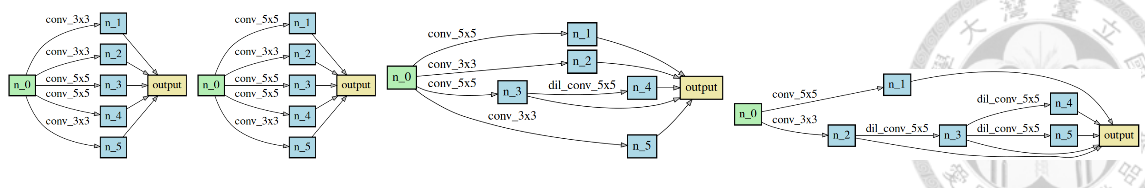
We first conducted experiments to compare the three finetuning approaches described in Subsection 4.3.1, as shown in Table 4.6. Especially for “Adapt pruned arch.+params”, the architecture was pruned by removing all transformations but the top three ones with the highest $\{\alpha_{i,j}^{f}\}$ values in each edge. Then the pruned $\{\alpha_{i,j}\}$ kept being finetuned jointly with remaining parameter weights.

From Table 4.6, we can observe “Adapt arch.+param.” finetuning approach obtained the best performance on average CER. However, “Adapt only param.” and “Adapt pruned arch.+param.” were only a little worse than “Adapt arch.+param.”. It indicates after pre-training, DARTS-ASR can find a generally good architecture and parameter weights for different languages. And the pruned architecture can reduce computational cost while suffering little performance drop. We used “Adapt arch.+param.” finetuning approach for DARTS-ASR in the following experiments.

Then we compared DARTS-ASR with VGG-Small and VGG-Large. The results are listed in Table 4.7. All three kinds of CNN modules got much better performance on multilingual ASR than monolingual ASR. On multilingual ASR, VGG-Large achieved better results than VGG-Small on average CER. Among those, DARTS-ASR still outperformed both VGG-Small and VGG-Large by a significant margin. It indicates DARTS-ASR can also benefit from multilingual learning to build a shared acoustic pretrained model with a better architecture and parameter weights.

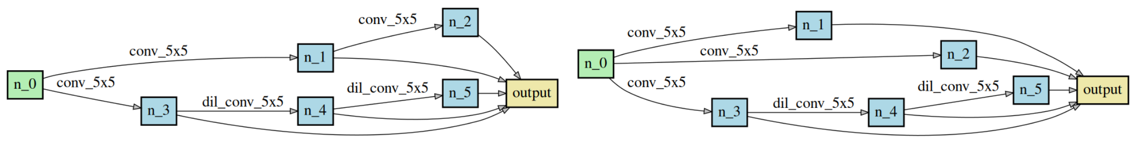
Analysis of Searched Architectures We further plot and analyze the searched architectures by DARTS-ASR. Similar to the original DARTS paper [117], to simplify the illustration of architecture, for each node n_i, we plot the most dominant transformation

doi:10.6342/NTU202302463



(a) Vietnamese. (b) Swahili. (c) Tamil. (d) Kurmanji.

Figure 4.10: Architectures for different languages found by DARTS-ASR in monolingual ASR.



(a) Vietnamese and Kurmanji. (b) Swahili and Tamil.

Figure 4.11: Architectures for different languages found by DARTS-ASR in multilingual ASR.

f_j among all transformations in all entering edges. The selection of the most dominant transformation can be formulated as below.

$$f_j = \arg\max_{j'<i} \ \alpha_{i,j}^{f_{j'}}, \tag{4.34}$$

$$where\ f_{j'} = \arg\max_{f'\in\mathbb{F}} \ \alpha_{i,j}^{f'}. \tag{4.35}$$

The searched architecture for each language on monolingual ASR is shown in Figure 4.10. The architectures of Vietnamese and Swahili were similar, while those of Tamil and Kurmanji were quite different from one another. For multilingual ASR, we plot the searched architectures under the "Adapt arch.+params." finetuning approach. The searched architectures of Vietnamese and Kurmanji were the same as shown in Figure 4.11(a), and those of Swahili and Tamil were the same as shown in Figure 4.11(b). We can observe all of the four searched architectures on multilingual ASR were quite similar, where the

doi:10.6342/NTU202302463

patterns for nodes n_3 to n_5 in the bottom were all the same. Only the patterns for n_1 and n_2 were slightly different. It shows that this kind of network architecture shown in Figure 4.11 is the architecture generally suitable for a wide range of languages.

4.3.3 Conclusion

In this section, we propose an ASR approach with efficient gradient-based architecture search, DARTS-ASR. In order to examine the generalizability of DARTS-ASR, we apply our approach not only on many languages to perform monolingual ASR, but also on a multilingual ASR setting. The experiment results show that our approach outperformed the baseline fixed-topology architecture significantly under both monolingual and multilingual ASR settings. Furthermore, we perform some analysis on the searched architectures by DARTS-ASR.

doi:10.6342/NTU202302463

Chapter 5 Conclusion

Different speech processing tasks require different kinds of information from speech signals, and have different input and output modalities. In this thesis, we explore approaches to disentangle and extract different kinds of information from speech signals with adversarial training, and approaches to handle various speech processing tasks using one single model with multi-task learning.

In Chapter 3, we study the disentangled representation learning using adversarial training. More specifically, in Section 3.1, we explore a two-stage framework to perform phonetic-and-semantic embedding on spoken words considering the context of the spoken words, with the initial experiments on spoken document retrieval. In Section 3.2, we explore a pretraining framework AIPNet based on adversarial training for accent-invariant representation learning and further finetune this model by connecting the accent-invariant module with an attention-based encoder-decoder model for multi-accent speech recognition. In Section 3.3, we extend the disentangled speech representations learning from the word level to the utterance level by proposing a new segmental audio word2vec in which unsupervised spoken word boundary segmentation and disentangled representation learning are jointly learned and mutually enhanced.

In Chapter 4, we study multi-task learning and universal modeling of speech processing tasks. More specifically, in Section 4.1, we use a state-of-the-art SSL pretrained shared model and further finetune it with MTL on various discriminative speech processing tasks, and then evaluate the model on a speech multi-task benchmark. In Section 4.2, we design a universal modularized model for not only discriminative but also generative speech processing tasks. In Section 4.3, we propose an ASR model explored with an ef-

doi:10.6342/NTU202302463

ficient gradient-based architecture search on multilingual tasks, where we use a unified model for extracting representations for data of different languages.

If there is artificial general intelligence in the future, I think it should be a generalized and universal model for various tasks with different modalities. This thesis is one step towards the exploration of that dream.

doi:10.6342/NTU202302463

References

[1] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee, "Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder," *Interspeech 2016*, pp. 765–769, 2016.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[5] L. Tu, K. Gimpel, and K. Livescu, "Learning to embed words in context for syntactic tasks," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017, pp. 265–275.

[6] X. Yu and N. T. Vu, "Character composition model with convolutional neural networks for dependency parsing on morphologically rich languages," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2017, pp. 672–678.

doi:10.6342/NTU202302463

[7] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 260–270.

[8] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2016, pp. 412–418.

[9] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[10] M. Ballesteros, C. Dyer, and N. A. Smith, "Improved transition-based parsing by modeling characters instead of words with lstms," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 349–359.

[11] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[13] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

doi:10.6342/NTU202302463

[14] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence-video to text," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4534–4542.

[15] I. Konstas, S. Iyer, M. Yatskar, Y. Choi, and L. Zettlemoyer, "Neural amr: Sequence-to-sequence models for parsing and generation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 146–157.

[16] W. He, W. Wang, and K. Livescu, "Multi-view recurrent neural acoustic word embeddings," *arXiv preprint arXiv:1611.04496*, 2016.

[17] S. Settle and K. Livescu, "Discriminative acoustic word embeddings: Recurrent neural network-based approaches," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 503–510.

[18] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4950–4954.

[19] S. Bengio and G. Heigold, "Word embeddings for speech recognition," in *Proceedings of the 15th Conference of the International Speech Communication Association, Interspeech*, 2014.

[20] K. Levin, K. Henry, A. Jansen, and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings," in *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE, 2013, pp. 410–415.

doi:10.6342/NTU202302463

[21] S. Toshniwal, H. Tang, L. Lu, and K. Livescu, "Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition," *Proc. Interspeech 2017*, pp. 3532–3536, 2017.

[22] S. Settle, K. Levin, H. Kamper, and K. Livescu, "Query-by-example search with discriminative neural acoustic word embeddings," *Proc. Interspeech 2017*, pp. 2874–2878, 2017.

[23] K. Cho, B. van Merriënboer, Ç. Gulçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[24] Y.-A. Chung and J. Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," *Proc. Interspeech 2018*, pp. 811–815, 2018.

[25] Y.-C. Chen, C.-H. Shen, S.-F. Huang, and H.-y. Lee, "Towards unsupervised automatic speech recognition trained by unaligned speech and text only," *arXiv preprint arXiv:1803.10952*, 2018.

[26] Y.-H. Wang, H.-y. Lee, and L.-s. Lee, "Segmental audio word2vec: Representing utterances as sequences of vectors with applications in spoken term detection," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6269–6273.

[27] T. Tran, S. Toshniwal, M. Bansal, K. Gimpel, K. Livescu, and M. Ostendorf, "Parsing speech: a neural approach to integrating lexical and acoustic-prosodic infor-

doi:10.6342/NTU202302463

mation," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 69–81.

[28] H. Tang, L. Lu, L. Kong, K. Gimpel, K. Livescu, C. Dyer, N. A. Smith, and S. Renals, "End-to-end neural segmental models for speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1254–1264, 2017.

[29] H. Kamper, K. Livescu, and S. Goldwater, "An embedded segmental k-means model for unsupervised segmentation and clustering of speech," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 719–726.

[30] H. Kamper, A. Jansen, and S. Goldwater, "A segmental framework for fully-unsupervised large-vocabulary speech recognition," *Computer Speech & Language*, vol. 46, pp. 154–174, 2017.

[31] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Tenth Annual conference of the international speech communication association*, 2009.

[32] K. Levin, A. Jansen, and B. Van Durme, "Segmental acoustic indexing for zero resource keyword search," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5828–5832.

doi:10.6342/NTU202302463

[33] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2015, pp. 5236–5240.

[34] A. L. Maas, S. D. Miller, T. M. O' neil, A. Y. Ng, and P. Nguyen, "Word-level acoustic modeling with convolutional vector regression," in *Proc. ICML Workshop Representation Learn*, 2012.

[35] N. Holzenberger, M. Du, J. Karadayi, R. Riad, and E. Dupoux, "Learning word embeddings: Unsupervised methods for fixed-size representations of variable-length speech segments," in *Interspeech 2018.* ISCA, 2018.

[36] H. Kamper, "Truly unsupervised acoustic word embeddings using weak top-down constraints in encoder-decoder models," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2019, pp. 6535–3539.

[37] A. Jansen, M. Plakal, R. Pandya, D. Ellis, S. Hershey, J. Liu, C. Moore, and R. A. Saurous, "Towards learning semantic audio representations from unlabeled data," in *NIPS Workshop on Machine Learning for Audio Signal Processing (ML4Audio)*, 2017.

[38] C.-T. Chung and L.-S. Lee, "Unsupervised discovery of structured acoustic tokens with applications to spoken term detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 394–405, 2017.

doi:10.6342/NTU202302463

[39] A. Garcia and H. Gish, “Keyword spotting of arbitrary words using minimal speech resources,” in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 1. IEEE, 2006, pp. I–I.

[40] A. Jansen and K. Church, “Towards unsupervised training of speaker independent acoustic models,” in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[41] A. Jansen, K. Church, and H. Hermansky, “Towards spoken term discovery at scale with zero resources,” in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[42] A. S. Park and J. R. Glass, “Unsupervised pattern discovery in speech,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 186–197, 2007.

[43] V. Stouten, K. Demuynck *et al.*, “Discovering phone patterns in spoken utterances by non-negative matrix factorization,” *IEEE Signal Processing Letters*, vol. 15, pp. 131–134, 2008.

[44] L. Wang, E. S. Chng, and H. Li, “An iterative approach to model merging for speech pattern discovery,” *Proc. APSIPA*, 2011.

[45] N. Vanhainen and G. Salvi, “Word discovery with beta process factor analysis,” in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

doi:10.6342/NTU202302463

[46] J. Driesen *et al.*, "Fast word acquisition in an nmf-based learning framework," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5137–5140.

[47] Y. Zhang and J. R. Glass, "Towards multi-speaker unsupervised speech pattern discovery," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010, pp. 4366–4369.

[48] C.-H. Lee, F. K. Soong, and B.-H. Juang, "A segment model based approach to speech recognition," in *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*. IEEE Computer Society, 1988, pp. 501–502.

[49] H. Wang, C.-C. Leung, T. Lee, B. Ma, and H. Li, "An acoustic segment modeling approach to query-by-example spoken term detection," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5157–5160.

[50] C.-y. Lee and J. Glass, "A nonparametric bayesian approach to acoustic model discovery," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2012, pp. 40–49.

[51] H. Kamper, A. Jansen, and S. Goldwater, "Unsupervised word segmentation and lexicon discovery using acoustic word embeddings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 4, pp. 669–679, 2016.

[52] M. Elsner and C. Shain, "Speech segmentation with a neural encoder model of working memory," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1070–1080.

doi:10.6342/NTU202302463

[53] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin *et al.*, "Superb: Speech processing universal performance benchmark," *arXiv preprint arXiv:2105.01051*, 2021.

[54] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. R. Glass, "An unsupervised autoregressive model for speech representation learning," in *INTERSPEECH*, 2019.

[55] Y.-A. Chung, H. Tang, and J. Glass, "Vector-quantized autoregressive predictive coding," *Proc. Interspeech 2020*, pp. 3760–3764, 2020.

[56] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6419–6423.

[57] A. T. Liu, S.-W. Li, and H.-y. Lee, "Tera: Self-supervised learning of transformer encoder representation for speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2351–2366, 2021.

[58] A. H. Liu, Y.-A. Chung, and J. Glass, "Non-autoregressive predictive coding for learning speech representations from local dependencies," *arXiv preprint arXiv:2011.00406*, 2020.

[59] S. Ling and Y. Liu, "Decoar 2.0: Deep contextualized acoustic representations with vector quantization," *arXiv preprint arXiv:2012.06659*, 2020.

[60] A. Van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv e-prints*, pp. arXiv–1807, 2018.

doi:10.6342/NTU202302463

[61] M. Riviere, A. Joulin, P.-E. Mazaré, and E. Dupoux, “Unsupervised pretraining transfers well across languages,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7414–7418.

[62] S. Schneider, A. Baevski, R. Collobert, and M. Auli, “wav2vec: Unsupervised pre-training for speech recognition,” *Proc. Interspeech 2019*, pp. 3465–3469, 2019.

[63] A. Baevski, S. Schneider, and M. Auli, “vq-wav2vec: Self-supervised learning of discrete speech representations,” in *International Conference on Learning Representations*, 2019.

[64] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[65] W.-N. Hsu, Y.-H. H. Tsai, B. Bolte, R. Salakhutdinov, and A. Mohamed, “Hubert: How much can a bad teacher benefit asr pre-training?” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6533–6537.

[66] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, and Y. Bengio, “Learning problem-agnostic speech representations from multiple self-supervised tasks,” *Proc. Interspeech 2019*, pp. 161–165, 2019.

[67] M. Ravanelli, J. Zhong, S. Pascual, P. Swietojanski, J. Monteiro, J. Trmal, and Y. Bengio, “Multi-task self-supervised learning for robust speech recognition,” in

doi:10.6342/NTU202302463

ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 6989–6993.

[68] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[69] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[70] K. Rao and H. Sak, "Multi-accent speech recognition with hierarchical grapheme based models," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP).* IEEE, 2017, pp. 4815–4819.

[71] H. Lin, L. Deng, D. Yu, Y.-f. Gong, A. Acero, and C.-H. Lee, "A study on multilingual acoustic modeling for large vocabulary asr," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 4333–4336.

[72] M. Elfeky, M. Bastani, X. Velez, P. Moreno, and A. Waters, "Towards acoustic model unification across dialects," in *2016 IEEE Spoken Language Technology Workshop (SLT).* IEEE, 2016, pp. 624–628.

[73] H. Kamper and T. Niesler, "Multi-accent speech recognition of afrikaans, black and white varieties of south african english," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

doi:10.6342/NTU202302463

[74] D. Vergyri, L. Lamel, and J.-L. Gauvain, “Automatic speech recognition of multiple accented english data,” in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[75] X. Yang, K. Audhkhasi, A. Rosenberg, S. Thomas, B. Ramabhadran, and M. Hasegawa-Johnson, “Joint modeling of accents and acoustics for multi-accent speech recognition,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 1–5.

[76] B. Li, T. N. Sainath, K. C. Sim, M. Bacchiani, E. Weinstein, P. Nguyen, Z. Chen, Y. Wu, and K. Rao, “Multi-dialect speech recognition with a single sequence-to-sequence model,” in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 4749–4753.

[77] M. Chen, Z. Yang, J. Liang, Y. Li, and W. Liu, “Improving deep neural networks based multi-accent mandarin speech recognition using i-vectors and accent-specific top layer,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[78] S. Yoo, I. Song, and Y. Bengio, “A highly adaptive acoustic model for accurate multi-dialect speech recognition,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5716–5720.

[79] N. T. Vu, D. Imseng, D. Povey, P. Motlicek, T. Schultz, and H. Bourlard, “Multilingual deep neural network based acoustic modeling for rapid language adaptation,”

doi:10.6342/NTU202302463

in *2014 IEEE international Conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2014, pp. 7639–7643.

[80] Y. Huang, D. Yu, C. Liu, and Y. Gong, "Multi-accent deep neural network acoustic model with accent-specific top layer using the kld-regularized model adaptation," in *Fifteenth Annual Conference of the International Speech Communication Association*. Citeseer, 2014.

[81] Y.-C. Chen, S.-F. Huang, C.-H. Shen, H.-Y. Lee, and L.-S. Lee, "Phonetic-and-semantic embedding of spoken words with applications in spoken content retrieval," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 941–948.

[82] D. Serdyuk, K. Audhkhasi, P. Brakel, B. Ramabhadran, S. Thomas, and Y. Bengio, "Invariant representations for noisy speech recognition," *arXiv preprint arXiv:1612.01928*, 2016.

[83] S. Sun, C.-F. Yeh, M.-Y. Hwang, M. Ostendorf, and L. Xie, "Domain adversarial training for accented speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4854–4858.

[84] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3722–3731.

doi:10.6342/NTU202302463

[85] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, “One model to learn them all,” *arXiv preprint arXiv:1706.05137*, 2017.

[86] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, “The natural language decathlon: Multitask learning as question answering,” *arXiv preprint arXiv:1806.08730*, 2018.

[87] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *arXiv preprint arXiv:1910.10683*, 2019.

[88] Z. Chen, S. Watanabe, H. Erdogan, and J. R. Hershey, “Speech enhancement and recognition using multi-task learning of long short-term memory recurrent neural networks,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[89] A. Tjandra, S. Sakti, and S. Nakamura, “Listening while speaking: Speech chain by deep learning,” in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 301–308.

[90] Y. Ren, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, “Almost unsupervised text to speech and automatic speech recognition,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 5410–5419.

[91] A. Jain, M. Upreti, and P. Jyothi, “Improved accented speech recognition using accent embeddings and multi-task learning.” in *Interspeech*, 2018, pp. 2454–2458.

doi:10.6342/NTU202302463

[92] Y.-C. Chen, Z. Yang, C.-F. Yeh, M. Jain, and M. L. Seltzer, "Aipnet: Generative adversarial pre-training of accent-invariant networks for end-to-end speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6979–6983.

[93] Z. Tang, L. Li, and D. Wang, "Multi-task recurrent model for speech and speaker recognition," in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2016, pp. 1–4.

[94] J.-Y. Hsu, Y.-J. Chen, and H.-y. Lee, "Meta learning for end-to-end low-resource speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7844–7848.

[95] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. Lopez Moreno, Y. Wu *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," *Advances in neural information processing systems*, vol. 31, 2018.

[96] J.-X. Zhang, Z.-H. Ling, Y. Jiang, L.-J. Liu, C. Liang, and L.-R. Dai, "Improving sequence-to-sequence voice conversion by adding text-supervision," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6785–6789.

[97] T. Kinnunen, L. Juvela, P. Alku, and J. Yamagishi, "Non-parallel voice conversion using i-vector plda: Towards unifying speaker verification and transformation," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 5535–5539.

doi:10.6342/NTU202302463

[98] M. Zhang, X. Wang, F. Fang, H. Li, and J. Yamagishi, "Joint training framework for text-to-speech and voice conversion using multi-source tacotron and wavenet," *Proc. Interspeech 2019*, pp. 1298–1302, 2019.

[99] J.-X. Zhang, Z.-H. Ling, and L.-R. Dai, "Non-parallel sequence-to-sequence voice conversion with disentangled linguistic and speaker representations," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 540–552, 2019.

[100] S. Tong, P. N. Garner, and H. Bourlard, "An investigation of deep neural networks for multilingual speech recognition training and adaptation," in *Proc. of INTERSPEECH*, no. CONF, 2017.

[101] J. Cho, M. K. Baskar, R. Li, M. Wiesner, S. H. Mallidi, N. Yalta, M. Karafiat, S. Watanabe, and T. Hori, "Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 521–527.

[102] S. Tong, P. N. Garner, and H. Bourlard, "Multilingual training and cross-lingual adaptation on ctc-based acoustic model," *arXiv preprint arXiv:1711.10025*, 2017.

[103] J. Yi, J. Tao, Z. Wen, and Y. Bai, "Adversarial multilingual training for low-resource speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4899–4903.

[104] O. Adams, M. Wiesner, S. Watanabe, and D. Yarowsky, "Massively multilingual adversarial speech recognition," in *Proceedings of the 2019 Conference of the North*

doi:10.6342/NTU202302463

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 96–108.

[105] R. Sanabria and F. Metze, "Hierarchical multitask learning with ctc," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 485–490.

[106] S. Dalmia, R. Sanabria, F. Metze, and A. W. Black, "Sequence-based multi-lingual low resource speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4909–4913.

[107] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[108] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[109] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[110] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 826–834, 1983.

doi:10.6342/NTU202302463

[111] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[112] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[113] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4780–4789.

[114] A. Baruwa, M. Abisiga, I. Gbadegesin, and A. Fakunle, "Leveraging end-to-end speech recognition with neural architecture search," *arXiv preprint arXiv:1912.05946*, 2019.

[115] T. Véniat, O. Schwander, and L. Denoyer, "Stochastic adaptive neural architecture search for keyword spotting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2019, pp. 2842–2846.

[116] H. Mazzawi, X. Gonzalvo, A. Kracun, P. Sridhar, N. Subrahmanya, I. Lopez-Moreno, H.-J. Park, and P. Violette, "Improving keyword spotting and language identification via neural architecture search at scale." in *Interspeech*, 2019, pp. 1278–1282.

[117] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *International Conference on Learning Representations*, 2018.

doi:10.6342/NTU202302463

[118] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 734–10 742.

[119] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," in *International Conference on Learning Representations*, 2018.

[120] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1294–1303.

[121] Y.-C. Chen, S.-F. Huang, H.-y. Lee, Y.-H. Wang, and C.-H. Shen, "Audio word2vec: Sequence-to-sequence autoencoding for unsupervised learning of audio segmentation and representation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 9, pp. 1481–1493, 2019.

[122] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," *Advances in neural information processing systems*, vol. 28, 2015.

[123] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," *Advances in neural information processing systems*, vol. 30, 2017.

[124] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems*, vol. 29, 2016.

doi:10.6342/NTU202302463

[125] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[126] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," *Advances in neural information processing systems*, vol. 29, 2016.

[127] Z. Meng, Z. Chen, V. Mazalov, J. Li, and Y. Gong, "Unsupervised adaptation with domain separation networks for robust speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU).* IEEE, 2017, pp. 214–221.

[128] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP).* IEEE, 2018, pp. 126–130.

[129] L.-s. Lee, J. Glass, H.-y. Lee, and C.-a. Chan, "Spoken content retrieval—beyond cascading speech recognition with text retrieval," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 9, pp. 1389–1420, 2015.

[130] B. Chen, K.-Y. Chen, P.-N. Chen, and Y.-W. Chen, "Spoken document retrieval with unsupervised query modeling techniques," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 9, pp. 2602–2612, 2012.

doi:10.6342/NTU202302463

[131] P.-N. Chen, K.-Y. Chen, and B. Chen, "Leveraging relevance cues for improved spoken document retrieval," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[132] H.-y. Lee, Y.-C. Li, C.-T. Chung, and L.-s. Lee, "Enhancing query expansion for semantic retrieval of spoken content with automatically discovered acoustic patterns," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8297–8301.

[133] T.-H. Wen, H.-Y. Lee, P.-h. Su, and L.-S. Lee, "Interactive spoken content retrieval by extended query model and continuous state space markov decision process," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8510–8514.

[134] G. Lample, A. Conneau, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data," in *International Conference on Learning Representations*, 2018.

[135] Y. Hoshen and L. Wolf, "An iterative closest point method for unsupervised word translation," *arXiv preprint arXiv:1801.06126*, vol. 3, 2018.

[136] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[137] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

doi:10.6342/NTU202302463

[138] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[139] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, 2013.

[140] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[141] M. A. Hasegawa-Johnson, P. Jyothi, D. McCloy, M. Mirbagheri, G. M. Di Liberto, A. Das, B. Ekin, C. Liu, V. Manohar, H. Tang *et al.*, "Asr for under-resourced languages from probabilistic transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 50–63, 2016.

[142] N. F. Chen, B. P. Lim, M. A. Hasegawa-Johnson *et al.*, "Multitask learning for phone recognition of underresourced languages using mismatched transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 3, pp. 501–514, 2017.

[143] A. Jansen, E. Dupoux, S. Goldwater, M. Johnson, S. Khudanpur, K. Church, N. Feldman, H. Hermansky, F. Metze, R. Rose *et al.*, "A summary of the 2012 jhu clsp workshop on zero resource speech technologies and models of early language acquisition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8111–8115.

doi:10.6342/NTU202302463

[144] E. Dunbar, X. N. Cao, J. Benjumea, J. Karadayi, M. Bernard, L. Besacier, X. Anguera, and E. Dupoux, "The zero resource speech challenge 2017," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 323–330.

[145] M. Versteegh, R. Thiolliere, T. Schatz, X. N. Cao, X. Anguera, A. Jansen, and E. Dupoux, "The zero resource speech challenge 2015," in *Sixteenth annual conference of the international speech communication association*, 2015.

[146] C.-H. Shen, J. Y. Sung, and H.-Y. Lee, "Language transfer of audio word2vec: Learning audio segment representations without target language data," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2231–2235.

[147] C.-T. Chung, C.-a. Chan, and L.-s. Lee, "Unsupervised spoken term detection with spoken queries by multi-level acoustic patterns with varying model granularity," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 7814–7818.

[148] Y. Zhang and J. R. Glass, "Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams," in *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 2009, pp. 398–403.

[149] Y.-H. Wang, C.-T. Chung, and H.-Y. Lee, "Gate activation signal analysis for gated recurrent neural networks and its correlation with phoneme boundaries," *Proc. Interspeech 2017*, pp. 3822–3826, 2017.

doi:10.6342/NTU202302463

[150] O. Rasanen, “Basic cuts revisited: Temporal segmentation of speech into phone-like units with statistical learning at a pre-linguistic level,” in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 36, no. 36, 2014.

[151] Y. Qiao, N. Shimomura, and N. Minematsu, “Unsupervised optimal phoneme segmentation: Objectives, algorithm and comparisons,” in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 3989–3992.

[152] Y. Bengio, N. Léonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” *arXiv preprint arXiv:1308.3432*, 2013.

[153] J. Chung, S. Ahn, and Y. Bengio, “Hierarchical multiscale recurrent neural networks,” in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[154] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[155] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings, 2012, pp. 37–49.

[156] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, “Generating sentences from a continuous space,” in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 10–21.

doi:10.6342/NTU202302463

[157] S. Semeniuta, A. Severyn, and E. Barth, "A hybrid convolutional variational autoencoder for text generation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 627–637.

[158] P. Nema, M. M. Khapra, A. Laha, and B. Ravindran, "Diversity driven attention model for query-based abstractive summarization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1063–1072.

[159] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[160] T. Schultz, "Globalphone: a multilingual speech and text database developed at karlsruhe university," in *Seventh International Conference on Spoken Language Processing*, 2002.

[161] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

[162] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[163] C.-a. Chan, "Unsupervised spoken term detection with spoken queries," *Ph. D Dissertation, National Taiwan University, July*, 2012.

doi:10.6342/NTU202302463

[164] Y.-C. Chen, S.-w. Yang, C.-K. Lee, S. See, and H.-y. Lee, "Speech representation learning through self-supervised pretraining and multi-task finetuning," *AAAI 2022 Workshop on Self-supervised Learning for Audio and Speech Processing*, 2021.

[165] Y.-C. Chen, P.-H. Chi, S.-w. Yang, K.-W. Chang, J.-h. Lin, S.-F. Huang, D.-R. Liu, C.-L. Liu, C.-K. Lee, and H.-y. Lee, "Speechnet: A universal modularized model for speech processing tasks," *arXiv preprint arXiv:2105.03070*, 2021.

[166] Y.-C. Chen, J.-Y. Hsu, C.-K. Lee, and H. yi Lee, "DARTS-ASR: Differentiable Architecture Search for Multilingual Speech Recognition and Adaptation," in *Proc. Interspeech 2020*, 2020, pp. 1803–1807.

[167] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[168] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 794–803.

[169] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[170] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

doi:10.6342/NTU202302463

[171] Z. Wang, Y. Tsvetkov, O. Firat, and Y. Cao, "Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models," in *International Conference on Learning Representations*, 2020.

[172] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European conference on computer vision*. Springer, 2012, pp. 746–760.

[173] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[174] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018, pp. 353–355.

[175] P. Warden, "Speech commands: A public dataset for single-word speech recognition," *Dataset available from http://download. tensorflow. org/-data/speech_commands_v0*, vol. 1, 2017.

[176] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language*, vol. 60, p. 101027, 2020.

[177] J. Cosentino, M. Pariente, S. Cornell, A. Deleforge, and E. Vincent, "Librimix: An open-source dataset for generalizable speech separation," *arXiv preprint arXiv:2005.11262*, 2020.

doi:10.6342/NTU202302463

[178] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio, “Speech model pre-training for end-to-end spoken language understanding,” *Proc. Interspeech 2019*, pp. 814–818, 2019.

[179] C.-I. Lai, Y.-S. Chuang, H.-Y. Lee, S.-W. Li, and J. Glass, “Semi-supervised spoken language understanding via self-supervised speech and language model pretraining,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7468–7472.

[180] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, “Iemocap: Interactive emotional dyadic motion capture database,” *Language resources and evaluation*, vol. 42, no. 4, pp. 335–359, 2008.

[181] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, “Multi-task learning for dense prediction tasks: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[182] Y. Zhang and Q. Yang, “A survey on multi-task learning,” *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[183] S. Ruder, “An overview of multi-task learning in deep neural networks,” *arXiv preprint arXiv:1706.05098*, 2017.

[184] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, “Fastspeech: Fast, robust and controllable text to speech,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

doi:10.6342/NTU202302463

[185] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang *et al.*, "A comparative study on transformer vs rnn in speech applications," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 449–456.

[186] A. H. Liu, H.-y. Lee, and L.-s. Lee, "Adversarial training of end-to-end speech recognition using a criticizing language model," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6176–6180.

[187] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[188] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *Proc. Interspeech 2020*, pp. 5036–5040, 2020.

[189] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5884–5888.

[190] J. Kim, M. El-Khamy, and J. Lee, "T-gsa: Transformer with gaussian-weighted self-attention for speech enhancement," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6649–6653.

doi:10.6342/NTU202302463

[191] S.-W. Fu, C.-F. Liao, T.-A. Hsieh, K.-H. Hung, S.-S. Wang, C. Yu, H.-C. Kuo, R. E. Zezario, Y.-J. Li, S.-Y. Chuang *et al.*, "Boosting objective scores of a speech enhancement model by metricgan post-processing," in *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2020, pp. 455–459.

[192] A. Nicolson and K. K. Paliwal, "Masked multi-head self-attention for causal speech enhancement," *Speech Communication*, vol. 125, pp. 80–96, 2020.

[193] S. V. Katta, S. Umesh *et al.*, "S-vectors: Speaker embeddings based on transformer's encoder for text-independent speaker verification," *arXiv preprint arXiv:2008.04659*, 2020.

[194] P. Safari, M. India, and J. Hernando, "Self-attention encoding and pooling for speaker recognition," *Proc. Interspeech 2020*, pp. 941–945, 2020.

[195] Y. Shi, M. Chen, Q. Huang, and T. Hain, "T-vectors: Weakly supervised speaker identification using hierarchical transformer model," *arXiv preprint arXiv:2010.16071*, 2020.

[196] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," in *International Conference on Learning Representations*, 2020.

[197] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

doi:10.6342/NTU202302463

[198] Z. Zeng, J. Wang, N. Cheng, T. Xia, and J. Xiao, “Aligntts: Efficient feed-forward text-to-speech system without explicit alignment,” in *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP).* IEEE, 2020, pp. 6714–6718.

[199] W.-C. Huang, T. Hayashi, Y.-C. Wu, H. Kameoka, and T. Toda, “Voice transformer network: Sequence-to-sequence voice conversion using transformer with text-to-speech pretraining,” *Proc. Interspeech 2020*, pp. 4676–4680, 2020.

[200] R. Liu, X. Chen, and X. Wen, “Voice conversion with transformer network,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2020, pp. 7759–7759.

[201] Y. Y. Lin, C.-M. Chien, J.-H. Lin, H.-y. Lee, and L.-s. Lee, “Fragmentvc: Any-to-any voice conversion by end-to-end extracting and fusing fine-grained voice fragments with attention,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2021, pp. 5939–5943.

[202] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, “A structured self-attentive sentence embedding,” *arXiv preprint arXiv:1703.03130*, 2017.

[203] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, “Self-attentive speaker embeddings for text-independent speaker verification.” in *Interspeech*, vol. 2018, 2018, pp. 3573–3577.

[204] I. Loshchilov and F. Hutter, “Fixing weight decay regularization in adam,” 2018.

doi:10.6342/NTU202302463

[205] G. Hu and D. Wang, "A tandem algorithm for pitch estimation and voiced speech segregation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 8, pp. 2067–2079, 2010.

[206] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs," in *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, vol. 2. IEEE, 2001, pp. 749–752.

[207] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in *2010 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2010, pp. 4214–4217.

[208] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," *Proc. Interspeech 2017*, pp. 2616–2620, 2017.

[209] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "Libritts: A corpus derived from librispeech for text-to-speech," *Proc. Interspeech 2019*, pp. 1526–1530, 2019.

[210] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi." in *Interspeech*, vol. 2017, 2017, pp. 498–502.

[211] J. Kominek and A. W. Black, "The cmu arctic speech databases," in *Fifth ISCA workshop on speech synthesis*, 2004.

doi:10.6342/NTU202302463

[212] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, no. 2, pp. 236–243, 1984.

[213] R. Sennrich, B. Haddow, and A. 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)*, 2016, pp. 1715–1725.

[214] M. J. Gales, K. M. Knill, A. Ragni, and S. P. Rath, "Speech recognition and keyword spotting for low-resource languages: Babel project research at cued," in *Fourth International workshop on spoken language technologies for under-resourced languages (SLTU-2014)*. International Speech Communication Association (ISCA), 2014, pp. 16–23.

[215] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N.-E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "Espnet: End-to-end speech processing toolkit," *Proc. Interspeech 2018*, pp. 2207–2211, 2018.

[216] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[217] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

doi:10.6342/NTU202302463

[218] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[219] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

doi:10.6342/NTU202302463