Continually learning new languages

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation

von
NGOC QUAN PHAM
aus Hanoi (Vietnam)

Tag der mündlichen Prüfung: 02.02.2023
Erster Gutachter: Prof. Dr. Alexander Waibel
Zweiter Gutachter: Prof. Dr. Shinji Watanabe

Abstract

Speech recognition is the task of automatically generating transcriptions of given speech utterances. This research area is widely applied in our daily lives such as helping foreign students in lectures or controlling intelligent devices such as smart TVs or cars with speech commands.

The 7000 languages being spoken in the world poses a challenge to speech recognition system. Traditionally speech recognition methods using Hidden Markov Model are not practically applicable for many languages simultaneously due to the requirement of pronunciation dictionaries together with a pipeline of separated components. The neural end-to-end approach relaxed both of these requirements by using a single neural network to learn a direct mapping from speech signal inputs to word surface outputs, and all components in the network are directly optimized for this learning objective. This nature enables training one model to recognize many languages at the same time.

The desiderata of a multilingual recognition system includes the following factors:

- The training procedure with a combined dataset of multiple languages should not be overly complicated than dealing with one single language, and is much more industrially efficient than constructing multiple monolingual recognition system.
- The recognition quality of the multilingual system should be competitive or better than the monolingual individual systems. Since many languages share similar acoustic features, the architecture of the recognizers should reflect this nature, while having room to model each language's unique traits.

- It is crucial for the system to be expandable, by being able to continually learn new languages. The intervention of new languages should not affect the performance for the previously learned languages. Meanwhile, the quality of these new languages must be competitive compared to the ideal scenario of having all languages at once. This factor is important when scaling recognition systems in a world-wide setting, in which acquiring data for new languages is difficult and data storing is not eternal due to storage requirement and privacy issues.

The goals of the thesis are designed to further develop speech recognition towards such desiderata. The first objective was achieved by constructing a neural end-to-end speech recognition model. The motivation here was that, due to the high level of abstraction compared to traditional methods, such as the removal of dictionaries and concrete alignment learning, the performance of neural end-to-end models was still inferior to traditional system. By using **very deep Transformer networks** with stochastic layers and enhanced with relative position self attention mechanism, it is possible to reach a competitive level performance in standard benchmark of conversational English, which was the state-of-the-art result achieved with end-to-end models.

Having achieved a strong neural end-to-end model, the next objective is to apply it for large scale multilingual recognition. Because this approach is not limited by the language specific pronunciation dictionaries, it is trivial to create a multilingual system of dozens of languages, that satisfied the first desiderata objective. In order to have a clear learning strategy that separates language dependent and independent features, we proposed **weight factorization** as a technique that factorizes each weight matrix in the network into language dependent and independent factors. The language dependent weights are further factorized to encourage the network to learn universal features. Furthermore, this technique can be combined with **transfer learning**, as a result reducing the word error rate for 32 languages by 33% compared to a competitive Transformer baseline.

With weight factorization, **continual learning** new languages is now enabled. The networks can allocate new weights for new languages without interfering the logic of old languages, thus completely avoiding

catastrophic forgetting, the major problem of losing knowledge of previous languages when training on new ones. Weight factorization, a form of progressive neural networks, can be also combined with Elastic Weight Consolidation, a regularization method to prevent catastrophic forgetting. This combination makes the whole network architecture more flexible, by finding empty space in the network to learn new knowledge. By doing so, it is possible to learn new languages with the same quality of having all languages initially, without severely compromising in terms of catastrophic forgetting.

The last contribution of the thesis is the application in *direct speech translation*. For many languages, it is more convenient to collect the translation instead of transcript for speech utterances. The same neural architecture can be applied in this situation without any modification. The contribution is hightlighted by showing that, a neural model is now powerful enough to outperform cascaded approaches, in either large scale speech translation or resource-limited multilingual speech translation scenarios.

Zusammenfassung

Spracherkennung ist die Aufgabe, automatisch Transkriptionen von gegebenen Sprachäußerungen zu erstellen. Dieser Forschungsbereich findet in unserem täglichen Leben breite Anwendung, z. B. bei der Unterstützung ausländischer Studierenden in Vorlesungen oder bei der Steuerung intelligenter Geräte wie Smart-TVs oder Autos durch Sprachbefehle.

Die 7000 Sprachen, die auf der Welt gesprochen werden, stellen eine Herausforderung für Spracherkennungssysteme dar. Herkömmliche Spracherkennungsmethoden, die Hidden Markov-Modelle verwenden, sind in der Praxis nicht für viele Sprachen gleichzeitig anwendbar, da sie Aussprachewörterbücher und eine Pipeline von getrennten Komponenten erfordern. Der neuronale End-to-End-Ansatz entspannt diese beiden Anforderungen, indem ein einziges neuronales Netzwerk verwendet wird, um eine direkte Abbildung von Sprachsignaleingaben auf Wortoberflächenausgaben zu erlernen, und alle Komponenten im Netzwerk werden direkt für dieses Lernziel optimiert. Auf diese Weise kann ein einziges Modell trainiert werden, um viele Sprachen gleichzeitig zu erkennen.

Zu den Desideraten eines mehrsprachigen Erkennungssystems gehören die folgenden Faktoren:

- Das Trainingsverfahren mit einem kombinierten Datensatz aus mehreren Sprachen sollte nicht übermäßig kompliziert sein und ist industriell viel effizienter als die Entwicklung mehrerer einsprachiger Erkennungssysteme.
- Die Erkennungsqualität des mehrsprachigen Systems sollte konkurrenzfähig oder besser sein als die der einsprachigen Einzelsysteme. Da viele Sprachen ähnliche akustische

Merkmale aufweisen, sollte die Architektur der Erkennungssysteme dies widerspiegeln und gleichzeitig Raum für die Modellierung der einzigartigen Merkmale der einzelnen Sprachen lassen.

- Entscheidend ist, dass das System erweiterungsfähig ist, d. h. dass es ständig neue Sprachen lernen kann. Die Einführung neuer Sprachen sollte die Leistung der bereits erlernten Sprachen nicht beeinträchtigen. Gleichzeitig muss die Qualität dieser neuen Sprachen im Vergleich zum idealen Szenario, in dem alle Sprachen auf einmal vorhanden sind, konkurrenzfähig sein. Dieser Faktor ist wichtig für die Skalierung von Erkennungssystemen in einem weltweiten Umfeld, in dem die Beschaffung von Daten für neue Sprachen schwierig ist und die Datenspeicherung aufgrund von Speicherbedarf und Datenschutzproblemen nicht ewig möglich ist.

Die Ziele dieser Arbeit sind darauf ausgerichtet, die Spracherkennung in Richtung dieser Desiderate weiterzuentwickeln. Das erste Ziel wurde durch die Konstruktion eines neuronalen End-to-End-Spracherkennungsmodells erreicht. Die Motivation hierfür war, dass aufgrund des hohen Abstraktionsniveaus im Vergleich zu traditionellen Methoden, wie z.B. der Entfernung von Wörterbüchern und konkretem Alignment-Lernen, die Leistung von neuronalen End-to-End-Modellen immer noch schlechter war als bei traditionellen Systemen. Durch die Verwendung von **sehr tiefen Transformer-Netzen** mit stochastischen Schichten und einem Mechanismus für relative Positionsselbstaufmerksamkeit ist es möglich, eine konkurrenzfähige Leistung in Standard-Benchmarks für Konversations-Englisch zu erreichen, was das modernste Ergebnis war, das mit End-to-End-Modellen erzielt wurde.

Nachdem ein starkes neuronales End-to-End-Modell erreicht wurde, besteht das nächste Ziel darin, es für eine groß angelegte mehrsprachige Erkennung einzusetzen. Da dieser Ansatz nicht durch sprachspezifische Aussprachewörterbücher eingeschränkt ist, ist es trivial, ein mehrsprachiges System mit Dutzenden von Sprachen zu erstellen, das das erste Desiderat-Ziel erfüllt. Um eine klare Lernstrategie zu haben, die sprachabhängige und unabhängige Merkmale trennt, haben

wir die **Gewichtsfaktorisierung** als Technik vorgeschlagen, die jede Gewichtsmatrix im Netzwerk in sprachabhängige und unabhängige Faktoren zerlegt. Die sprachabhängigen Gewichte werden weiter faktorisiert, um das Netzwerk zu ermutigen, universelle Merkmale zu lernen. Darüber hinaus kann diese Technik mit **Transfer-Lernen** kombiniert werden, wodurch die Wortfehlerrate für 32 Sprachen um 33% im Vergleich zu einer konkurrierenden Transformer-Baseline reduziert wird.

Mit der Gewichtungsfaktorisierung ist nun das **kontinuierliche Lernen** neuer Sprachen möglich. Die Netze können neue Gewichte für neue Sprachen zuweisen, ohne in die Logik der alten Sprachen einzugreifen, wodurch ein katastrophales Vergessen, das Hauptproblem des Verlusts von Wissen über frühere Sprachen beim Training neuer Sprachen, vollständig vermieden wird. Die Gewichtsfaktorisierung, eine Form der progressiven neuronalen Netze, kann auch mit der elastischen Gewichtskonsolidierung kombiniert werden, einer Regularisierungsmethode zur Vermeidung des katastrophalen Vergessens. Diese Kombination macht die gesamte Netzarchitektur flexibler, da sie Leerstellen im Netz findet, um neues Wissen zu lernen. Auf diese Weise ist es möglich, neue Sprachen mit der gleichen Qualität zu erlernen, wie wenn man alle Sprachen von Anfang an beherrschen würde, ohne dass es zu ernsthaften Kompromissen in Bezug auf das katastrophale Vergessen kommt.

Der letzte Beitrag der Arbeit ist die Anwendung im Bereich der direkten Sprachübersetzung. Für viele Sprachen ist es bequemer, die Übersetzung anstelle des Transkripts von Sprachäußerungen zu sammeln. Die gleiche neuronale Architektur kann in dieser Situation ohne jegliche Modifikation angewendet werden. Der Beitrag wird dadurch hervorgehoben, dass gezeigt wird, dass ein neuronales Modell jetzt leistungsfähig genug ist, um kaskadierte Ansätze zu übertreffen, entweder in groß angelegten Sprachübersetzungen oder in ressourcenbeschränkten mehrsprachigen Sprachübersetzungsszenarien.

Acknowledgement

During the time of writing this acknowledgement, in the background was the beautiful piano concerto in B-flat major of Johannes Brahms. As Brahms himself considered his piano concerto as a tiny piano concerto, I also think about my disseration as a tiny tiny Ph.D thesis with a tiny tiny contribution. The former is indeed a description that cannot be further than the truth, but the latter is truly what I think about my thesis. Maybe it is an inevitable feeling given how vast the world is and how fast the deep learning fields are progressing.

Nevertheless, accomplishing this tiny tiny achievement is impossible without the people that I want to acknowledge here. The most important ones, are absolutely my parents and professor Alexander Waibel. For my parents, I simply cannot wait for the moment that I could finish the thesis, so that I can play for them the favourite Etude “Tritesse” and the Barcarolle of Chopin. I would like to express my gratitude to professor Waibel, professor Watanabe and professor Niehues for helping me in the process of finishing this work.

I came to the ISL lab at the transition between two generations. The previous generation consists of Jan, Kevin, Marcus, Matthias, Teresa, Eunah, Bastian, Le Thai-Son and others during the time of the statistical approaches in both Speech Recognition and Machine Translation field. Elizabeth, Thomas and myself can probably be thrown into this generation as well, especially if those two had stayed here eventually. The next generation starts from Stefan, and then Christian, Leonard, Dogucan and Irem, Enes, Fabian, Carlos, Ilya, Dan and now many other colleagues coming to join after Jan came back as a professor. I can feel an unspeakable difference between the two, like how Brahms ended his first movement in the concerto in a grandiose way, just to start a "scherzo" (joking) like movement, followed by a queiter one.

I thank all of the colleagues for the day-by-day struggles spent with each other and that makes everyone's life colorful.

I also want to thank the little Vietnamese community here including Thanh-Le, Ngoc-Linh and their children Ngoc-Nhi and Minh-Khanh, Thai-Son, Phuong-Nhung and their children Nhu-Van and Nam-Phong, my new Vietnamese colleagues Tuan-Nam and Thai Binh and more Vietnamese people around that helped me everytime with the nostalgic feelings. My biggest thanks for my landlord and landlady who provided me with shelters and even meals during the pandemic, and also to my friend Hoang-Bui for giving me musical lessons in the first three years. My deepest thanks for Ha-Giang for the cat gift.

And of course, during the dark times of a "Blight"[1] and a restless war happening just from 2000 kilometers away, I have received help from various people some of whom I could not ask for the name. My very best thank yous for you all.

Last but not least, I was lucky to receive a tremendous and unexpected amount of love from Z.H. Without her, this Thesis would not be mentally possible.

[1] A global pandemic as being called in Dragon Age: Origins.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context and motivation

In 2019, humanity witnessed a global pandemic happening once every century, with the last known pandemic happened before the first World War. Due to this catastrophic event, the human society has to witness a rearrangement with measures to counteract the transmitting viruses. Social distancing, home office and various measures were put into practice, and the main strategy is to reduce close communication replaced by telecommunication technologies. Despite of the crisis entailing the pandemic, there are actual research and engineering areas that bloom from the new world in which telecommunication became more importantly dominant. The increasing amount of indirect communication via the Internet has made way for new applications to thrive, with *speech recognition* and *translation* at the core. Such applications are being used ubiquitously in video conferencing, home office or online lectures. This new order raises new demand in recording and transcribing speech automatically.

At the time of this manuscript, automatic speech recognition (ASR) has witnessed more than half a century of development. This research area has been considered as an interdisciplinary research sub-field of computer science and computational linguistics with the purpose of generating the transcription of the spoken utterance. For a long time, the dominant solution in speech recognition has been following the statistical approach, with a Hidden Markov Model (HMM) at the core and using statistical models to estimate the likelihood of generating the observation (the acoustic signals) from a latent sequence (often using acoustic units). Such approach can be characterized by a pipeline of distinct models, as illustrated in Figure 1.1. In this pipeline, many

component designs are based on *handcrafted features* including *feature representation* based on Mel Frequency Cepstral or the latent representation being acoustic units such as phoneme states. Even though the major components are machine learning based and can benefit from state-of-the-art machine learning methods such as deep neural networks (DNN), they are optimised towards different learning objectives. The presence of many separated components requires a complicated search process at the end of the pipeline to find an optimized solution by balancing the outputs of each probabilistic component. Under this approach, speech recognition systems have gradually been improved to reach acceptable performance for applications and there even are claims about reaching human parity [273] in English conversational speech.

English, however, is one of about 7100 languages being spoken within human societies. In the bigger picture of speech recognition applications, it is beneficial to handle multiple languages with one single system. On the industrial side, building on system on the combination of datasets for multiple languages reduces the effort significantly compared to the efforts of constructing separated systems. The cost in time and energy in building one system is not necessarily smaller than the multilingual one [3, 9]. On the other hand, it is even possible to improve the performance of languages individually in a multilingual system thanks to many assumptions transferable between languages [230]. Apart from the cost of learning language simultaneously, in practice it is necessary to consider the situations in which new languages are added to the collection after training, and the system has to handle without retraining the whole system, a costly option.

Expanding Hidden Markov Model based models into other languages' territory has always been challenging and ineffective. On the one hand, the involvement of a pronunciation dictionary limits the applicability to other languages because of the linguistic effort in constructing one for each new language. On the other hand, there are certain modeling assumptions that can be transferred between languages, for example sharing components between them [88], that cannot be easily manifested in the classical approach. The pipeline nature of the classical approach was simply not appropriate for this task.

The deep learning approach, with neural networks at the core, changes the modeling nature. From a very early time, neural networks were shown to be able to learn powerful features and capable of recognizing speech, despite the small scale [259, 262] at the time due to the limitation in data and computation. After the winter of neural networks at the end of the 20th century, neural networks have reappeared in speech recognition by replacing the neural components in the statistical pipeline. Deep multi-layer perceptrons were shown to be possibly trained for acoustic emission probabilities in HMMs [97] to replace the Gaussian Mixture Models while the latter could also be vastly improved by the neural bottleneck features, obtained by training deep networks with one small hidden layer [77, 69]. Meanwhile, continuous space language models [23, 145, 166] using neural networks dominated the n-gram based language models. Connectionist Temporal Classification [75] (CTC) is a specific algorithm allow for neural networks to even generating output symbols (such as characters or phonemes), despite the assumption that the outputs are independently generated. It is observable that neural networks were playing an increasingly important role in the speech processing picture. So what characteristics do these machine learning models have create such a revolution?

Novel neural architectures such as Encoder-Decoder neural networks [247, 255, 15] were capable of transforming one sequence to another even when those two belong to two different modalities. From a hierarchical standpoint, the encoder and decoder can be considered as acoustic modeling and language modeling components in the stochastic approach, with all of the intermediate layers being abstracted in hidden representations, while the output of the model's search space is word sequences directly. Training this model successfully can be viewed as jointly training different speech recognition components towards the same objective, known as **end-to-end** training.

Aforementioned, multilingual speech recognition development was still infant before the time of this thesis. With the end-to-end neural network approach, the thesis address the following research questions to push multilingual speech recognition into new boundaries:

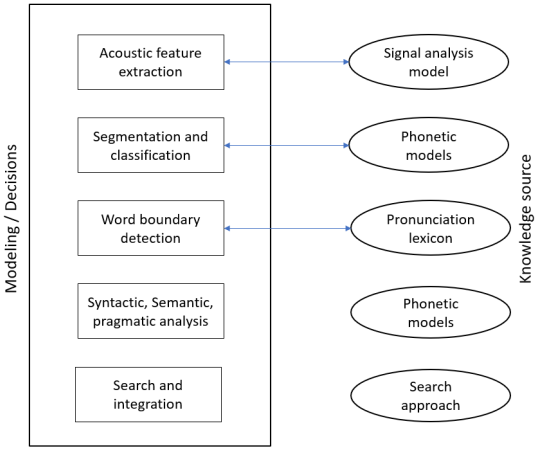


Figure 1.1: An overview of the statistical approach in ASR.

- Can the end-to-end models approach the performance of the statistical HMM-based systems in English recognition? It is notable that the latter one has had decades of development, in which many tricks were used to improve the performance, and also neural networks were used to replace acoustic and language components. It is the best of both world if one can prove the end-to-end approach is superior in both conventional benchmarks as well as being expandable to a multilingual scenario.
- How can we design a model architecture that can handle many languages with the highest efficiency? The literature showed us that multilingual learning in neural networks often employs a semi-shared strategy, in which some components are designed to be language specific, while most of the network components are shared. However, they are often very specific and cannot be applied to state-of-the-art architectures. The desiderata is to be able to multilingual-ize any neural architecture with ease and the highest efficiency.

- When dealing with new languages, neural networks are often suffering from **catastrophic forgetting**, a phenomenon happening when networks fine-tuned on new languages are deteriorated on previously learned languages. How can we design a continual learning strategy to combat this problem when learning new languages?

1.2 Contribution

With those targets in mind, the contribution of the thesis is as follows:

First, we developed a attention-based Encoder-Decoder neural network being able to overcome competitive HMM-based systems in conversational speech recognition. Previously, Encoder-Decoder models were applied to speech recognition [40, 16] but struggled to overcome previous approaches such as HMM or even CTC. By proposing a deep Transformer [255] based network that uses self-attention as the main network component in both encoders and decoders, it is possible to outperform other approaches. The key idea is to construct very deep *encoder* due to the complexity of the acoustic modality, yet at the same time keeping the network regularised and computationally feasible. This is realised by the residual connections in Transformers [90] allowed for layers to be randomly dropped during training, known as Stochastic layers. Furthermore, the relative positions in Transformers are important to deal with high variability in the acoustic inputs.

Second, the success of end-to-end training a neural based speech recognizer is expanded to other languages, enabling multilingual speech recognition. The main concern here is how to fully utilise the multilingual data, with the assumption that most languages can share certain acoustic features but also have distinct qualities [194]. In contrast with the previously proposed model-specific integration of language-specific features, our approach starts from the core of neural networks: feature projection (or also known as matrix multiplication). Each weight matrix in the network can be *factorized* into shared and language-specific parts, in which the latter can be compactly

represented as low-rank matrices. Not only does this method high generalizes other neural modification for multilingual purposes but it also remains network agnostic. The factorized network can enjoy a combination with *unsupervised learning* in which the shared body of the network is pretrained with unlabeled data, while the language specific parts are adapted during multilingual fine-tuning. Such combination managed to significantly outperform a competitive Transformer baseline.

Third, the language specific weights also enable continual learning for multilingual speech recognition. Being exposed to new languages, a typical model often has two choices: fine-tuning for the new languages but suffer from *catastrophic forgetting* - losing the knowledge of the old ones, or retraining with all of the available data. To avoid the costly second option and the catastrophic first option, the factorized weights are potentially useful because they maintain the knowledge about the previously learned languages and are not changed in the subsequent multilingual fine-tuning process. By exploring this direction, we also investigated *elastic weight consolidation*, a regularization technique aiming at reducing the changes in the weights to avoid forgetting. Our experiments indicated that catastrophic forgetting can be mitigated with a minor cost to performance compared to fine-tuning. This is the first work in continual learning for multilingual speech recognition, to the best of our knowledge.

Finally, the neural paradigm shift from statistical approach to end-to-end neural approach is successfully applied on direct speech translation. The main difference between speech translation and speech recognition lies in the originality of the target sequences, that come from another language in speech translation, resulting in difficulty in alignment between source and target sentences. This difference makes direct speech translation a difficult mapping to learn by the neural networks, and is the reason why a cascaded approach breaking down into recognition and translation is preferred when the data is limited. In this thesis, using a single neural network to overcome cascaded became possible thanks for better modeling and better data utilization. This achievement is two fold: first a Transformer based model is able to outperform a competitive cascaded baseline with the

key improvements being *pseudo data generation, transfer learning and data-driven segmentation*. Second, the same result is also observed in low-resourced multilingual translation in which pseudo-data generation is the highlight.

One of the important concepts of deep learning is replacing handcrafted features or manual processes with neural components learning high level representation. These components are connected together and trained with gradient-based optimization via back-propagation [220]. This thesis relied on this principle to revolutionise speech recognition followed by speech translation. Originated from the earlier works of neural speech recognisers [262] and multilingual speech recognition [265] as well as neural factorization [88], this thesis emphasized on the practicality of the neural approach, by empirically proving its superiority and expandability. The works in the thesis are published in conferences [199, 197, 193, 190, 196, 192, 194].

1.3 Thesis Organization

The second chapter in the thesis is dedicated to describe the background related to the main contribution of the thesis. Firstly, a brief description of the statistical speech recognition approach is provided in order to clarify the necessity of a trainable end-to-end approach. The chapter also concerns with several important neural architectures that play important roles in the development of speech recognition system.

The third chapter goes into the details of the first contribution in the thesis, which is speech recognition with very deep Transformer networks, that managed to stay competitive compared to the most prominent Hybrid systems in the literature. Afterwards, the fourth chapter describes the expansion of the network in multilingual speech recognition. The highlight in this chapter includes a factorization method to allocate weights for each language, together with the combination with unsupervised pre-training in order to improve the quality of low-resourced languages. The next chapter contains the details of using weight factorization in combination with elastic weight consolidation for learning new languages for a pre-trained system.

The penultimate chapter modifies the application of the same architecture for speech translation, describing the process of making this end-to-end approach favourable for this problem. Lastly, the thesis is ended with the conclusion and future works that remain problematic in the field of speech recognition, as well as learning representations for speech.

2 Backgrounds

From the introduction in the previous section, the main storyline of this Thesis revolves around evolving the Speech Recognition approach to overcome the weaknessses in the previous counterparts. As such, this literature review section would start from covering the Hidden Markov Model-based approach for ASR, and continue with the Connectionist Temporal Classification approach and finally coming to the end-to-end approach with Neural Networks.

2.1 Overview about Automatic Speech Recognition

The goal of a speech recognition system is to recognize the sequence of written symbols being the origin of the utterance spoken. Typically microphones are used to measure variations in air pressures, record the data and then digitalized into computer-understandable signals (sequences of floating points or integers, with 16000 - 256000 samples taken per second). Until very recently when neural networks are shown to be capable of processing these raw waveforms directly, such raw signals need to be processed into Mel-frequency cepstral coefficients (MFCCs) or perceptual linear predictive coefficients (PLPs). These formats are often shorter and more condensed than raw waveforms to facilitate recognition.

In general, the system receives and views the audio signal as a sequence of feature vectors $X = X_1, X_2, \ldots, X_N$. The task is now to find the word sequence $W = W_1, W_2, \ldots, W_M$ from which the utterance is generated. Or, from a statistical point of view, W is the most probable sequence W among all sequences, that generates the utterance.

$$\hat{W} = \arg\max_{W} P(W|X) \tag{2.1}$$

This is originated from the "Noisy Channel" framework considering the observed sequence (the audio) is an unobserved noisy version of the original input sequence. While the intuition here is straightforward when both sequences are discrete and share the same set of symbols, where one symbol in the output sequence is a corrupted counterpart of one or several tokens in the input sequence,] it can also be applied for continuous speech sequences. This equation suggests that it is necessary to establish a *search space* consists of all possible hypotheses and assign a score for each hypothesis to find the optimal solution. This very idea lies at the core of speech recognition over four decades, and different approaches also aim at approximating the probability distribution and searching for optimal outputs differently.

Early approaches had already been using Neural Networks [260, 261, 262, 259] to learn representation features for phoneme based speech recognition. This approach remains limited and cannot be applied for word-based speech recognition or continous and large vocabulary speech recognition due to lacking a mechanism to find an alignment - which is the task of finding the speech segment corresponding to each phonetic unit (either word, phoneme or sub-word).

During the 1990s, the statistical approach (or the stochastic approach, which is to be explained later) was the dominant approach in ASR. The core component of this approach is the Hidden Markov Models (HMM). Being a generative model, the HMM models the joint distribution of the observation sequence (which is the acoustic signals) and a sequence of *hidden* states being the phonetic tokens W. The joint distribution is maximised after trransforming Equation 2.1 with the Bayes Theorem:

$$\hat{W} = \arg\max_{W} \frac{P(X|W)P(W)}{P(X)} \tag{2.2}$$

Since we are considering the space of possible symbols W, the prior probability $P(X)$ is a constant with given observed speech utterance.

As a result, the problem resorts to finding the optimal words W to maximize only the nominator part by looking for the sequences that achieve the maximum of $P(X|W)P(W)$.

Two main components are then considered based on this equation 2.2. The *acoustic model* estimates the probability distribution $P(X|W)$: how likely the feature vectors are generated by a word sequence W. The *language model* estimates probability for each possible text sequence W. In this statistical approach, these two models are handled *separately* with their different sets of free (or learnable parameters). As a result, when the system has to find the optimal sequence $\hat{W}$ given an acoustic observation, there is a search procedure involved to satisfy the equation 2.2.

2.1.1 Evaluation in speech recognition

In order to evaluate speech recognition systems, it is required to compare the hypothesis (which is the output of the ASR system given an audio utterance) and a ground truth, oftentimes the transcript as human reference. In the literature, the most commonly used metric is the word error rate (WER).

Word error rate The WER is computed by looking for the number of insertions, deletions and substitutions that are required to change the hypothesis into the ground truth. The equation is then equivalent to calculating the Levenshtein distance between two sequences:

$$WER = \frac{del + ins + sub}{total} * 100 \tag{2.3}$$

The word accuracy is then $100 - WER$ (in percentage).

Certain languages do not have a clear word boundary, therefore character error rates (CER) or sometimes phone error rates (PER) is evaluated instead, with the same application of Equation 2.3.

Other measurement The error rates can be used as the most important metric for an ASR system, however they are not the direct training objective of neural models, especially end-to-end models. Due to the fact that most models are probabilistic and aim at maximizing the log-likelihood of the data, we often rely on *perplexity* during training.

Let us assume that we have a trained statistical model M and a corpus D containing L words, which the (conditional) language model has not observed during the training process. The quality of model M is evaluated by using it to predict the distribution of the corpus D. The resulted perplexity (PPL) is then obtained by estimating the probability of all words given their context in the corpus D, we use P_M to denote the probability distribution produced by model M.

$$PPL(D) = \exp\left(\frac{\sum_{l=1}^{L} -\ln P_M(w_i|H_i)}{L}\right) \tag{2.4}$$

The perplexity showed how close the model prediction can be in *reading* mode, i.e. when the model only evaluates the likelihood of given input/output pairs. However, in *generation* mode, perplexity might not be a good indicator and does not align with the generation quality. In practice, *unigram accuracy* can be a reliable choice for model selection.

On the other hand, word error rate is only applied for speech recognition. Other speech-to-text applications such as speech translation would have to be evaluated differently. For speech translation, the *BLEU* score [181] is often used by computing the n-gram precisions, up to 4-grams to measure the closeness between the hypothesis and the reference.

Hidden Markov Model A Hidden Markov Model is a system modeling a **Markov** process of a sequence of visible observations $o_0, o_1, \ldots, o_T$ being generated from an internally "hidden" sequence of states $s_0, s_1, \ldots, s_T$. These states belong to a predefined set of states S. While the observations can be either continuous or discrete, the

states are strictly discrete (so as to the model advances across discrete time-steps) with a vocabulary of possible states $s^1, \ldots s^N$.

In a particular time step t, the model staying in state s_t^i would transit into state s_{t+1}^j and emits an observation o_{t+1}. With a vocabulary of N possible states, there are N^2 possible transition paths, modeled by a set of transitional probabilities matrix A with each element a_{ij} denoting the probability of transiting from s^i to s^j (identical at any time step t). Intuitively, if each hidden state corresponds to a word, then the transition set allows the model to know if the model is still "hearing" the same word s^i or is transiting to the next word s^j.

In a similar manner, the observations are also generated from a set of probabilities $b_j(o_t)$ denoting the probability of emitting the discrete observation o_t from state j, which is also identical at any time step t. When the observation o is continuous, then we have to use probability density functions $p(o|s)$ to model the emission probabilities. The popular choice was to use Gaussian Mixture Models that use linear combination of Multivariate Gaussians to acquire the statistics of all feature vectors in the training data that is generated from a state (word/phoneme).

There are two main variants of HMM/GMM models: semi-continuous and fully-continuous models. The former employs a global pool (codebooks) of M $n-$dimensionsal Gaussians $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_M$ and these Gaussians are shared between all models. Each state's emission probability is estimated as a weighted-sum over all Gaussians:

$$p(o|s^i) = \sum_{j=1}^{M} \alpha_{ij} \mathcal{N}_j(x_i, \mu_j, \Sigma_j) \tag{2.5}$$

In contrast, the continuous HMM/GMM acoustic models have a separate set of Gaussians for each state.

$$p(o|s^i) = \sum_{j=1}^{M} \alpha_{ij} \mathcal{N}_i j(x_i, \mu_i j, \Sigma_i j) \tag{2.6}$$

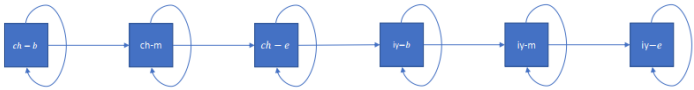


Figure 2.1: A 3-state topology in HMM for ASR.

It is also possible to combine these approaches by having codebooks that are by certain states only (local codebooks instead of a global pool), for example all variants of a phone state in a context-dependent acoustic model.

One certain problem of the Gaussian Mixture models is that the dimension of the feature vector can be very large, as a result of aggregrating the features in multiple frames (a window of feature vectors) leading to large dimensional Gaussians. Dimensionality reduction techniques become necessary to avoid inflating the number of parameters of Gaussians (the same problem in Neural Networks known as the curse of dimensionality, but the Neural models can have finer control over the number of parameters). For example, principal component analysis (PCA) is one of the dimensionality reduction techniques that can be used to reduce the feature size. While PCA can be used without labels, linear discriminant analysis (LDA) requires to be trained with labels and optimized for class separability [64]. When neural networks regained its popularity at the beginning of the 2010s, auto-encoder using multi-layer perceptron is also an option [78].

Training a Hidden Markov Model A HMM is represented by a set of states $s^i, s^j \dots s^N$, the transitional probabilities A and the emission probability density functions $p(x|s)$. The states are designed in a topology, often the words to be recognized need to be broken down into phonemes, and each phoneme is represented by three states *b, m and e* denoting the beginning, middle and end states of each phoneme. The topology is then designed so that the beginning states can only transit to the middle states of the same phone or themselves, while the middle states are limited to either themselves or the end states, and the end states are the only states that can go to the beginning states of

other phones, leading to a sparse transition matrix. This is illustrated in figure 2.1.

The structure of the HMMs suggest that, given one sequence of observation, there are many (N^T) possible state sequences that can generate the observation. Each state sequence is referred to as an *alignment* in which each state is matched with one time-step in the observation sequence. The Markov property indicates that the emission of this state depends only on this state and nothing else, such as the states before or after, or other observations.

Training the HMMs, or optimizing the parameters (in this case, they are the Gaussian parameters of the GMMs for the emission densities) to maximize the likelihood of the observation is basically solving the following dilemma:

- If we know the correct alignment of the observation and the states, we can find the feature vectors corresponding to each state, and then update the Mean μ (by taking the average of the features) and the Covariance Matrices Σ (which are the deviations of the feature vectors with the Mean).
- However we do not know the alignment and the model requires to have good parameters to find the optimal alignment.

In practice, training the HMMs is possible using the Baum-Welch algorithm (or also the EM algorithm), by iteratively update the models and the alignments:

- The current optimal alignment can be found with the current parameters using the Viterbi Algorithm
- Using the current alignment, we can update the emission probabilities
- With the new parameters, better alignment can be found to repetitively iteratively update the model parameters

This training regime is often called Viterbi training, because the Viterbi algorithm [258] is used to find the optimal states given the current model parameters. Note that, training the HMMs is formally done by the forward-backward algorithm that considers *all* paths in the

state space, which is much more complicated than using a single path that has the highest probability to update the model parameters. An additional note is that, training with Gaussian Mixture Models is further complicated due to the fact that the mixtures themselves are a hidden variable (because we do not know which mixture generates the data points). As a result, a combinatory EM algorithm can be used to train the HMM with GMM emission densities.

Context-dependent Acoustic Models The GMM-HMM models, despite the complexity, are not effective in modeling speech signals due to the high variety of speech, especially given the fact that the articulation of each phoneme would differ based on the context - the surrounding phonemes. The limitations of the Markov assumption that allows HMMs to be tractable, unfortunately do not allow longer context to be incorporated.

One of the possible trick to improve contextual learning is to use context-dependent phoneme sets. The set of states is expanded by having one state for each contextual configuration. For example, in a triphone setup, each phone has a unique variation for its left and right context, ending up in a N^3 number of possible states. Afterwards, each state is modeled with a separated Gaussian Mixture Model, allowing the model to learn to discriminate phonemes within context.

Naturally, this scheme leads to several drawbacks. Firstly, data scarcity disallows the model to have sufficient statistics for every phoneme configuration. Secondly, the number of parameters quickly explodes, ending up in an overfitted model using too many parameters to model various configurations that have the same origin (the root phoneme). Consequently, context-dependent training is only viable with several additions:

- Smoothing strategy: the rare triphones' statistic estimation can be done using the less-specific models (such as using the statistics from the biphones or monophones). In addition to this Back-off strategy, it is also possible to interpolate different levels of modeling, such as using the linear combination of triphones, biphones and monophones).

- Parameter sharing: the triphones that have similar structures can be joint as a single states, or parts of the Gaussian Mixture Models can be shared for more efficient modeling. This is often implemented via linguistically based clustering tree, by asking questions about the phonetic properties of the left and right phones. The triphones at the leaf nodes of such trees would be tied as a single state, ending up with a smaller state space and reduced amount of parameters.

Language Model Given a word sequence W with N words, the language model estimates the likelihood of W, i.e how likely W can be observed in the context of all possible sequences (most of the time, limited by the data that covers the sequences). Over the decades, the exponentially increasing amount of data together with effective statistical modeling and machine learning methods have made deterministic grammar-based approaches obsolete. While these methods concerning with formal and explicit models of syntax and semantic can only handle a limited amount of data due to the cost of expert labeling. Statistical language modeling, therefore, were the dominated method until the appearance of neural network language models offering a better estimation approach.

In order to factorize the complicated $P(W)$ Language models have always implied an auto-regressive process assuming that the sequence $W = W_1 W_2 W_3 \dots W_N$ (assuming the sequence has N words) is generated from "left to right" sequentially from the first position until the last. Therefore, the likelihood of each word W_j only depends on the words precedently generated. Such assumption allows for a factorization:

$$P(W) = \prod_{i=0}^{N} P(W_i | W_{i-1}, W_{i-2}, \dots, W_N) \tag{2.7}$$

This equation used to be estimated by $N-$gram models, in which the condition is limited to only a limited value of n, for example unigram ($n = 1$), bi-grams ($n = 2$), tri-gram ($n = 3$), 4-grams ($n = 4$) and so on. These probabilities can be estimated from counting the number

of appearances of these n-grams in the training data. This statistical approach is heavily criticized due to the fact that, most n-grams in the possible combinations do not exist even in very large training data, resulting in zero probabilities for the whole chain. Smoothing techniques [172] can be applied to assign some probabilities to the unseen events, however with very limited success because the model does not have the knowledge regarding the relationship between words.

For this reason, continuous space language models [23, 231, 145] use neural networks to jointly learn word embeddings [165] with a conditional probabilities of $P(W_i|W_{i-1}, W_{i-2}, \ldots, W_N)$. The words in the context are presented with continuous vectors [100], and then feed-forward neural networks learn to map the input to the probabilities of all words in the vocabulary. This approach is much more effective than counting-based n-gram estimation, and resulted in improvement in rescoring the outputs in speech recognition [145].

While feed-forward networks are still limited by the Markov assumption, and the number of words in the context are fixed, *recurrent neural networks* is the effective replacement. Using the recurrent structure, the network can input each word one-by-one and output the probability for the next word, thus allows for a more flexible structure with virtually any context size [166]. Naturally, training this network is difficult due to the gradient vanishing problem when the recurrent structure cannot maintain the information flow. Long Short-Term memory networks [102] and other variants such as Gated-Recurrent Units [36] or Highway networks [292] can effectively alleviate this problem and further improve speech recognition [151]. This recurrent structure also played an important role in establishing the sequence-to-sequence model, which extended neural language model into a conditional neural language model being used in end-to-end speech recognition later.

The invention of Transformers and self-attention enabled neural language models to exploit longer context. The temporal fully-connected nature of self-attention is effective in long-range dependency, and can be applied in language models for even longer contexts [46, 113]. This network architecture remains to be the state-of-the-art in language models [28].

The weaknesses of the HMM based models Hidden Markov Models provide a *tractable* approach to model a sequence of hidden states that generate the observations, and therefore became the backbone of the ASR models for decades. Due the Markov assumptions limiting the transitional and emission probabilities of the model to depend on a very limited context, using HMMs for practical speech applications is a great challenge. Evidently, when searching for the optimal state sequence (triphones or quinphones, for example) for a sentence based application, it is required to incorporate the language models into searching, making the Viterbi algorithm more complicated, as the language model provides a word-based context which is much more helpful than the very local phone-based context in the HMMs.

It is important to note that, the HMM-based models were greatly improved with the presence of Neural Networks. First of all, the emission densities with Gaussian Mixture Models can be replaced with a Deep Neural Network [97]. Instead of using Gaussian Mixture Models, using a multi-layer Neural Network to learn to predict the states from the acoustic features ended up in a much better estimation of the emission. Due to the EM training regime, however, it was still required to use the GMMs to bootstrap the model to have reasonable Viterbi training in the first place. Neural Networks were also use to obtain better features for the GMMs using Bottleneck features acquired from a middle layer of the predictive network mentioned above [69]. And finally, Neural Networks replaced the statistical N-gram models in Language modeling [24, 145, 166] allowing for a stronger modeling with the Fundamental Equation 2.2.

Later models gradually explore Neural-only architecture without the presence of Hidden Markov Models anymore. First we have the Connectionist Temporal Classification models that resemble the hidden states of the HMMs in learning, but rely on Recurrent Neural Networks (RNNs) to model long-range dependencies within the input states. Afterwards, the Encoder-Decoder models gradually became the dominant approach, since they can combine acoustic modeling - language modeling and alignment models in a single Neural model.

2.2 Neural Networks

Given the introductory section of the HMM based models, the next sections will cover the impact of the Neural Networks in changing the Speech Recognition approaches. Therefore, this section is dedicated to introduce the fundamentals of Neural Networks to prepare for the upcoming sections.

Neural networks, or sometimes referred to as Artificial Neural Networks, are a class of machine learning models that *approximate functions* (despite the possible belief that they "learn"" by simulating the human mind). Neural networks receive tensory inputs with float values (occasionally complex values are also possible [101]) and they are organized into *layers* that sequentially transform the values to eventually predict the final output in the final layer.

Early neural networks often consist of only a single layer and the transformation is a linear function firstly proposed in the Perceptron [215] model. Later model stacks multiple linear layers on top of each other, with non-linear activation functions in between, in order to approximate complicated functions. The so called Multilayer Perceptron (MLP) has since then become the backbone of many important modern neural networks. In this section, we focus on explaining the descriptions of MLPs as well as the important neural variations that evolved from MLPs: Recurrent Neural Networks and Transformers.

Feed-Forward Neural Network While Perceptrons are capable of learning to linearly classify, most of the data distributions are not linearly separable, thus making Perceptron inappropriate for complicated tasks as well as learning meaningful representations. Multi-layer Perceptrons (MLP) or Feed-Forward networks combine different layers of Perceptrons with non-linear *activation functions* to allow for more complicated representation learning.

Here we would use the neural language model from [24] to illustrate the operation as well as the back-propagation process in a typical neural network, as illustrated in Figure 2.2 The model takes the input features i as inputs and outputs the conditional probability distribution

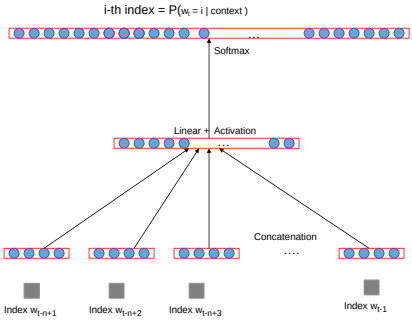


Figure 2.2: Feed-forward neural network for Language Modeling.

over *all* all possible classes which expresses $P(c_j|I)$. A simple three-layer MLP has the following configuration: an input layer, one or many hidden layers and an output layer. In a standard feed-forward neural network, each layer is a real-valued **vector**, while the (learnable) weights (or parameters) are real-valued **tensors** connecting the layers together. The following description considers a network with a single hidden layer for the sake of explanation. Practical uses often involves having many layers leading to better performance [147, 97].

Input layer The input layer is basically the initial feature representation of the inputs. This representation varies based on the modality of the input (such as text, speech or image). For language models (textual input), this layer is often a concatenation of the vectorized representations of each word (word embeddings). While the input layer of speech models is often the concatenation of the frame-level features. These details will be covered in the next section related to Automatic Speech Recognition (ASR).

Hidden layer In the hidden layer, the input i is transformed nonlinearly, where each layer activation values are defined by

$$h = f(W^h i + b^h) \tag{2.8}$$

In equation 2.16, the hidden layer h has the corresponding weights W^h and b^h. The input of the hidden layer is the context vector produced from the input (projection) layer. The size of the hidden layers are tunable hyper parameters. f denotes a nonlinear activation function. Popular choices for the activation function are Tangent Hyperbolic, Sigmoid or ReLU, expressed in equation 2.9. While ReLU is commonly used thanks to the low computation cost, as well as the ability to reduce gradient vanishing (happening when the other two functions are saturated), recent activation functions such as SiLU [58, 212] or GeLU [93] can improve over ReLU with a heftier computational cost [237].

$$f(x) = \begin{cases} \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} \text{ if } f = \text{Tanh} \\ \frac{1}{1+\exp(-x)} \text{ if } f = \text{Sigmoid} \\ \max(0, x) \text{ if } f = \text{ReLU} \\ x * Sigmoid(x) \text{ if } f = \text{SiLU} \\ x * \Phi(x)^1 \text{ if } f = \text{GeLU} \end{cases} \tag{2.9}$$

Output layer The final layer of the network produces the probability distribution for all words in the vocabulary, thus having totally V nodes. Each neuron in the layer is associated to the probability of one word, as shown in Figure 2.2. Firsft, a linear transformation is used to obtain the unnormalised distribution:

$$o = W^o h + b^o \tag{2.10}$$

Each element in o corresponds to the similarity score of the hidden state compared with one class representation (vector) in the weight

matrix W^o. Subsequently, the true probability distribution is estimated thanks to the *softmax* function:

$$p(c_i|h) = \frac{\exp(o_i)}{\sum_j \exp(o_j)} \tag{2.11}$$

In equation 2.11, the probability of each class c_i given the encoded context h is estimated by normalizing all values in o. The number of trainable parameters (tuned during the training process) depends on the number of classes.

Training the network At first, the model is initialized with random parameters and the training process aims at obtaining parameters that allow the models to effectively recognize the classes. This is achieved by using stochastic gradient descent: adjusting the parameters in the opposite direction of the gradients of the objective function (how close the network output compared to the labels). For classification problems, the objective functions often the log-likelihood of the outputs given the inputs and the parameters, computed for every training sample. SGD and variants such as Adadelta [282] or RMSProp [251] require the computation of the first order derivatives of the loss function with respected to the parameters, which can be performed efficiently with the back-propagation algorithm [220, 219].

The backpropagation process In this section, we describe the back-propagation flow in the standard feed forward model - the core of the optimisation process. Back-propagation [219] involves using a *dynamic programming strategy* to compute the derivatives of the loss function with respect to the parameters layer by layer, based on the chain rules. In the standard network, the error derivatives are back-propagated from the output layer to the input (projection) layer.

Objective Function The smoothing function that we approximate with the neural network has parameters that can be iteratively tuned in order to **maximise the log-likelihood of the training data** [24]. The

objective function is therefore chosen as the Negative Log-Likelihood function, since SGD requires the training objective to be minimised. We can compute the loss function over the training data as follows:

$$\mathcal{L} = -\sum_{i}^{N} \log P(c_i|input) \tag{2.12}$$

The loss function is also in line with the Perplexity later used in neural language models and their conditional variants. For ease of understanding, we denote the derivative of the loss function $\mathcal{L}$ at *each* sample or mini-batch of samples with respect to a variable $x \in \Theta$ by dx.

For each sample c_i and its input $input_i$, we have:

$$\begin{aligned} & -\log P(c_i|input_i) \\ &= -log(\frac{\exp(o_c)}{\sum_j exp(o_j)}) \\ &= log(\sum_j \exp(o_j)) - o_w \end{aligned} \tag{2.13}$$

Subsequently, we compute the error derivatives dx given the parameters in each layer using back-propagation:

Output layer The derivatives at the output layer are computed as follows:

$$do_i = \begin{cases} 1 - p_i \text{ if } i == w \\ -p_i \text{ otherwise} \end{cases} \tag{2.14}$$

Here o_i denotes the i^{th} element of the vector o, which is the unnormalised conditional distribution of the classes given the encoded representation of the input, in the hidden layer h.

Hidden layers As a result, we can compute the derivatives with respect to the parameters and the previous hidden layer h, based on the original inference from Equation 2.10.

$$\begin{aligned} dW^o &= doh^T \\ db^o &= do \\ dh &= W^{oT} do \end{aligned} \tag{2.15}$$

Input Layer The inference equation for the hidden layer from the input layer:

$$h = f(W^h i + b^h) \tag{2.16}$$

which implies that:

$$\begin{aligned} d[W^h i + b^h] &= f'(h) * dh \\ db^h &= d[W^h i + b^h] \\ dW^h &= d[W^h i + b^h] i^T \\ di &= W^{hT} d[W^h i + b^h] \end{aligned} \tag{2.17}$$

In order to have the derivatives for the activation function f, we have:

$$f'(x) = \begin{cases} 1 - \text{Tanh}(x)^2 \text{ if } f = \text{Tanh} \\ \text{Sigmoid}(x) - \text{Sigmoid}(x)^2 \text{ if } f = \text{Sigmoid} \\ 1 \text{ when } x > 0 \text{ and } 0 \text{ otherwise if } f = \text{ReLU} \end{cases} \tag{2.18}$$

Parameter Update After obtaining the derivatives of the loss function with respect to all parameters in the network, we can update the parameters according to Stochastic Gradient Descent. The method is based on the phenomenon that the gradient of a function always points towards the direction of maximal increase at any point. The update rule is as follows with the learning rate parameter $\alpha > 0$ and an arbitrary parameter x:

$$x = x - \alpha dx \tag{2.19}$$

The learning rate is also considered as a function of the number of samples trained in the data. From experiments, the learning rate is updated after the model observes a number of training examples with two typical ways. The first way is to exponentially decrease the learning rate after some training samples with a *learning rate decay*, normally an epoch (training all samples in the training data). The second way is to reduce the learning rate based on a validation data. After each epoch, if the perplexity on the validation data is decreased, the learning rate is kept the same, otherwise it is multiplied by the learning rate decay.

Recurrent Neural Networks RNN [59] are a class of neural networks that can efficiently model sequences by using a dynamic memory structure. While the feed-forward network can only receive one input and compute the corresponding output without any relation with other inputs, the recurrent counterpart takes the input as a *series of time step* $x_1, x_2, \ldots, x_n$ and processes them one by one, taking into account the information stored in the previous steps. Concretely, for each input x_i, the network updates the hidden memory h^i based on the previous one h^{i-1}.

The first recurrent language model (RNNLM) [166] employed the "Vanilla" model of Elman et al [59] as can be seen in Figure 2.3, in which the hidden steps are updated as follows:

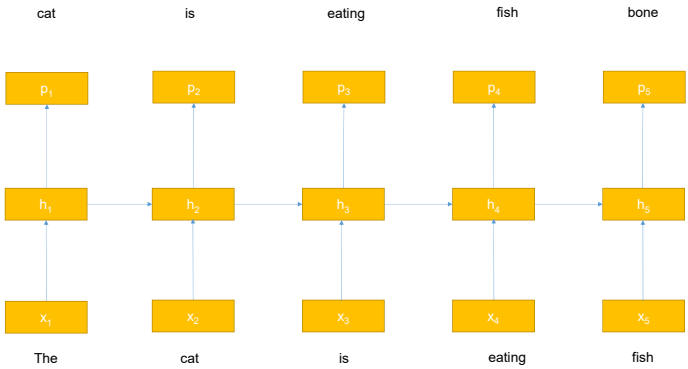


Figure 2.3: A simple RNNLM [166] predicting the sequence "The cat is eating fish bone".

$$h^t = f(W^i i^t + W^h h^{t-1} + b^h) \tag{2.20}$$

The activation function f can be either Tangent Hyperbolic, Sigmoid or ReLU as mentioned before. The starting state h_0 is set to 0 to denote the initial state of the memory. In each time step, the RNNLM can optionally produces the probability distribution for a predicted word, given the sequence that network has scanned previously. The probability distribution over the vocabulary is derived similarly to the feed-forward networks:

$$\begin{aligned} o^t &= W^o h^t + b^o \\ p^t &= \text{softmax}(o^t) \end{aligned} \tag{2.21}$$

with the Softmax function explained in Equation 2.11. To be clear, o^t and p^t denote the unnormalised and normalised distribution generated at time step t. For the language modeling scenario, the input and output samples of the network in each training iteration are two sequences i and y in which the output sequence is the shift-by-1 version of the input sequence. The parameter set of the network including W^i, W^h, b^h, W^o and b^o are shared across time steps.

Training RNNs Similarly to the feed-forward models, The recurrent models can be efficiently trained with stochastic gradient descent (SGD). However, since the networks contain shared parameters at arbitrary numbers of time steps, the gradients are computed differently using back-propagation through time (BPTT). It can be observed that, a change in the parameters in an arbitrary time step t can lead to the change of the objective function in **all** subsequent steps.

The main idea of back-propagation through time is to unfold the network to achieve a MLP-like form, in which back-proagation can be done normally. Due to the parameter sharing over time property, back-propagation through time *accumulates* the gradients over time into the network weights.

This accumulation process involves a long chain of matrix multiplication, that leads to gradient exploding when the values of the gradients become too large, or gradient vanishing when the values of the gradients are not sufficiently large to make updates. This phenomenon happens when dependency between two elements across time cannot be learned, potentially due to the long distance. The Long Short-Term Memory (LSTM) structure is often used as a remedy.

LSTM Structure The intuition of an LSTM starts from the integration of a linear memory unit, so that the gradient can flow smoothly during the back-propagation through time steps using a memory cell c_t.

$$c_t = c_{t-1} + f(Wx_i + Uh_{i-1} + b)h_t = c_t \tag{2.22}$$

This approach is referred as "Leaky integration units" [22]. In the BPTT process, the gradient can flow over exactly one path through the memory units c_i, and since $dc_i = dc_{i-1}$, the gradients are guaranteed to not vanish. The recurrent architecture should also be able to be adequately robust to train long sequences, where there are certain inputs which are irrelevant to the modeling task. Sometimes, the memory of the network should be refreshed, for example at the beginning of a new utterance in Speech Recognition [73] or a new sentence in Machine Translation [247]. hochreiter1997long enhanced the architecture by adding flexible and trainable gates that allows the RNN to reset the memory, control the amount of input and output respectively. The adaptive gates are built from the current input x_t and the previous hidden memory h_t.

The gates of the network include: the forget gate f_t is used to directly control the memory flow c_t to cut the connection with the previous steps, the input gate i_t decides the amount of input to be incorporated, the output gate o_t controls the amount of memory flow to be produced for the task and finally the candidate memory unit $\tilde{C}$ that contributes to the current memory flow. All gates are defined similarly, with the first three gates use the Sigmoid activation to force the values to be in $\{0, 1\}$, while the candidate memory uses the Tanh activation function.

$$\begin{aligned} f_t &= \text{Sigmoid}(W_f x_t + U_f h_t + b_f) \\ i_t &= \text{Sigmoid}(W_i x_t + U_i h_t + b_i) \\ o_t &= \text{Sigmoid}(W_o x_t + U_o h_t + b_o) \\ \tilde{C}_t &= \text{Sigmoid}(W_c x_t + U_c h_t + b_c) \end{aligned} \tag{2.23}$$

In the next step, we decide the new information to be stored in the new memory cell. The cell is updated by combine the input gate and the candidate memory unit. Also, the forget gate is employed to drop certain information from the previous memory cell. Consequently, we come up with a new memory cell as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.24}$$

Finally, we update the hidden state with the new cell state and the output gate:

$$h_t = o_t * \text{Tanh}(C_t) \tag{2.25}$$

The implementation of LSTM can be efficient by computing all gates in one single matrix multiplication, then applying the activation functions on different parts of the output. In practice, one can experience different implementation variations of LSTMs and RNNs in terms of initialisation, bias usage or different gate implementations such as the Gated Recurrent Unit [35]. The empirical research of [281] shows that there is not any substantial difference in terms of performance between different LSTM variations.

Backpropagation with LSTM is more complicated than a simple recurrent neural network. The key idea in LSTM backprop is that the gradient propagation through the cell states c_t is robust, with only one decay factor which is the forget gate. This specific property allows LSTM to avoid gradient vanishing more effectively than simple recurrent neural networks.

2.3 Neural Networks in statistical ASR

Given the details of several neural network architectures, we come back to how they were used in Hidden Markov Model-based ASR. Gaussian Mixture Models, for a long time, were the preferred method in estimating the emission probabilities of the phonetic states (how likely each state can emit observable acoustic features). Despite arduous effort to improve, especially using discriminative training [17, 206, 266] - focus on the wrong decisions of the models - the neural networks were found to be a better regularizer without the overfitting problems of Gaussian Mixture Models. In the literature, using deep neural networks to learn acoustic emission is referred as a Hybrid HMM-DNN system.

In this approach, the input of the neural network is often the Log-Mel frequency features or the Mel Frequency Cepstral Coefficients features with other optional features such as minimum variance distortionless response [272] (MVDR). The networks were traditionally multilayer perceptrons [99] and later replaced with LSTMs.

Initially, a deep configuration multilayer perceptrons containing upto 4 or 5 hidden layers was difficult to train [144] due to the difficulty in backpropagating the gradient signals from the output layer to the lower levels. Such difficulty was overcome using a combination of techniques, ranging from a careful initialization [71], ReLU activation function [45] and more importantly deep belief networks using Boltzmann Machines to pre-train the values of each hidden layer in the neural networks [98].

When the multilayer perceptron models were able to converged in training acoustic models, where they are trained to classify contextual phonemes from a window of features, a large improvement in recognition was observed compared to Gaussian Mixture Models [240].

Acoustic models with neural networks The acoustic models can be materialized with feed-forward neural networks. The input of such networks is the concatenation of the acoustic features in a window of frames. After multiple hidden layers, the network produces the probabilities which correspond to the states in the HMM. The softmax activation function lets the feed-forward network generate the probability $P(s_i|x_i)$ for each state:

$$P(s_i|x_i) = \frac{P(x_i|s_i)P(s_i)}{P(x_i} \tag{2.26}$$

The prior probabilities $P(x_i)$ of the feature vectors are generally set to a constant value such as 1 [214]. The emission probability of the HMM, can then be approximated using the output of the neural networks and the prior probabilities of the states:

$$P(x_i|s_i) \approx \frac{P(s_i|x_i)}{P(s_i)} \tag{2.27}$$

The deep neural networks effectively improved speech recognition systems thanks to the acoustic models [97]. In order to enable effect training with multiple layers, it was necessary to use layer-wise pre-training as a deep belief network to initialize the parameters of the acoustic models.

Despite the improvement, it is notable that the deep neural acoustic models only operate on a *local* level, in which only a few number of frames are concatenated in order to make a local prediction. In contrary, the subsequent end-to-end approach uses neural network on a *global* utterance level, so that the prediction takes into account a larger amount of context.

2.3.1 The end-to-end approach with neural networks

Time-delay neural networks Time-delay neural networks were designed roughly in the late 1980s and were one of the first attempt to use a single neural networks for speech recognition. Despite limited in single unit such as phoneme/word recognition, this network architecture presents a number of properties and shortcomings that are later improved by the recent architectures.

What were the motivations behind the construction of time-delay neural networks? From an early time, feed-forward neural networks had been discovered to be powerful to learn complex non-linear decision surfaces [153, 60]. Such networks are often designed to satisfy a number of requirements.

- First, the network should have multiple layers, each of which is defined by a linear projection coupled with an activation function. The units between each layer must be fully connected.
- Second, the network must have the ability to represent temporal dependencies of the features. Assuming that the features are organized and input into the networks based on a chronological

order, the network architecture should be constructed based on this property, this is also known as the *inductive bias*.

- The hidden representation learned by the networks should be invariant under translation in time. This property is required especially in speech recognition, when the input can be exposed in high variation, for example with recording noise, affecting the lengths of the utterance.
- The learning procedure should not require any precise temporal alignment between of the labels. This include phoneme level alignment - the starting and ending marks for each phonetic unit or word level alignment. This process can give extra information easing the learning process yet the network should be able to learn it based on the multi-layer representation.
- The networks should not be over-parameterized, so that the neurons actually encapsulate the regularities in the training data instead of memorization.

These properties laid out by [259] stay true even decades later, when modern networks are widely applied in the time of the thesis, and in the next sections we can observe that the subsequent approaches improve upon these prerequisites using modern techniques.

TDNN network layer Compared to a standard multi-layer perceptron, a TDNN layer receives input as a vector of vectors rather than a vector of neurons, with the size of $T_{in} \times D_{in}$ in which T_{out} is the length of the input and D_{in} is the size of each feature vector. This input is obtained by taking melscale spectral coeffcients from the speech signal. At the time of TDNN publication, D is a small number 16 while nowadays the values of D can range from 40 to 80 and it can even be tripled using the first and second derivatives from the melscale coefficients [264].

The output of each TDNN layer is also a vector of vector (2D Tensor) with the size $T_{out} \times D_{out}$. The hidden state in each time step t is fully connected with the input states in time steps $t, t+1, \ldots, t+(w-1)$, a window of time step.

$$O_t = sigmoid(W(I_t, I_{t+1}, I_{t+2}, \ldots, I_{t+w-1}) + b) \tag{2.28}$$

The weights in the MLP (W and b) are fully shared between the time steps for the network. The window w suggests that the output is delayed compared to the input by a $w - 1$ amount. The whole network consists of multiple layers being stacked, leading to the final output size being significantly shorter than the input.

The shared weights allow for feature detection being time-transition invariant. Data patterns can be discovered regardless of their positions in the data. Feed-forward neural networks are often overparameterized due to lacking this property [166].

TDNN can be learned efficiently using backpropagation not unlike normal MLP models. For speech recognition specifically, it has been shown empirically that it is possible to outperform Hidden Markov Model based systems in phoneme recognition [259]. Despite the simplicity of the architecture, TDNN has proven that *weight sharing* over time is one of the important properties of neural networks to effectively model time-series properties.

It is also worth noting that Recurrent Neural Networks [59] (RNN) and especially its Long Short-Term Memory variation also carry the weight-sharing paradigm. TDNNs and RNNs are often used interchangeably for sequence modeling for this reason [188, 67, 68], with each network having a different strength and weaknesses. For example RNNs have a longer temporal connection that TDNNs that are limited by the window size. TDNNs on the other is computationally faster and can benefit from a deeper architecture [68, 238].

Connectionist temporal Classification Neural Networks were shown to be capable of recognizing digits [260] and then replacing the Gaussian Mixture Models in the HMM framework to improve the acoustic models, the HMM themselves remain as the base structure. The difficulty of employing a full neural network lies in the asynchronous nature of the input and output signals, having different lengths.

Hidden Markov Models rely on using a low-order Markov Chain to model the alignment between the speech and text signals. The acoustic model has a set of emission probabilities, depicting how each observation is generated in every timestep t, while assuming that this generative problem corresponds to a hidden state (in most systems, its the beginning/end/middle states of phonemes).

Connectionist Temporal Classication, or CTC, has a similar function by employing a single neural network to learn feature represenation, output representation and alignment at the same time. Given an input X with length T_{in} and output Y with length T_{out} with the condition that $T_{out} \leq T_{in}$, CTC is capable of learning an alignment between X and Y:

- Assigns one output token in the output vocabulary to each input step and collapse repeats. For example if "g g g e e t" is the prediction for an input with length 7, then the final output is "get".
- Introduces an extra *blank* token ϵ in the set of the possible outputs, this token allows the model to either output silence, or produce outputs with multiple tokens in a row, such as "hello".

Intuitively, the CTC model learns to "pad" the output tokens with possible repeats and the *blank* token to have the same length with the input tokens, so that a **monotonic** alignment can be established. Obviously, given an output sequence, there are many possible ways to insert repeats or blank tokens to achieve the input length, each of which is considered a **path** A. Learning a CTC model is then equivalent with computing the probabilities of all possible paths A_n, the sum of which is the conditional probability $P(Y|X)$.

$$P(Y|X) = \sum_{A \in \mathbf{A}_{(X,Y}} \prod_{n=1}^{N} p_n(a_n|X) \tag{2.29}$$

While we can use standard networks such as RNN or CNNs to learn the mapping between input features and output tokens for each timestep, it is possibly expensive to compute the CTC loss function due to the large amount of possible paths. Thankfully, dynamic programming

is the solution to quickly estimate $P(Y|X)$ by merging the alignments having reached the same output at the same step.

Since $P(Y|X)$ is the sum of the products of probabilities, it is differentiable and therefore allows backpropagation to estimate the gradients of the weights in the neural networks connected to the output layer[2].

In inference, we need to find the most probable sequence $\hat{Y}$ given an input sequence X. A simple heuristic here is to choose the most likely output at each time-step, so as to obtain the alignment with the highest probability.

$$\hat{A} = \arg\max_{A} \prod_{t=1}^{T} p_t(a_t|X) \tag{2.30}$$

This approach is prone to missing the outputs with much higher probabilities due to the fact that, one output sequence (after merging and removing blanks) can have multiple alignments with the input. Therefore, the most likely alignment does not necessarily correspond to the most likely output sequence. Beam search is often employed to find multiple sequence.

- Beam search operates by computing a new set of hypotheses at each input step. The new set is an extension of the previous set, by extending each hypothesis in the previous set with all possible tokens in the vocabulary, and then keeping the K most likely candidates.
- With CTC outputs, hypotheses can be collapsed into the same output sequence, therefore we need keep track of the prefixes being generated, and sum up the probabilities of the hypotheses having the same output.

CTC has become an important milestone of training ASR models with single neural networks, since this output function allows for learning an alignment between speech inputs and textual labels. Typically,

[2] CTC is also considered as an output layer or loss function in Deep Learning frameworks.

the application of CTC only requires a hidden representation of the input with length $D_{in} \leq D_{out}$, so the networks that generate such representation can be freely chosen, from TDNN [288] to LSTM [239] or Transformers recently [14]. Notably, due to this simplicity, the CTC loss function is the obvious choice when coupled with unsupervised pretraining for speech modality [14].

The weakness of CTC lies in the assumption that output tokens are generated independently. In theory, this property can make the model struggle against words that sound similarly but have different writings. Later CTC models would end up require language models to obtain competitive performance compared to statistical systems [103]. In some cases, the role of language models can be considered to be dominant, when the result can be improved dramatically. Notably, the combination of a CTC and a language model would require a more complicated beam-search to guide two models in looking for the best candidate. Besides, such combination would also nullify the 'end-to-end' approach because the CTC model apparently plays the role of the acoustic model without the need of a pronunciation dictionary.

Attention-based models The CTC models are appealing for a number of reasons. First, they can enable one single neural network to learn latent alignment and perform speech recognition without relying on Hidden Markov Models, therefore being unchained from the Markov Assumption that each state only depends on the previous one. Second, CTC does not require a lexicon or pronunciation dictionary. This advantage helps CTC to be adopted in speech recognition systems for other languages in which a lexicon is difficult to obtain.

For such reasons, we can consider the neural net based CTC models to be a step up compared to the hybrid HMM approach, even though performance wise the latter still has the advantage. When the language model is factored in the CTC model, the performance gap becomes closer but one can argue that the role of the CTC becomes the same with the acoustic model in the hybrid.

On the other hand, neural network language models [23] have bloomed and completely overthrown statistical language models performance

wise [145] and opened up a new world of possibilities with conditional language models (CLM). A typical neural language model uses a sequential structure such as recurrent neural network [166] or TDNN [188] to model long range dependency between words in the sequence, often starting from a special "begin of sentence" or "<bos>" word. A conditional language model would connect such sequential language model with another network learning a different representation. Such idea was manifested first in sequence-to-sequence learning [247] with the goal of learning to translate between sentences of different languages and later in image captioning [121]. The latter employs one convolutional based model to learn image representation and another recurrent model to generate sentence-based captions for the image.

When we have two different neural networks, how can we possibly connect the hidden representations in a meaningful way. For example such connecting network needs to be capable of learning alignment between sentences, or alignment between words and images, or in our application, words and speech frames.

The most important neural advance that enables such applications is Encoder-Decoder with **attention**, in which Encoder-Decoder is a design scheme, while attention is a neural network architecture. In the literature, such development has two phases: first an Encoder-Decoder model with LSTM, which is later completed by the attention network connecting the encoder and the decoder.

Encoder-Decoder Model In this section, we describe the Encoder-Decoder model with attention which is the network design approach for the main speech recognition works in this thesis.

As mentioned in the description of the language models using Long-Short Term memory, sequential representation can be learnt effectively with such neural networks with weight-sharing and time-controlling properties, so that the time dimension is not necessary a fixed number but can vary between samples/applications.

In *sequential transformation* problem, it is desirable to map such sequence of acoustic signals into a sequence of words representing the

transcription (or later, we can directly map it to even the translation) of the original acoustics. A neural network that can map sequences to sequences while being domain independent would be useful for this task.

The main idea is to use a sequential network (TDNN, LSTM or at the moment the state-of-the-art architecture is Transformers) to read the input sequence, one timestep at a time and encapsulate the information of the sentence into one fixed-dimensional vector representation. The second network, presumably an LSTM, can be employed to "uncompress' information from such representation. This is possible thanks to the specific structure of the LSTM having the cell states as the memory of the previous state. Such cell states can be initilized using the final cell states of the LSTM that reads the input sentence, or an hierarchical convolutional architecture [125].

This specific architecture was used solely in Machine Translation due to an architectural flaw in the design which hinders backpropagation to suit long sequences. The performance for this task, therefore, still could not match the statistical approach [56] with the neural counterpart, until the contribution of attention.

Attention model In the neural-decoder design, the *Encoder* needs to be able to contain the information of the sequence in one single state. This common design has been manifested by either using pooling with convolution [124] or using the memory state of a recurrent neural network [247]. Intuitively the network should be dynamically allocated for this task [43] and more importantly, long sequences prevent the back-propagation algorithm due to problems such as gradient vanishing or gradient explosion [128] happening during the backward pass in recurrent networks. Finally, the network does not expose any mechanism to learn alignment between input and output sequences.

Attention formulation With the purpose of avoiding the representation bottleneck between the encoder RNN and the decoder RNN, attention tries to establish the connections from all decoder RNN states

H_i^D to all encoder RNN states H_j^E. As a result, we should have an attention matrix A, in which A_{ij} connects decoder state H_i^D and encoder state H_j^E.

The value of A_{ij} decides the importance of H_i^D versus H_j^E. Intuitively, if $A_{ij} \approx 1$ it can be interpreted as H_i^D is aligned with H_j^E, and more generally the token at position i in the target sequence is aligned with the token at position j in the source sequence. But how does attention generate A_{ij}? The answer is a neural network with a single unit at the output layer.

$$H = tanh(W_{attn1}[H_i^D, H_j^E] + b_{attn1}) \tag{2.31}$$

$$e_{ij} = (W_{attn2} tanh(H) + b_{attn2}) \tag{2.32}$$

$$A_{ij} = \frac{exp(e_i j)}{\sum_n exp(e_i n)} \tag{2.33}$$

Here $[H_i^D, H_j^E]$ is the concatenation of two hidden states in encoder and decoder. e_{ij} is often called the energy function between the two states, and A_ij is obtained from the softmax function over all encoder states.

After obtaining $A_i j$, we can decide the total information of the encoder state involved in decoding time step j as the weighted sum over all encoder states.

$$C_j = \sum A_{ij} H_i^E \tag{2.34}$$

The parameters W_{attn1}, W_{attn2}, b_{attn1}, b_{attn2} are shared between computation of encoder and decoder states, and they can be trained using gradient descents, after receiving gradients from the loss function via back-propagation.

C_J is often called Context vector denoting the contextual information from the encoder. Afterwards, the hidden layer at timestep j before

the hidden layer is decided by another neural layer combining H_i^D and C_j.

Attention advantage From the description above, the amount of information from the encoder changes over time during the generation process of the decoder. These equations allow the decoder to dynamically estimate the relevance of information in the encoder.

The abundance of *connections* between the encoder and decoder removes the gradient bottleneck previously endured in the Encoder-Decoder model. The gradient can flow from the decoder to the furthest encoder states via the attention connections, and gradient vanish is no longer a concern. This is evidently showed by the fact that models with attention are more comfortable translating long sentences than the counterparts [160, 15].

Attention application The Encoder-Decoder with attention model has virtually improved all of the problems presented in the non-attentional model and achieved success in various domains including machine translation [160, 118], image captioning [274] and eventually speech recognition. However, it is also noteworthy that such improvement comes with a price which is the increased space complexity. The attention network connecting encoder and decoder has the space complexity $O(mn)$ in which m and n are the respectively lengths of the input and output sequences.

As a result, the application in speech recognition was unlike other sequence-to-sequence problem, due to the excessive lengths of the speech signals often found in acoustic data. An acoustic signal sampled at 16000 Khz sample rate for one second can yield approximately 2000 timesteps, making the memory cost for attention considerable.

Various works were proposed to deal with this particular problem. The earliest application of the attentional model to speech recognition quickly realized this problem in which long utterances not only make training slower and more expensive, but also reduced the overall performance for shorter ones, indicating that attention was unable to learn alignment [40]. In this very particular work, it was necessary to use

the location of focus (the attention matrix) in the previous decoding step in computing the current attention matrix [74] by using trainable convolutional filters. This very first work was modestly halted at only phoneme recognition due to the modeling power limit.

Further improvement were made to enable large vocabulary continuous speech recognition with this approach [16]. Here, attention is modified so that the process of learning to focus does not involve the whole sequence but instead only specific parts [274], for the sake of efficiency. This is implemented as windowing in the speech input. Furthermore, the recurrent neural network can also reduce the length of the representation to be shorter than the initial sequence by pooling frames neighboring in time, an idea introduced in Clockwork RNN [140].

One more problem of this approach is also realized here. The language model is now internally fused into the main model, as the decoder is fundamentally a conditional language model. However, compared to a normal language model, this concept bears a disadvantage which is the amount of data to be trained is much more limited. The conditional language model requires parallel data containing speech and text formats. In this case, it is able to fuse a language model inside the hidden states of the decoder [81], by using a weighted sum of the hidden states or softmax outputs between the language model and the decoder, given that they have the same nature.

It is also possible to reduce the overall memory usage with LSTMs, by using a specific pyramidal bidirectional LSTM approach [31], dubbed as pBLSTM.

In a normal LSTM layer, the output at the i-th time step, from the j-th layer is computed as:

$$h_i^j = BLSTM(h_{i-1}^j, h_i^{j-1}) \tag{2.35}$$

In this specific bBLSTM layer, each cell computation also takes into account the outputs at the consecutive time-steps in the below layers:

$$h_i^j = pBLSTM(h_{i-1}^j, [h_{2i}^{j-1}, h_{2i+1}^{j-1}]) \tag{2.36}$$

The very bottom layer connecting with the input features remain as a normal bidirectional LSTM. With three more pyramidal LSTM layers stacking on top, the number of time steps at the top layer is reduced by $2^3 = 8$ times, allowing the decoder network with attention to extract the relevant information from the encoded representation more easily. This architecture does not necessarily result in information loss due to the fact that the order of features is still preserved in concatenation.

There are many lessons to be learned from this attempt of adapting an attention-based architecture for speech recognition. Compared to the first attempts [40], it overcame many difficulties in training not only by using an architecture with pooling capability, but also heavy hyper-parameter selection supported by a high amount of computation. Another highlight is to overcome overfitting and training/testing mismatch by using the network's output as input instead of gold labels [21]. This is implemented by randomly sampling from the softmax output distribution at the decoder time step t and feeding it as the input for the next time step $t + 1$.

The authors applied the the "Listen, Attend and Spell" model in a large scale Google voice search with 2000 hours of data and the result can be evaluated from multiple perspectives. On the one hand, there is a clear difference between the HMM baseline (8% WER) and the end-to-end counterpart (14.1% WER). A language model is still necessary to reduce the difference, as can be seen from the previous works [16]. An important discovery here is that, using phonemes as an intermediate representation learning process does not improve the model. This side-experiment indicates a direct mapping function from audio to text does not necessarily require additional phonetic information.

3 Deep Transformers for ASR

In the previous chapter, the role of neural networks in speech recognition literature was studied. Initially they were used as stand-alone models mapping directly from speech features to acoustic units in time-delay neural networks, the same purpose as they are used for the acoustic models in the HMM-based systems. Either they can be used in a Tandem approach, in which neural features are used to enhance GMMs, or Hybrid approach that completely uses deep neural networks for estimation.

Aside from the popularity of neural networks in acoustic models, the connectionist models also gradually dominated *language models*. Similar to the changes in acoustic models, the neural language models fundamentally changed the idea of estimating the probabilities of word sequences, by using learnable parameters for each word [100] and using neural architectures to connect the neurons in the word vectors and learn to output the likelihood of each word in the vocabulary given a context. The main difference comes from two abilities: first, word vectors are a continuous representation and are alleviated from the curse of dimensionality affecting large vocabularies. Second, powerful neural networks such as recurrent neural networks [166] can effectively model the temporal dependencies between words in a sequence.

Due to the high computational cost [145], neural language models were initially employed to solely rescore the outputs of a traditional speech recognition system [145, 167]. Later development showed that neural language models are also practically generative [128] and can be conditioned on a pre-defined representation during generation [124, 37].

The development of both main components in a traditional system suggested that HMMs are not the absolute necessity in modeling the temporal structure, as well as the ability to *abstract* the intermediate phonetic representation that the HMM models relied on. As research moved on, training the neural networks has become more and more feasible thanks to the discovery (or rediscovery) of many different techniques addressing the weaknesses in the architectures. In the big picture, a single neural architecture can promisingly replace the pipeline in the HMM based architecture, by learning a direct mapping function from acoustic signals to the word surface without an indirect discrete representation.

Why is an end-to-end model attractive? As declared in the Introduction chapter of the thesis, the development of the end-to-end model is crucial for the further multilingual advancement. This, however, is far from the only benefit of pursuing this approach:

- The end-to-end offers an easier training process for speech recognition systems. The previous approach includes many models in the pipeline such as an acoustic model, a language model, typically also a tree-based model clustering phonemes into cluster to reduce the dimensions of the output features. Each of these models would require a different training procedure and data organization, such as audio features with phoneme labels for acoustic models, monolingual text data for language models. In contrast, if a single neural architecture is capable of reaching competitive results using only parallel data, the training process would be considerably simplified. Industrial applications of speech recognition would be possible to reduce effort and cost.
- The success of training such an end-to-end would make other speech-to-text applications possible with a similar approach. The specific HMM framework was designed for ASR and might be limited to this application, being difficult to extend to other applications, such as direct translation to other languages' text when transcription is not possible, or directly transforming into the acoustic signals of other languages. The theoretical

advantage of an end-to-end neural network is to avoid decision bottlenecks in discrete/cascaded systems in which consecutive models receive the outputs from the previous counterparts. In the pen-ultimate chapter of this thesis, the model being proposed for ASR is also applied for speech translation achieving competitive result.

- As mentioned before, multilinguality would become a remarkable advantage of the end-to-end approach because of the abstraction of the phonetic information required in the HMM approach. The next chapter would be dedicated to further describe this advantage and how neural networks could be designed to benefit from a multilingual setting.

Modeling Barriers In the literature review, we have describe two attempts to create end-to-end speech recognition models using solely neural networks predating the architectures proposed in the thesis, namely the Connectionist Temporal Classification [75] (CTC) and the Sequence-to-sequence with recurrent neural networks [16, 40] and attention [15].

From the theoretical point of view, the end-to-end model has two advantages compared to the HMM system. First, the neural architecture is not limited by Markov assumption such that the emission of each state depends on only the local previous and next states. For example, a recurrent neural network can deliver information from the beginning of the sequence to the current assessing point (when a certain feature vector is input to the network) via the recurrent connections (and the gating mechanism in Long Short-Term memory neural networks). Second, the output decision is based on a joint network modeling acoustic and language model features. Traditional speech recognition relies on a first realization to the phoneme level, and then the final output sequence is done based on the language model scores combined with the pronunciation dictionary. These systems had to divide representation into different levels in the acoustic model, in particular separating global features (such as channel and speaker characteristics), local features on phoneme level. The language model and acoustic model is then combined at decoding while they are trained with different

loss functions. Meanwhile, the end-to-end approach has the ability to jointly train all components towards a common goal. This ability inherently reduces system complexity, i.e how many components are involved in building a system and have to be trained separately. But more importantly, such an approach can theoretically benefit the most from the powerful neural networks without *error propagation* existing in the decoding process of the traditional systems.

Despite these two advantages, the performance of the end-to-end model still falls behind either the hybrid models or the CTC-based models [279, 270] and more importantly, many of them rely on *external language models* to reach a competitive level of performance which defeats the initial benefit of simplying the training/decoding procedure from the end-to-end setup. From the analysis in the literature above, there was a conundrum about the position of the attention based model replacing the whole multi-model pipeline in statistical approach. On the one hand, the HMM-based models have been industrially enhanced for decades and over the years better and better configurations have been found, from using neural acoustic model to using neural networks to learn features with auto-encoders [97, 207, 69]. More importantly, all of such improvements are from the involvement of neural networks in acoustic and language models, and suggest a futuristic fusion of the whole pipeline into one single neural network.

- Speech signals can vary in volume, noise level, recording qualities, and speaker differences. These variations end up creating unlimited versions of the same utterance that can be mapped to the same transcript. This is a challenge for the neural models having to map directly to word levels without guidance at alignment or phoneme level.
- The data usable to train language models can be more abundant than the parallel speech-text data that the end-to-end models require. As a result, the conditional language models can be more limited compared to unconditional ones. The resulting model often overfits on the limited monolingual data provided from the parallel corpus.

3.1 The Speech Transformer model

From the observation above, it is imperative to enhance the neural sequence-to-sequence models to combat those problems. Starting from the root of sequence-to-sequence models which were designed for text-based translation, the encoder and decoders are required to model two different modalities: the neural encoder reads the audio features into high level representations, which are then fed into an auto-regressive decoder which attentively generates the output sequences [15, 16], while both components are jointly trained towards maximizing the likelihood of the generated output sequence. Unlike text translation, however, the difference in modalities leads the the necessary difference in modeling choice between encoder and decoder, as can be seen from neural image captioning [274, 121]. Due to the high complexity of audio features, the encoder is required to be much deeper, at least more than the standard level of the decoder.

In the neural network literature, training deep neural networks was difficult [144] due to the difficulty in back-propagating the signals from the output layer to the shallower layers. Careful initialization [71] and unsupervised pre-training the values of layers with deep belief networks [98]. Residual connections was probably the most important discovery allowing for deeper networks to be trained.

$$H_t = F(H_{t-1}) + H_{t-1} \tag{3.1}$$

The hidden layer H_t is obtained by taking the sum of the previous layer H_{t-1} and output of the function $F(H_{t-1})$. The gradient in the previous layer $\frac{\delta_L}{\delta_{H_t}}$ can propagate through the additive connection if the function F is complicated. This connection enables even thousands of convolutional layers to train effectively [91]. LSTM employs a similar strategy for *temporal connections* to ensure that the gradient does not vanish after a long time of propagation.

In the mean time, an important invention of modern deep learning is self-attention being able to efficiently represent different structures including text [15] or graph [257] and even acoustic signals [243] with

impressive results. Self-attention refers to the application of attention in one sequence of neural states in contrast with the Encoder-Decoder attention networks that operate on decoder and encoder neural states. For self-attention, the similarities between states are used to establish direct connections between all states in the sequence.

Transformers is a neural design that combines self-attention and residual connection in one package [255]. This encoder-decoder variation divides the network structure into blocks of transformation, each of which contains self-attention and feed-forward neural networks. The purpose of the former is to connect features in different positions in the sequence, while the later adds nonlinearity into the transformation. Combined with residual connection, the Transformers are very powerful in sequence-to-sequence modeling and can deliver state-of-the-art performance in many tasks, especially in natural language processing [50].

From the analysis in the literature, the main contribution of the Thesis is to adopt the Transformer for speech recognition, with two key changes. First, very deep self-attention layers are used for the encoder to handle the high complexity of speech features. Naturally, increasing the number of layers is challenging because of the resulting increase in computational cost and memory cost, due to the fact that back-propagation requires storing the result of each intermediate layer for the backward pass. Not only is computational cost a problem, the network can also overfit on training data due the being overparameterized. In order to facilitate training deep networks by reducing the computational cost and allow the model to generalize better during training, we make the network layers *stochastic*, by allowing layers to be randomly dropped during training. This stochastic layer played an important role in enabling the deeper configuration to scale up to 60 layers in contrast to a typical 6 layer configuration employed in other tasks, as can be empirically demonstrated, by reducing the computational cost and allow the networks to generalize better especially on noisy testing conditions.

Self-Attention As mentioned in the previous chapter, the architectural key to enable end-to-end speech recognition (and other sequence

to sequence problems) is attention. Attention is the mechanism connecting the encoder and decoder representations, and helps the decoder to know which information is locally important. The main quality of attention is the ability to fully connect all states of the first sequence with all states of the second sequence, and each connection is dynamically weighted based on the content of the source/target states at the time.

The formulation of attention can be seen from the context-based information retrieval approaches [74]. The necessary inputs of attention is a query vector $Q \in R^{D \times T_Q}$, a pool of memory containing keys $K \in R^{D \times T_K}$ and values $V \in R^{D \times T_K}$. The similarity between the query Q and each key K_t is measured to weight the importance α_t for the value V_t. In the attention network between encoder and decoder, each hidden state at time i in the recurrent is a query Q_i, while the hidden states in the encoder are the keys and values.

When operating on a sequence, attention can be used to connect the information in different time steps together, replacing recurrent or convolution nets, by treating each hidden state H_i as Q_i, K_i, V_i for attention. In the literature review, we presented attention step-by-step, by computing the relationship between query Q_i and K_i as energy function, presented by a neural network.

$$
\begin{aligned}
e_{ij} &= F_{attn}(Q_i, K_i) \\
a_{ij} &= \frac{\exp^{e_{ij}}}{\sum_k \exp^{e_{kj}}} \\
O &= AV
\end{aligned}
$$

In Transformer, the computation of the energy function can be accelerated by using a simple dot-product operation instead of a neural network (F_{attn}) between the query vector $Q \in R^{T_q \ timesD}$ and key vector $K \in R^{T_k \times D}$. This is known as dot-product attention, and all equations can be manifested as the matrix multiplication between Q,

K, V with the softmax function applied on the energy QK^T, which is the collection of dot-product between vectors Q and K.

$$A = softmax(QK^T)$$
$$O = AV$$

Deep network for Speech Recognition In this section, we provide a detailed description of our deep Transformer model which is inspired by the translation model "Transformer" proposed as in [255]. In general, this model belongs to the class of sequence-to-sequence model with neural networks [247], which takes a source sequence $X = \{x_1, x_2, \ldots, x_m\}$ and transforms into hidden representation H using an encoder network. Another decoder network would consume H and subsequently generate an output sequence $Y = \{y_1, y_2, \ldots, y_n\}$.

The main components of the model include an encoder, which consumes the source sequence and then generates a high level representation, and a decoder generating the target sequence. The decoder models the data as a conditional language model - the probability of the sequence of discrete tokens is decomposed into an ordered product of distributions conditioning on both the previously generated tokens and the encoder representation.

Both encoder and decoders are neural networks and require neural components that are able to learn the relationship between the time steps in input and output sequence. The decoder also requires a mechanism to analyze the encoder representation. For the Transformer, attention or its common variation Multi-head attention is the core of the model.

The overall architecture is demonstrated in figure 4.2. Here we tailored the Transformer model [255] without changing the main motivation of using self-attention for sequence learning.

The encoder and decoder of the Transformers are constructed by layers, each of which contains self-attentional sub-layers coupled with feed-forward neural networks.

Adapting the Transformer models for speech recognition

To adapt the encoder to long speech utterances, we follow the reshaping practice from [243] by grouping consecutive frames into one step. Subsequently we combine the input features with Sinusoidal positional encoding [255]. While adding directly acoustic features to the encoding is harmful which potentially leads to divergence during training [243], we resolved that problem by simply projecting the concatenated features to a higher dimension (512 as other hidden layers in the model) before adding. In the case of speech recognition, the positional encoding offers a clear advantage compared to learnable positional embeddings [68] because the speech signals can be arbitrarily long with a higher variance compared to text sequences.

The Transformer encoder passes the input features to a self-attention layer followed by a feed-forward neural network with 1 hidden layer with the ReLU activation function. Before these sub-modules, we follow the original work to include residual connections which establishes short-cuts between the lower-level representation and the higher layers. The presence of the residual layer massively increases the magnitude of the neuron values which is then alleviated by the layer-normalization [12] layers placed after each residual connection.

The decoder is the standard Transformer decoder in the recent translation systems [255]. The notable difference between the decoder and the encoder is that, the self-attention layer of the decoder has to be masked so that each state has only access to the past states, to maintain the auto-regressive nature of the model. Moreover, an additional attention layer using the target hidden layer layers as queries, and the encoder outputs as keys and values is placed between the self-attention and the feed-forward layers. Residual and layer-normalization are setup identically compared to the encoder.

This particular design of the Transformer has various advantages compared to previously proposed RNNs and CNNs networks. First, com-

putation of each layer and sub-module can be efficiently parallelized over both mini-batch and time dimensions of the input. Second, the combination of residual and layer normalization is the key to enable deep configurations to be trainable, which is the main reason for the performance breakthrough in recent works in both MT and natural language processing [50, 189].

Stochastic layers While deep transformer layers are powerful models for S2S task, the number of parameters dramatically increases over the model depth, leading to over-fitting and expensiveness in terms of time and space during training. In order to attack both problems at once, we use the stochastic network method which was applied to convolutional ResNet [107]. Specifically, the observation of the residual connections [90] is that the network during training consists of multiple sub-networks taking different paths through the shortcut connections [256]. The deep transformer is established by a set of residual blocks, most of which consist of attention or feed-forward networks. These submodules, as a result of residual connections, are connected no matter how deep the network becomes. Motivated by dropout [244] dropping connections between neurons and the stochastic ResNet [107], we propose to apply the stochastic property on top of the residual connections.

The original residual connection of an input x and its corresponding neural layer F has the following form:

$$R(x) = \text{LayerNorm}(F(x) + x) \tag{3.2}$$

In equation 3.3, the inner function F is either self-attention, feed-forward layers or even attention from the decoder to the encoder. The Layer Norm as in [12] keeps the magnitude of the hidden layers from growing large,. Stochastic residual connection is fundamentally applying a mask M on the addition of the input, as follows:

$$R(x) = \text{LayerNorm}(M * F(x) + x) \tag{3.3}$$

Mask M only takes 0 or 1 as values, generated from a Bernoulli distribution similar to Dropout [244]. When $M = 1$, the inner function

F is activated while it is skipped when $M = 0$. This stochastic connection enables more sub-network configurations to be established during training, and during inference the full network is presented, causing the effect of ensembling different sub-networks, as analyzed in [256]. Computationally, we only generate one mask per minibatch using a hyper-parameter p for each residual connection, thus the layer is skipped for the whole minibatch depending on the mask value. The computation cost is then decreased because the expectation of the network length is shorter than the typical Transformer. More important is how the the layer-wise dropping parameter p is chosen, since the amount of residual connections for the Transformer is not insignificant. As suggested by [107], the lower layers of the networks handle raw-level acoustic features on the encoder side, and the character embeddings on the decoder side. Our early experiment with a constant p for all connections found out that, lower level representations turn out to be less tolerable than the higher ones in terms of dropping effect. In other words, the lower the layer is, the lower its probability is required to be set. As a result, p values are set with the following policy:

- Sub-Layers inside a Transformer layers (Encoder or Decoder) share the same p value and are dropped or kept at the same time. In other words, we assign values for each Transformer layer, not each sub-layer inside.
- Lower layers are dropped more seldomly. This is achieved by using the Linear strategy suggested in [107]. One hyper parameter p is set before training, which equals the dropping probability of the top most layer. The lower layers have lower probability linearly scaled by its depth. The intuition here is that the lower levels are more important due to being close to the initial feature layers, the top layers should be able to be more optional, since the high level representations are more organized.

Lastly, since the layers participate with probability $1-p$ during training and are always presented during inference, we scale the layers' output by $\frac{1}{1-p}$ whenever they are not skipped. Therefore, each stochastic residual connection has the *training* form (the scaler is removed during testing):

$$R(x) = \text{LayerNorm}(M * F(x) * \frac{1}{1-p} + x) \tag{3.4}$$

Figure 3.1: The downsampling process before inputting the features into the Transformers.

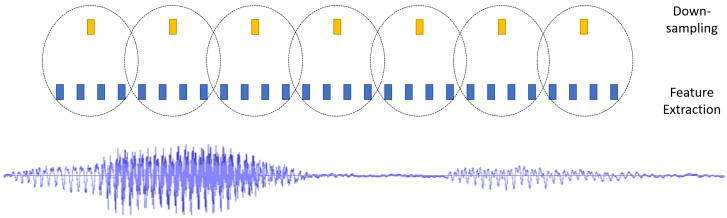
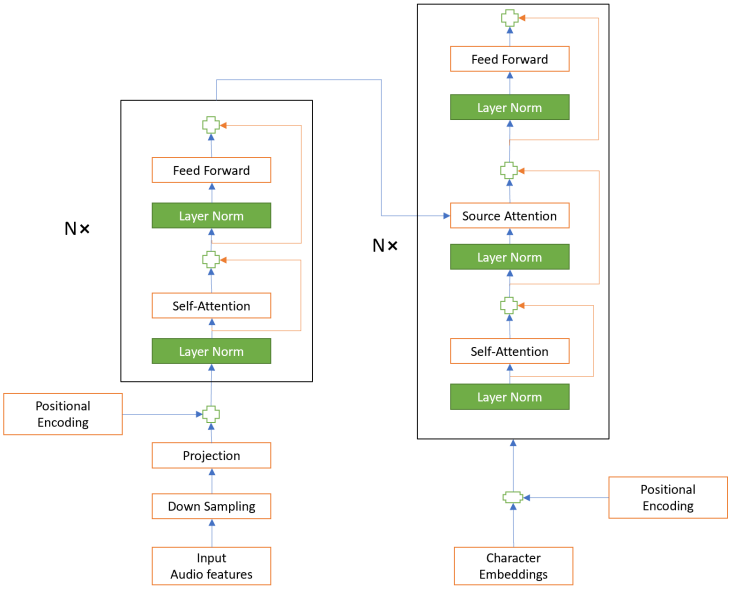


Figure 3.2: A diagram of transformation from acoustic features to character-level transcriptions. The red connections represent the residual connections, which are rescaled according to Equation 3.4 for stochastic transformers.



3.2 Stochastic Transformer experiments

Data Our experiments were conducted on the Switchboard-1 Release 2 (LDC97S62) training corpus which contains over 300 hours of speech. The Hub5'00 evaluation data (LDC2002S09) was used as test set. All the models were trained on 40 log mel filter-bank features which are extracted and normalized per conversation. We also adopted a simple down-sampling method in which we stacked 4 consecutive features vectors to reduce the length of input sequences by a factor of 4. Beside the filter-bank features, we did not employ any auxiliary features. We followed the approach [137] to generate a speech perturbation training set. Extra experiments are also conducted on TED-LIUM 3 [94] dataset which is more challenging due to longer sequences.

Implementation details Our hyper-parameter search revolves around the *Base* configuration of the machine translation model in [255]. For all of our experiments in this work, the embedding dimension d is set to 512 and the size of the hidden state in the feed-forward sub-layer is 1024 [1]. The mini-batch size is set so that we can fit our model in the GPU, but we accumulate the gradients and update every 25000 characters. Adam [133] with adaptive learning rate over the training progress:

$$lr = init\_lr * d^{-0.5} * min(step^{-0.5}, step * warmup^{-1.5}) \qquad (3.5)$$

in which the init_lr is set to 2, and we warm up the learning rate for 8000 steps. After reaching the peak, the learning rate linearly decreases over time to avoid overfitting, with the intuition that the learning process should be slow at first when the gradient is unstable. Dropout [244] (applied before residual connection and on the attention weights) is set at 0.2. We also apply character dropout [65] with $p = 0.1$ which randomly replaces character embeddings with a zero vector and label smoothing [248] with $\epsilon = 0.1$.

[1] Because of GPU memory consumption reduction.

Table 3.1: The performance of deep self-attention networks with and without stochastic layers on Hub5'00 test set with 300h SWB training set.

Layers	#Param	SWB	CH
04Enc-04Dec	21M	20.8	33.2
08Enc-08Dec	42M	14.8	25.5
12Enc-12Dec	63M	13.0	23.9
+*Stochastic Layers*		13.1	23.6
24Enc-24Dec	126M	12.1	23.0
+*Stochastic Layers*		11.7	21.5
+*Speed Perturbation*		10.6	20.4
48Enc-48Dec	252M	-	-
+*Stochastic Layers*		11.6	20.9
48Enc-48Dec(half-size)	63M	-	-
+*Stochastic Layers*		12.5	22.9
08Enc-08Dec(big)	168M	13.8	25.1
24Enc-12Dec	113M	13.3	23.7
+*Stochastic Layers*		11.9	21.6
36Enc-8Dec	113M	12.4	22.6
+*Stochastic Layers*		11.5	20.6
36Enc-12Dec	113M	12.4	22.6
+*Speed Perturbation*		11.2	20.6
+*Stochastic Layers*		11.3	20.7
+*Speed Perturbation*		**10.4**	**18.6**
40Enc-8Dec	109M	-	-
+*Stochastic Layers*		11.9	21.4

Results We conducted the first experiments on the standard Switch-Board (SWB) dataset containing approximately 300 hours of audio, as illustrated in table 3.1. A shallow configuration (i.e 4 layers) is not sufficient for the task, and the WER reduces from 20.8% to 12.1% on the SWB test when we increase the depth from 4 to 24. However, increasing the depth of the models to 12 However increasing the number of layers to 12 and 24 reduces only 5% relative WER, which seems to be a symptom of overfitting.

Table 3.2: Comparing our best model to other hybrid and end-to-end systems reporting on Hub5'00 test set with 300h SWB training set.

Hybrid/End-to-End Models	**Unit**	**SWB**	**CH**
TDNN +LFMMI [207]	Phone	10.0	20.1
BLSTM +LFMMI [207]	Phone	**9.6**	19.3
CTC+CharLM	Char	21.4	40.2
LSTM w/ attention [15]	Char	15.8	36.0
Iterated-CTC +LSTM-LM [293]	Char	14.0	25.3
Seq2Seq +LSTM-LM [285]	BPE	11.8	25.7
Seq2Seq +Speed Perturbation [270]	Char	12.2	23.3
CTC-A2W +Speed Perturbation [279]	Word	11.4	20.8
36Enc-12Dec (ours) (3)	Char	10.4	18.6
48Enc-12Dec (ours) (4)	Char	10.7	19.4
60Enc-12Dec (ours) (5)	Char	10.6	19.0
Ensemble		9.9	**17.7**

Our suspicion for this problem is confirmed by the stochastic networks. At 12 layers, the stochastic connections only improve the CH performance by a small margin, however the improvement was significantly better on the 24 layer setting. Following the trend, the stochastic 48-layer model keeps improving on the CH test set, showing the model's ability to generalize better. Our best result is obtained at 11.6% on SWB, and 20.9% on CH.

Arguably, the advantage of deeper models is to offer more parameters, as can be seen from the second column. We performed a contrastive experiment using a shallow model of 8 layers, but the model size is doubled so that its parameter count is comparable with the deep 24-layer model. By showing that the performance of this model is significantly worse than the 24 layer one, we showed that a deeper network with smaller size is more beneficial than simply increasing the network size.

Our second discovery is that the encoder requires deeper networks. As analyzed above, the encoder has to undertake learning the representations starting from audio features, while the decoder handles generating the character sequence conditionally based on the encoder representation. The difference in modality in turn suggests us to setup the configuration accordingly. By holding the total number of layers as 48, we add more biases to the encoder layers. The result shows us that, a much shallower decoder with only 8 layers but with 40 encoder layers is as good as the 24-layer configuration. More stunningly, we were able to obtain the best result with 20.7% WER which is competitive with the best previous work using data augmentation.

Finally, it was revealed that the combination of our regularization techniques (dropout, label-smoothing and stochastic networks) compensate with data augmentation, which furthermore improved our result to 18.1% with the 36-12 setup. This model, as far as we know, establishes the state-of-the-art result for the SwitchBoard benchmark among end-to-end speech recognition models, as shown in table 3.5. Comparing to the best hybrid models with similar data constraint, our models out-performed on the CH test set while remaining competitive on the SWB testset without any additional training data for the language models. This result suggests the strong ability to generalize of the Stochastic Transformer for the speech modality.

However, we would like to emphasize that our model is significantly deeper than the previous works. The experiments with similar depth suggests that self-attention performs competitively compared to LSTM [102] or TDNN [259]. However, the former benefits strongly from building a deep residual network, in which our main finding shows that depth is crucial for a uing self-attention for speech recognition.

On TED-LIUM dataset Table 3.3 shows our result on TED-LIUM (version 3) dataset. With a similar configuration to the Switchboard models, we managed to outperform a strong baseline which uses both an external language model trained on larger data than the available transcription and speed perturbation with the model with 36 encoder layers and 12 decoder layers. We also show the trend that the models are benefit from a deeper encoder, and together with the stochastic

Table 3.3: The Transformer results on the TED-LIUM test set using TED-LIUM 3 training set.

Models	test WER
CTC [94]	17.4
CTC/LM + speed perturbation [94]	13.7
12Enc-12Dec (ours)	14.2
Stc. 12Enc-12Dec (ours)	12.4
Stc. 24Enc-24Dec (ours)	11.3
Stc. 36Enc-12Dec (ours)	**11.1**

residual connections we managed to improve WER by 21.8% relatively, from 14.2 to 11.1%. Given the potential of the models, it is strongly suggested that better result can be obtained by further hyper-parameter optimization.

3.3 Relative Positional Encodings

From the previous work, it is now evident that neural sequence-to-sequence models [247] are capable of directly transcribing or translate speech in an end-to-end approach. A single neural model which directly maps speech inputs to text outputs advantageously eliminates the individual components in non end-to-end or cascaded approaches, while yielding competitive performance [242, 175]. The hybrid approach for speech recognition and the cascaded approach for speech translation may still give the best accuracy in many conditions, but as neural architectures continue to develop, the gap is closing [179].

The Transformer [255] is a popular architecture choice which has achieved state-of-the-art performance for many sequence learning tasks, particularly machine translation [173]. When applied to speech recognition and direct speech translation, this architecture also stands out as the highest performing option for several datasets [243, 199, 52].

The disadvantage of the Transformer is that, its core function – *self-attention* – does not have an inherent mechanism to model *sequential positions*. The original work [255] added position information to the word embeddings via a trigonometric position encoding. Specifically, each element in the sequence is assigned an *absolute* position with a corresponding encoding (a vector similar to embeddings of the discrete variables, but not updated during training). Recent adaptation to speech recognition [199][2] showed that the base model, extended in depth, is already sufficient for competitive performance compared to other architecture approaches.

However, this absolute position scheme is far from ideal for acoustic modeling. First, text sequences may have a stricter correlation with position; for example, in English the "Five Ws" words often appear at the beginning of the sentences, while there is possibly a larger variation in the absolute position of phones in speech signals and utterances. Second, speech sequences are often $10 - 60$ times longer than their transcript character sequence, which can be exacerbated by surrounding noises or silences. For example, if the speech is surrounded by applauses, the positions can change but should not affect the resulting transcript. Ideally, we want to keep positional information *time-shift invariant*.

Recently, relative positional encoding has become popularized as a consistent reinforcement for the self-attention. Originally proposed by [235] to replace absolute positions by taking into account the *relative positions* between the states in self-attention, this method has also been formalized to adapt into language modeling [46], which allows the models to capture very long dependency between paragraphs.

The next achivement of the Thesis is to bring the advantages of relative position encoding to the Deep Transformer [199] for speech-to-text architectures that can be used in both transcription and translation. The resulting novel model maintains the trigonometric position encodings to better scale with longer speech sequences, and is able to model bidirectional positions as well. On speech recognition, we show that

[2] This is the closest speech adaptation that does not change or introduce additional layers (e.g. LSTM [102] or TDNN [259]).

this model consistently improves the Transformer on the standard English Switchboard and Fisher benchmarks (on both 300h and 2000h conditions), and, to the best of our knowledge, is the best published end-to-end model without augmentation on these datasets.

Factorizing attention A speech-to-text model for either automatic speech recognition or direct translation transforms a source speech input with N frames $X = x_1, x_2, \ldots, x_N$ into a target text sequence with M tokens $Y = y_1, y_2, \ldots, y_M$. The encoder transforms the speech inputs into hidden representations $h^X_{1\ldots N}$. The decoder firsts generates a language model style hidden representation h^Y_i given the previous inputs, then uses the attention mechanism [15] to generate the relevant context c_i from the encoder states, which is then combined and generate the output distribution o_i.

$$h^X_{1\ldots N} = ENCODER(x_1 \ldots x_N) \tag{3.6}$$
$$h^Y_i = DECODER(y_i, y_{1\ldots i-1}) \tag{3.7}$$
$$c_i = \text{ATTENTION}(h^Y_i, h_{1\ldots N}) \tag{3.8}$$
$$o_i = \text{SOFTMAX}(c_i + h^Y_i) \tag{3.9}$$
$$y_{i+1} = sample(o_i) \tag{3.10}$$

The Transformer [255] uses attention as the main network component to learn encoder and decoder hidden representations. Given three sequences of vectorized states consisting of queries $Q \in R^{|Q|\times D}$, $K, V \in R^{|K|\times D}$, attention computes an energy function e_{ij} between each query Q_i and each key K_j. These energy terms are then normalized with a softmax function, and then used to take the weighted average of the values V. The energy function can be modeled with neural networks [160] or as simple as projected (with thre additional weight matrices) dot-product between two vectors $e_{ij} = Q_i K^T_j$ or as parallelized matrix-multiplication in Equation 3.11.

$$\begin{aligned} \hat{Q} = QW_Q; \hat{K} = KW_K; \hat{V} = VW_V \\ \text{Attention}(Q, K, V) = \text{softmax}(\hat{Q}\hat{K}^T)V \end{aligned} \tag{3.11}$$

[255] also improved the attention above through the concept of multi-head attention (MAH), which splits the transformed term $\hat{Q}, \hat{K}, \hat{V}$ to H different heads. The same dot-product operation is applied on each of the H query, key and values heads, and finally the result is the concatenation of the H outcomes.

The Transformer encoder and decoder are constructed based through stacked layers that have identical components. Each encoder layer has one self-attention (MAH) sub-layer, which is followed by a position-wise feed-forward neural network with ReLU activation function.[3] Each decoder layer is quite similar to the encoder counterpart, with the self-attention sub-layer to connect the decoder states, and the feed-forward network. There is an additional Encoder-Eecoder attention layer in between to extract the context vectors from the top encoder states. Furthermore, the Transformer uses residual connections boost information from bottom layers (e.g. the input embeddings) to the top layers. Layer normalization [12] plays a supportive role, keeping the norms of the outputs in check, when used after each residual connection.

Relative Position Encoding in Transformer Equation 3.11 suggests that attention is position-invariant, i.e if the key and value states change their order, the output remains the same. In order to alleviate this problem for this content-based model, positional information within the input sequence is represented in a similar manner with the word embeddings. The positions are treated as discrete variables and then transformed to embeddings either using a look-up table with learnable parameters [246] or with fixed encodings in a trigonometric form:

$$\begin{aligned} Pi, 2k &= sin(\frac{i}{10000^{2k/D}}) \\ Pi, 2k+1 &= cos(\frac{i}{10000^{2k/D}}) \end{aligned} \tag{3.12}$$

[3] It is a sub-layer from the top-down perspective, analyzing the network, but as a neural network itself, it has two hidden layers of its own

When applied to *speech* input, this encoding is then added to speech input features [199]. The periodic property of the encodings allow the model to generalize to unseen input length. Following the factorization in [46], we consider the cosine similarity between queries and keys as "Energy", we can rewrite the energy function in Equation 3.11 for self-attention between two encoder hidden states H_i and H_j that represent the states at timesteps i and j to decompose into 4 different terms:

$$\begin{aligned} \text{Energy}_{ij} &= Energy(H_i + P_i, H_j + P_j) \\ &= H_i W_Q W_K^T H_j^T + H_i W_Q W_K^T P_j^T \\ &\quad + P_i W_Q W_K^T H_j^T + P_i W_Q W_K^T P_j^T \\ &= A + B + C + D \end{aligned} \tag{3.13}$$

Equation 4.21 gives us an interpretation of the function: in which term A is purely content-based comparison between two hidden states (i.e speech feature comparison), term D gives a bias between two absolute positions. The other terms represent the specific content and position addressing.

The extension proposed by previously [235] and later [46] changed the terms B, C, D so that only the relative positions are taken into account:

$$\begin{aligned} \text{Energy}_{ij} &= Energy(H_i, H_j + P_{i-j}) \\ &= H_i W_Q W_K^T H_j^T + H_i W_Q W_R^T P_{i-j}^T \\ &\quad + u W_K^T H_j^T + v W_R^T P_{i-j}^T \\ &= A + \tilde{B} + \tilde{C} + \tilde{D} \end{aligned} \tag{3.14}$$

The new term $\tilde{B}$ computes the relevance between the input query and the relative distance between Q and K. Term $\tilde{C}$ introduces an additional bias v to the content of the key state H_j, while term $\tilde{D}$ represents the bias to the global distance. Terms $\tilde{B}$ and $\tilde{D}$ also have an additional linear projection W_R so that the positions and embeddings have different projections.

With this relative position scheme, when the two inputs H_i and H_j are shifted (for example, having extra noise or silent in the utterance), the energy function stays the same (for the first layer of the network). Moreover, it can also establish certain inductive bias in the data; for example, the average length of silence or applauses, given the global and local bias terms.

Adaptation to speech inputs For relative position encodings with speech inputs, should we use learnable embeddings or fixed encodings to represent the distance P_i? The latter has the clear advantage that it already has the periodic property, and given that speech input can be as long as thousands of frames, the former approach would require a necessary cut-off [235] to adapt to longer input sequences. These reasons make sinusoidal encodings a logical choice. Importantly, the relative position scheme above was proposed for autoregressive language models, in which the attention has only one direction. For speech encoders, each state can attend to both left and right directions, thus we propose to use positive distance when the keys are to the left ($j < i$) and negative distance otherwise. As a result, the encodings for P_k and P_{-k} will have the same sin terms while the cos terms will have opposite signs, which gives the model a hint to assign different biases to different directions. Implementation wise, it is able to efficiently compute terms $\tilde{B}$ and $\tilde{D}$ with the minimal amount of matrix operations. It is necessary to compute $2K-1$ terms $H_i W_Q W_R^T P_k^T$ with $-K < k < K$ for each query H_i (For a sequence with K states, the distance between one state to another k is always in that range).[4] This is followed by the shifting trick [46] to achieve the required energy terms.

[4] [46] only needs to compute K terms as it has only one direction

3.4 Experiments with Relative Position Encodings

Our baselines for all experiments use the Deep Stochastic Transformer [199]. We use the relative encoding scheme above for both encoder and decoder to yield relative Transformers.

For ASR, both our baseline Transformer and relative Transformer have 36 encoder and 12 decoder layers with the model size $D = 512$ and the feed-forward networks have the hidden layer size of 2048. Dropout is applied with the same mask across time steps [65] with $P_{drop} = 0.35$ and also directly at the discrete decoder inputs with $P_{drop} = 0.1$. All models are trained for at most 120000 steps and the reported model parameters are the average of the 10 checkpoints with lowest perplexities on the cross-validation data.

For all models, the batch size is set to fit the models to a single GPU [5] and accumulate gradients to update every 12000 target tokens. We used the same learning rate schedule as the Transformer translation model [255] with 4096 warmup steps for the Adam [133] optimizer.

We present ASR results on the Switchboard-300 benchmark in Table 3.4. It is important to clarify that spectral augmentation (dubbed as *SpecAugment*) is a recently proposed augmentation method that tremendously improved the regularization ability of seq2seq models for speech recognition [183]. In better demonstrate the effect of relative attention, we conduct experiments with and without augmentation.

Compared to the Deep Stochastic model [199], using relative attention is able to reduce our WER from 10.9 to 10.2 and 19.9 to 19.1 on SWB and CH, without any augmentation. Compared to other works under this condition, our results are second to none among the published end2end models, and can rival the LFMMI hybrid model [207] that has an external language model utilizing extra monolingual data.

[5] Titan V and Titan RTX with 12 and 24 GB respectively

Table 3.4: ASR: Comparing our best models to other hybrid and end-to-end systems on the **300h** SWB training set and Hub5'00 test sets. Absolute best is bolded, our best is italicized. WER↓ .

	Models	SWB	w/ SA	CH	w/ SA
Hyb.	[207] BLSTM+LFMMI	9.6	–	19.3	–
	[285] Hybrid+LSTMLM	8.3	–	17.3	–
End-to-End	[183] LAS (LSTM-based)	11.2	**7.3**	21.6	**14.4**
	[284] Shallow Transformer	16.0	11.2	30.5	22.7
	[284] LSTM-based	11.9	9.9	23.7	21.5
	[175] LSTM-based	12.1	9.5	22.7	18.6
	+SpecAugment +Stretching	–	*8.8*	–	17.2
Ours	Deep Transformer *(Ours)*	10.9	9.4	19.9	18.0
	+SpeedPerturb	–	*9.1*	–	*17.1*
	Deep Relative Transformer *(Ours)*	**10.2**	8.9	19.1	17.3
	+SpeedPerturb	–	*8.8*	–	**16.4**

With spectral augmentation, the improvement from relative attention is still noticeable, further reducing WER from 9.4 to 8.9, and 18.0 to 17.3 from the baseline (the relative gain on **CallHome** is kept at 4%). This is second only to [183], the state-of-the-art on this benchmark at 7.3 and 14.4; however their models use an aggressively regularized training regime on multiple TPUs for 20 days. Other end-to-end models [284, 175] using single GPUs showed similar behavior to ours with SpecAugment. Finally, with additional speed augmentation, relative attention is still additive, with further gains of 0.3 and 0.7 compared to our strong baseline.

The experiments on the larger dataset with 2000h follow the above results for 300h, continuing to show positive effects from that relative position encodings. The error rates on those SWB and CH decrease from 6.5 and 11.9 to 6.2 and 11.4 (Table 3.5). Our best model is significantly better than previously published CTC [11] and LSTM-based [175] models, and approaches the heavily tuned hybrid system [89] with dense TDNN-LSTM. It is likely possible to reach better error rates,

Table 3.5: ASR: Comparison on **2000h** SWB+Fisher training set and Hub5'00 test sets. Absolute best is bolded, our best is italicized. WER↓ .

	Models	SWB	CH
Hybrid	[207] Hybrid	8.5	15.3
	[225] Hybrid w/ BiLSTM	7.7	13.9
	[89] Dense TDNN-LSTM	**6.1**	**11.0**
End-to-End	[11] CTC	8.8	13.9
	[175] LSTM-based	7.2	13.9
	Deep Transformer *(Ours)*	6.5	11.9
	Deep Relative Transformer *(Ours)*	*6.2*	*11.4*

with the help of ensemble models, further data augmentation, and language models. Our experiments here, however, show that the novel relative model is consistently better than the baseline, regardless of the data size and augmentation conditions.

3.5 Conclusion

In this chapter, we described the deep stochastic Transformer models applied for large vocabulary speech recognition. The core structure contains self-attention based neural networks stacked for many layers to enhance the representation power. In order to facilitate training these deep networks up to even 60 layers, it is crucial to use stochastic layers by allowing layers to randomly involve in the network during training, but all layers are presented during evaluation, and allowed us to tame the deep Transformer networks which are slow and difficult to train [189]. Using this approach, we were able to compete with previous works using CTC-based and hybrid HMM models. In the noisy CallHome testset, our models are able to improve over a competitive baseline with LSTM-based hybrid models.

Noticing the design of Transformer is not ideal for speech processing due to using absolute position encodings, while we should take into account the relative distance between speech features, we proposed to use *Relative Positional Encodings* for *each attention layer* to better measure the similarity between states in the self-attention mechanism. Using this modeling scheme together with a high amount of data from Fisher helped improve the performance on the conversational benchmark even further. Later, we can show the impact of this modeling choice in speech translation, in which alignment quality can badly affect performance.

Followed impacts Following our works, Transformers have become the standard architecture for end-to-end speech recognition, beside another popular choice of TDNN-LSTM-based architecture in which the encoder is a mixture of TDNN (for downsampling the speech signals) and multi layers of LSTM for representation learning [183, 114] and the two models can trade blow in experiments [175]. The combination of the two models can reach a super human performance in speech recognition under a low-latency constraint [174]. Other works also improve upon the Transformers with other architectural enhancement. Conformers [80] include an additional point-wise convolution between the self-attention layers and the feed-forward neural networks layers, or Branchformers [186] that design branches of convolution and self-attention in the networks. Training an end-to-end model can also be combined with CTC by using both of the loss functions, the CTC loss function at the end of the encoder representations and the cross-entropy loss per timestep for the decoder component [130, 275].

On the other hand, Transformers have become the core of speech representation models for many tasks, especially with unsupervised learning and self-supervised learning. Models such as wav2vec [14] or HuBERT [106] use Transformer as the backbone architecture combined with contrastive learning, so that the Transformer output representations can be clustered into meaningful clusters (such as with codebooks) and downstream speech processing tasks can benefit from initializing the parameters after the self-supervised learning process.

In the next chapter, the self-supervised learning models are one of the ingredients helping us in advancing multilingual speech recognition.

4 Multilingual ASR with Weight Factorization

After the previous chapter, we have shown that end-to-end speech recognition is possible with a powerful neural sequence-to-sequence model such as a deep Transformer network, with the performance being competitive with the previous approaches. An important outcome from this line of research is the *elimination of the mandatory lexicon* in the statistical pipeline. Because of how difficult such resource can be gathered in other languages, especially for many languages that are morphologically rich such as Finnish and German compared to the ubiquitous English. Without a lexicon, the workflow of constructing a speech recognition model for different languages would become more language independent.

In the past, adapting a speech recognition recipe [208] to languages different than English was challenging. Not only are pronunciation dictionaries required, but the raining process also relies on certain language specific configuration imbued in context-dependent phoneme clustering in building HMMs. The end-to-end approach merely requires an acoustic input and and a flexible character level or byte-pair encoding of the text [234] to quickly bootstrap a system.

As there are many languages being spoken in the world, there are two main benefits of constructing a speech recognizer that can recognize multiple languages at the same time (multilingual system). First, building monolingual systems requires a lot of effort and it is more industrially efficient to use one single pipeline on a dataset containing different languages. The multilingual pipeline, at best, can be indifferent than training a system for one single language and will result in economical benefits. Second, different languages may sound

total differently, yet still share acoustic features. For example people who speak Spanish or Portuguese can find Italian familiar in terms of pronunciation. Such similarity can be learnt from the data using representation learning algorithms such as deep neural networks and can theoretically benefit low-resourced languages.

On this front, the thesis has two main contributions to multilingual speech recognition. First, the novel weight factorization method is proposed to adapt neural architectures in multilingual settings. This method is devised based on the pure observation of the main component of neural networks which is feature projection in the form of matrix multiplication. Instead of using customized neural components as language specific modules as often implemented in the literature, this method can be the unified approach to factorize a network into shared weights and language specific weights. In terms of methodology, the method can be the superset of other language specific module approaches in the literature, and it is also proved to improve the performance in large scale multilingual experiments.

While weight factorization helps the model have a better weight distribution for each language to more efficiently use the resources and avoid the unbalancing problem when the the distribution of languages is not uniform, it is still undesirable that *all of the weights are trained from scratch*. This brings two disadvantages: there is no clear instruction for the model to focus the information on the shared components in the factorization, and it is generally difficult to optimize a large and complex network. The second contribution in turn is to apply transfer learning to alleviate these weaknesses. Transfer learning is the practice of using weights for another network with the same or similar architecture that has experienced unsupervised pre-training, a training method applicable when there is an abundant source of unlabeled data. This method has been widely applied in computer vision [139], language models [50, 49, 28] and recently audio data [14, 226] under the form of learning to reconstruct or structured prediction. Thanks to the advantage of weight factorization being architecture agnostic, it is trivial to initialize a network with pre-trained components and then add factorized language-specific components. In this work, due to the nature of sequence-to-sequence learning being a conditional language

model, it is possible to combine two different pre-training architectures for the input modality (audio) and output modality (text). In such case, the factorized weights would act as a *adaptor* to guide the model in each respective language.

Respectively, we investigated the effectiveness of using two pretrained models for these two modalities: wav2vec 2.0 for audio and MBART50 for text, together with the adaptive weight techniques to massively improve the recognition quality on the public datasets containing CommonVoice and Europarl. Overall, we noticed an 44% improvement over purely supervised learning, and more importantly, each technique provides a different reinforcement in different languages. We also explore other possibilities to potentially obtain the best model by slightly adding either depth or relative attention to the architecture.

But before going into the details, the next section is dedicated for the literature. Before the rise of the end-to-end neural models, there were various attempts of combining different corpora to train one multilingual model, but most of them are limited by either methodology - the statistical approach being not friendly for multilingual setups - or insufficient data. But the previous works always left us important suggestions to design newer systems.

4.1 Multilingual ASR literature

Multilingual speech recognition indicates the construction of one single speech recognition system that is capable of generating transcriptions for many languages. It is notable that, the statistical approach would require at least two models (acoustic and language models) so the definition also includes the use of single acoustic and language models in the system. The overall concept of multilingual ASR is to enable the system to receive multilingual inputs, and give the transcript output in the corresponding language.

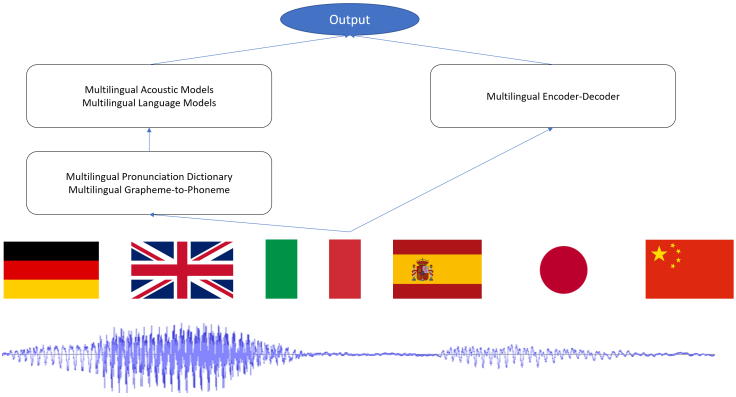


Figure 4.1: Multilingual ASR concept. On the left hand side, the previous approach requires many intermediate components to be multilingual, such as the pronunciation dictionary, a multilingual acoustic model towards the common phoneme set, and a multilingual language model. On the right hand side, the end-to-end approach only requires a shared vocabulary between the languages, while it is capable of mapping the acoustic input directly to the textual output.

4.1.1 Multilingual ASR with the statistical approach

Due to the availability of data, statistical speech recognition was mainly developed for English. When the method was also sufficiently developed for other languages, such as systems developed by IBM [41], Dragon [20], BBN [25], Cambridge [278], Phillips [54], MIT [70] and LIMSI [143], it was observed that the knowledge from an English system can be transferred to diverse languages like German, Japanese, French and Mandarin Chinese. Such observation illustrated that the method can generalize across languages and different languages can hold similar modeling assumptions. This observation later on, is especially applicable for the neural end-to-end approach thanks to the high level representations replacing discrete pipelines in the statistical approach. In contrary, the statistical approach heavily relied on the pronunciation dictionary which is an indispensable knowledge in the modeling process. Multilingual systems, therefore, require a *Global*

phone system using an unified phone system [227]. The reason here is that the acoustic model has to learn the likelihood of generating observation from phone states (in GMM-based acoustic models) or generating phone states from observation (with neural acoustic models). In a multilingual system, it is only meaningful to have a shared component implying in that the same modeling component is exposed to learn from different languages, otherwise the system is simply composed by a set of separated models assigned for each language.

Despite the advantage that the global phone system can be used to adapt for an unseen language with an existing pronunciation dictionary [229], it is obvious that the application for this system is heavily limited. Not only is it difficult to obtain a lexicon for a variety of languages, it is also challenging to have a widely-coveraged lexicon for languages that are morphologically rich such as German, Finnish, Czech. The global phone system also requires further efforts for languages that have a different acoustic systems than English/Romance family such as Asian languages. Different resources also have different alphabets such as WorldBet [95] or International Phonetic Alphabet (IPA) [142] and require further efforts to convert between symbols.

After the concern about the global phone system, the separation of acoustic/language models is also a problem when constructing multilingual speech systems. Early works often focused on one aspect of acoustic modeling while assuming that language models are given for the involved languages [228]. As a results, the earlier multilingual attempts often involved bootstrapping new systems using the parameters of a system from another language [265] resulting in a reduction of training time [245]. Specifically, the parameters of the Gaussian Mixture Models or the Neural Networks that estimate the emission probabilities can be borrowed to the next language as the starting point, if the languages share the same phone set.

The statistical approach also exhibits a high difficulty in multilingual modeling. Early works suggest that decoupling the models into language independent and dependent components [230]. However. due to the inflexiblity of the Gaussian Mixture Models and the dependence on a global phone system, this concept was difficult to execute and benefit.

For example, it was implemented in Subspace Gaussian Mixture Models [205] due to the property of this model as reducing the complexity of Gaussian Mixtures. Instead of typical GMMs that represent HMM states with a large number of parameters, the Subspace GMMs constrain the parameters to live in a relatively low dimension sub-space which is common to all the states. Using an observation that the variety of distributions corresponding to the voice of human articulatory tract is quite limited, SGMMs can use a compact representation to avoid overfitting that often happens with GMMs. When applied in a multilingual scenario, the shared parameters in the SGMMs can be trained on the multilingual data to benefit languages with limited resource [250, 29, 152]. However, the practical benefit of a multilingual setup is very limited in terms of reducing word error rate, when the relative improvement was often less than 5%.

Deep neural networks gradually replaced GMMs in the acoustic models. Sharing the neural parameters between languages proved to be b eneficial in the large scale multilingual experiment with 11 languages [92] when Catalan could benefit up to 10% using a multilingual acoustic model. However, since the language model remained monolingual, it was arguable that the language model is the performance bottleneck in the whole system and needs the same treatment with the acoustic model, as can be seen from the improvement of using a recurrent neural network language model instead of statistical ngrams [164]. In a similar manner sharing the bottleneck features [78] - a technique that learns high level representation as input features for GMMs - also yields little benefit in a multilingual setting. The experiences of the multilingual attempts with the statistical approach indicated that multilingual speech recognition would benefit the most in the end-to-end models when the restriction of global phone systems and language/acoustic model separation is removed.

4.1.2 Multilingual ASR with the end-to-end approach

The main difference between the two approaches is described in figure 4.1. The separated nature of the previous approach is not multi-

lingual friendly, especially on the multilingual phoneme representation. Due to the difficulty of obtaining a widely covered pronunciation dictionary for many languages, exotic methods such as multilingual grapheme-to-phoneme models [280, 217] being able to generate phonemes for arbitrary words are considered, however unfruitful with the risk of error. Subsequentially, the availability of end-to-end models made way for various research works about multilingual speech recognition. Early works attempted to apply CTC models and quickly realized two observations [171]:

- unlike the previous reports in GMM-HMM/hybrid systems, the performance of a single model is not necessarily better than monolingual systems. The gap is especially observable for the *high-resource languages* in the mixed dataset.
- the gap between multilingual models and monolingual counterparts is quickly shortened by increasing the model size of the multilingual models. This observation goes in line with previous multilingual research using deep neural networks for hybrid-HMM systems.

The second observation indicates that the models are under-parameterized or inefficiently constructed in contrast to the GMMs that are often over-parameterized. In fact, the size of the GMMs are often restricted by either feature reduction or subspace GMMs [205, 29]. Later on, this problem has been investigated further by scaling the experiments in terms of model sizes and data size. At the point of having $16,000$ hours of data coupled with models of billions of parameters, the multilingual models can even exceed the performance of the monolingual systems, especially in the low-resourced languages [209]. It is notable that the "billion parameter" is a relative estimation, while a neural network capacity is more accurately described with the number of layers and the number of hidden neurons per layer.

4.2 Weight factorization for multilingual neural networks

It is noticeable that the recent multilingual neural models are based on a semi-shared mechanism in which the largest body of the network architecture is exposed to all languages, while a smaller weight subset provides a language specific bias. This was shown to be more effective in a multilingual scenario than fully sharing the whole network[85, 123] since each language has certain unique features, and the single architecture often struggles to handle a variety of languages [201].

There are two main drawbacks that are typically presented in the existing implementations of the semi-shared mechanism. On the one hand, the implementations often depends heavily on a certain architecture being popular at the time, and the given improvement is going to be diminished when a new architecture evolves and the proposed architectural adaptation cannot be implemented on the new architecture. For example, the language-specifically biased attention [291] modified the self-attention architecture [255] specifically based on the assumption that each language can benefit from a bias added to the attention scores. They cannot be applied for models that abstract the attention scores by bypassing the expensive computation of these terms, such as FAVOR+ [38] or linear attention [129] whose purpose is to reduce computational cost for expensive Transformers. On the other hand, the language-dependent components might require a considerable amount of parameters and struggles to scale to the number of languages. For example, the language adapters added to the Transformer layers [19] are essentially feed-forward neural network layers being similar to the counterpart already in the shared Transformer body. A scenario with 20 languages consequently generates hundreds of these layers accounting for a large amount of parameters to be optimized. Adapters, however, remain to be one of the favourite methods in using specific models to control the desire language or task [105] in language processing.

In turn, the contribution of thesis is to address these two problems by proposing a multilingual architecture using a factorization scheme that is both effective and highly scalable with the number of languages

involved. Moreover, this scheme is applicable to any neural architectures as long as matrix-vector multiplication is the dominant operation. The key idea of our work is that each weight matrix in the shared architecture can be factorized into a shared component and multiple additive and multiplicative language dependent components. While each language is assigned with extra weights to learn distinctive features, simplicity and scalability are achieved by further representing those weights into as a rank-1 matrix, thus can be factored into two vectors. This method is demonstrated to be computational friendly with a minimal overhead and can be applied to a arbitrary neural architecture.

Model description A neural speech-to-text model transforms a source speech input with N frames $X = x_1, x_2, \ldots, x_N$ into a target text sequence with M tokens $Y = y_1, y_2, \ldots, y_M$. The encoder transforms the speech input into higher level feature vectors $h^X_{1\ldots N}$. The decoder jointly learns to generate the output distribution o_i based on the previous target tokens $y_1, y_2, \ldots, y_{M-1}$ while looking for the relevant inputs from the input via the attention mechanism [15, 255].

$$h^X_{1\ldots N} = ENCODER(x_1 \ldots x_N) \tag{4.1}$$

$$h^Y_i = DECODER(y_i, y_{1\ldots M-1}) \tag{4.2}$$

$$c_i = \text{ATTENTION}(h^Y_i, h_{1\ldots N}) \tag{4.3}$$

$$o_i = \text{SOFTMAX}(c_i * h^Y_i) \tag{4.4}$$

$$y_{i+1} = sample(o_i) \tag{4.5}$$

Notably, there is a large variety of model architectures that implement this Encoder-Decoder design. The core networks in the encoder and decoder range from LSTM [16], convolution/TDNN [68, 289] to self-attention [199] or even a mix of the above [80]

The universal multilingual framework [85, 123] employs a single model to learn on a joint training dataset containing multiple languages, which is different than the predating multi-way Encoder-Decoder approach [61].

Multilingual weight composition It can be seen that, the common ground of the aforementioned architectures is the usage of linear combinations of lower level features $X \in R^D$ which can be expressed as the matrix multiplication between input X and a weight matrix W. For example, the LSTM contains four different projections for its forget, input, output gates and candidate content [102], as can be seen in Equation 4.6.

$$f_t = sigmoid(W_{fx}^\top X_t + W_{fh}^\top H_{t-1} + b_f) \tag{4.6}$$
$$i_t = sigmoid(W_{ix}^\top X_t + W_{ih}^\top H_{t-1} + b_i) \tag{4.7}$$
$$\hat{c}_t = tanh(W_{cx}^\top X_t + W_{ch}^\top H_{t-1} + b_c) \tag{4.8}$$
$$o_t = sigmoid(W_{ox}^\top X_t + W_{oh}^\top H_{t-1} + b_o) \tag{4.9}$$

Similarly, the main components of the Transformer layers are self-attention layers and feed-forward layers. While the latter are fundamentally two layers of linear projections, the former is also comprised of linear projections that generate queries Q, keys K and values V from the input X:

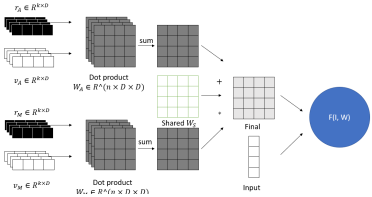
$$Q = W_Q^\top X \tag{4.10}$$
$$K = W_K^\top X \tag{4.11}$$
$$V = W_V^\top X \tag{4.12}$$
$$SelfAtt(X) = softmax(QK^\top)V \tag{4.13}$$

The main idea here is that each matrix multiplication $Y = W^T X$ in the multilingual model can be decomposed into a function of shared weights W_S and additional language dependent weights W_{ML} and W_{BL}

$$Y = (W_S \cdot W_{ML} + W_{BL})^\top X \tag{4.14}$$
$$= (W_S \cdot W_{ML})^\top X + W_{BL}^\top X \tag{4.15}$$

Here the added weights include the first multiplicative term W_{ML} that directly change the magnitude and direction of the shared weights W_S and the biased term W_{BL} provides the network with a content-based bias depending on the input features X. Each language maintains a distinctive set of W_{ML} and W_{BL} so that the whole architecture is semi-shared.

Figure 4.2: Weight factorization diagram. The *per-language* black and white parameters combine with the shared weight matrix to generate the specialized weight matrix W to transform input I.



Factorization for language-specific weights There is, however, an obstacle that both W_{ML} and W_{BL} require to be the same size with W_S, which makes the language dependent weights dominate the shared weights, while the intuition is the opposite. Fortunately, it is possible to use rank-1 matrices $\bar{W} \in R^{D_{in} \times D_{out}}$ that can be factorized into vectors [269, 277], for example with two vectors $r \in R^{D_{in}}$ and $s \in R^{D_{out}}$ such that $\bar{W} = rs^\top$ which reduces the number of parameters from $D_{in} \times D_{out}$ to $D_{in} + D_{out}$.

One drawback in this method is the lacking representational power of Rank-1 matrices. One solution is to modify the factorization into using k vectors per language so that there are k independent weight factors followed by a summation, which increases the rank of the additional weight matrices.

$$\bar{W} = \sum_{i}^{k} r_i s_i^\top \tag{4.16}$$

Computational cost The factorization above is applied to both W_{ML} and W_{BL} to ensure that the dominated force is still the shared weights, while each language at $k = 1$ is characterized by an additional $\frac{D_{in}+D_{out}}{D_{in}\times D_{out}}$ amount of weights. In a typical network architecture with D_{in} and D_{out} being typically $512 - 2048$, this amounts for $0.1 - 0.3$ percents of the total network's weights per language, therefore scalable to hundreds of languages while maintaining a reasonable cost[1].

About the time complexity, the amount of extra computation comes from generating the combinatory weight W from W_S and the multiplicative/bias terms W_{ML} and W_{BL}. Fortunately, this overhead coming from element-wise multiplication and addition is rather small compared to the matrix multiplication. More importantly, it is possible to utilize the optimized implementation of the original network[2] which minimizes the computational requirements of our approach.

Using a similar approach for Bayesian modeling, [269] proved that W does not have to be explicitly computed by multiplying r and s vectors to the input and output of the linear layer respectively, but their approach required to rewrite the graph operation for the core networks in popular deep learning frameworks.

4.2.1 In comparison to Related Works

In the world of speech recognition, training a single recognizer for multiple languages is not a thematic stranger [265] from Hidden Markov Model (HMM) based models [29, 152], hybrid models [92] to end-to-end neural based models with CTC [170, 131] or sequence-to-sequence models [252, 291, 290, 2, 126, 149], with the last approach being inspired by the success of multilingual machine translation [85, 123]. The literature especially mentions the merits of disclosing the language identity (when the utterance is supposed to belong to a single

[1] In practice this amount is even lower due to the weights in other parts of the network being not factorizable, such as the word embedding layers that account for a large portion of the network

[2] such as the CUDA implementations of LSTM and Self-Attention

language) to the model, whose architecture is designed to incorporate the language information.

One of the manifestations is language gating from either language embeddings [131] or language codes [170, 169] that aim at selecting a subset of the neurons in the network hidden layer. In our current approach, this effect can be achieved by factorizing further Equation 4.15 [269]:

$$Y = (W_S \cdot W_{ML})^\top X + W_{BL}^\top X \tag{4.17}$$

$$= (W_S \cdot (r_m s_m^\top)^\top X + (r_a s_a^\top)^\top X \tag{4.18}$$

$$= (W_S^\top (X \cdot s_m) \cdot r_m) + (r_a s_a^\top)^\top X \tag{4.19}$$

In Equation 4.17, the multiplicative matrix W_{ML} is factorized by two vectors r_m and s_m. The left hand size of Equation 4.19 shows us that the those vectors can be learned to gate the input vector X and the output of the linear projection $(W_S^\top (X \cdot s_m)$. This intuition also suggested us to initialize r_m and s_m to one-vectors similarly to normalization techniques [111, 12]. Since layer normalization often comes before the linear projection layers in Transformers, this scheme also helps our model to generalize to assigning to each language a different normalization scale and variance [287].

On the other hand, the right hand side of Equation 4.19 gives us the bias to the linear projection which has been used in either language embeddings [209] and customized attention layers with language biases [291].

A different line of research involves using language code [170] to differentiate language coming from a separate classifier. The language code is often trained separately and then mixed into the ASR architecture later [169] giving the lingual bias. Our method can provide a similar effect with end-to-end training and without architectural modification. The advantage of this method is to exploit unlabeled (transcript-wise) data to gather language-specific information.

Architecture wise, [19] makes the network language aware using language-dependently adaptive feed-forward layers at the end of each

Transformer block. While this method is able to be effective in translation [200] and speech recognition scenarios [291], it requires a considerable amount of parameters per language[3] and probably becomes incompatible with future architectures because it is specifically designed for Transformers.

The closest to our work is the parameter generator [201] that composes a weight matrix $W \in R^{D_{in} \times D_{out}}$ using a shared tensor $W_S \in R^{D_{in} \times D_{out} \times D_L}$ and a language embedding vector $L \in R^{D_L}$. The main disadvantage with that approach is that the amount of parameters linearly scales in the size of the language embedding D_L, and the whole body of parameters participates in every language. Our initial experiments cannot produce a reasonable result for a straight comparison, partly because the memory is quickly overwhelmed by the number of parameters.

For a larger context, weight factorization has been investigated to generate distinguishable yet cheaper copies of an existing network to allow for economical ensembles [269], Bayesian networks [57] or continual learning without catastrophe forgetting [277]. Similar ideas to use different weights for different languages have been investigated early on by [88, 261, 87] to rapidly adapt the system for different acoustic units, in which the idea of using frozen representation and adding new freely connected neurons (connectionist glue) [86]. Networks can also be expanded in a hierarchical way using mixture of experts [63], in an attempt to improve the HMM-based models that require tree-based models to cluster phonemes.

Application of factorization for neural ASR models Weight factorization can be effortlessly applied in both of the popular architectures for end-to-end ASR: the Long Short-Term Memories and the Transformer models.

The main structure of LSTMs is constructed based on four "gates" that control the information flow between the current inputs and the past

[3] Each feed-forward component accounts for around 25% the amount of parameters of each encoder block.

memory states. Each gate consists of two linear projection matrices mapping the inputs and the previous short term hidden states, and these projection matrices can be factorized. Bi-directional LSTMs employ two recurrent cells that run in two directions, so the number of parameters doubles compared to an unidirectional LSTM. In that case, there are 8 projection layers to factorize for multilingual learning.

On the other hand, Transformers have blocks of transformation layers with the main components being self-attention layers, cross-attention layers between decoder and encoder layers, and feed-forward neural networks. Each of these layers can be effortlessly factorized as follows:

The self-attention layer is characterized by a number of different linear projection, given an input $I \in R^{T \times D}$ with length T and number of features D.

- $Q = W_Q I + b_Q$
- $K = W_K I + b_K$
- $V = W_V I + b_V$
- $A = softmax(QK^T)$
- $C = AV$
- $O = W_O C + b_O$

The first three matrix multiplications creates queries, keys, values for the attention formulation by $softmax(QK^T)$ and the context vector $C = AV$. The output is the linear projection of C. The matrix multiplications with parameters W_Q, b_Q and W_K, b_K and W_V, b_V and W_O, b_O can be factorized. The cross-attention layers have the only difference in which the inputs of K and V are replaced by the encoder outputs.

The feed-forward neural networks are fundamentally two linear projections with an activation function upon an input $X \in R^{T \times D}$:

- $H_1 = ReLU(W_1 X + b_1)$
- $H_2 = W_2 + b_2$

In the situation X is batched, then the batch dimension and time dimension is group together as a single dimension for the matrix multiplication. The parameters W_1 and W_2 are factorized. With a common setting that quadruple the feature size of H_1 compared to X, a Transformer encoder layer has similar number of parameters compared to an unidirectional LSTM layer with the same input size.

4.3 Experiments

Datasets The effects of the weight factorization methods are measured on datasets publicly available including Mozilla Common Voice [8] containing up to 27 languages, Euronews [76] and Europarl-ST [112] having 4 and 9 languages respectively. The preprocessing steps include converting audio into 40-dimensional feature frames, and generating BPE for each language with 256 codes each. Only Japanese and Chinese are handled at character level[4]. All of the three mentioned datasets come with the predefined validation and test partition, which are used in our experiments.

Two experimental scenarios are investigated in our work: initially we work on a set of 7 European languages: German (de), Italian (it), Spanish (es), Dutch (nl), French (fr), Polish (pl) and Portuguese (pt) each of which contain at least 60 hours of training data. The second scenario later expands to a total of 27 languages of more origin and diversity.

Model and Training description The experiments are conducted with two model architectures, two of which are commonly used in end-to-end speech recognition [284]: a) LSTM-based Encoder-Decoder networks [183] in which the LSTMs have 1024 hidden units and the encoder is downsampled using two 3×3-filter convolutional layers, and b) Transformer networks [255] with relative attention [197] with

[4] Our initial experiments with joined BPE gave worse results for the 27-language dataset

weight factorization for this multilingual setup. For the Transformer, we use the Transformer-Big configurations in [255] with model size 1024 but with 16 encoder layers with stochastic layer dropout with the same setting as in [199]. We found that using $k = 4$ for the additive biases and $k = 1$[5] for them multilingual biases is sufficient.

All models are trained on single GPU by grouping a maximum $45,000$ frames per mini-batch[6], and the gradients are updated every 16 mini-batches with adaptive scheduling in [255] using the base learning rate 1.5[7] and $4,096$ warm-up steps. The inputs are masked with SpecAugmentation [183]. Given the large configuration, we train all models up to $150,000$ updates or up to 2 weeks. It is notable that the factorized versions have minimal overhead which results in a $10 - 15\%$ training speed reduction, while the adapter method requires at least 33% more.

Baseline models The comparison in the upcoming result section involves two previous works that were re-implemented. First, the language embedding was concatenated to the speech features and word embeddings at the encoder and decoder respectively which was used in [209]. Second, the language dependent adapters [19] were used. In this case, we use adapters in the form of feed-forward networks with $1,024$ neurons in the hidden layer. While theoretically the language embedding is a subset of our factorized network because the former is essentially a small set of weights dedicated for each language, the adapter network is fundamentally different because it requires extra layers, adding depths and nonlinearity levels to the overall architecture, while our factorization scheme keeps the interaction between inputs and weights unchanged.

[5] Partly because the initialization is desired to be 1.0

[6] speech inputs are often longer than their transcriptions, so grouping mini-batches by frames is more efficient

[7] The actual learning rate maximizes at around 0.001 and gradually decreases over the course of training

4.3.0.1 Experiments with 7 languages

The word error rates for each language using two baseline models (with Transformer (TF) and LSTM), their factorized versions and the TF with adapter [19] are shown in Table 4.1. Averaged over the 7 languages, the error rate is reduced by 15.5% and 7.2% rel. for the Transformer and LSTM respectively, and the improvement is significant across languages, unlike the Adapter technique which manages to reduce the error rate for 4 languages but is not better for the other languages.

Regarding the number of parameters, the Transformer and its factorized variation has twice as many parameters as the LSTMs, thus possibly explaining the improvement regarding performance. While this seems to contradicts the large number of parameters for the ADT model that needs 42% more space than the factorized TF, the ADT actually adds more depth (2 per TF block). This is a significant change to the architecture because with layer normalization, all languages share the same layer mean and variance at each level, while this is not changed with the adapter.

Experiments with 27 languages Under this condition, the factorization method maintains the improvement across all languages, with overall 26% rel. WER reduction in average for Transformer and 27.2% rel. for LSTM, as summarized in Table 4.2. Importantly, the factorized models are effective while using only 15% more parameters, while the ADT Transformer needs almost 1 billion parameters to achieve a 21.2% rel. improvement, due to each language requiring 2 more layers per block.

While the most resourceful languages such as German, Italian, Spanish and French observe the similar improvement compared to the 7-language experiment, the lower resource counterparts are often improved significantly compared to the baseline, regardless of the model architecture. The error rates on Japanese and Latvian testsets were decreased by 48% rel. compared to the base Transformer, and multiple languages were improved by 30% rel. including Arabic, Br, Cnh, Cv and Ta. The only language that remains a relative high error rate

is Dhivehi, in this case staying over 60% regardless of the architecture. One explanation for the large improvements regarding lower resource languages is that, the language weights are only learned to optimize for those particular languages, while the shared weights are frequently changed attempting to optimize all different language/task losses. This problem is often alleviated using learnable and weighted sampling [267] to help the gradients remain stable for the less frequently visited languages.

A direct comparison between two Transformer variations shows that the factorization is consistently better in 21 languages and the adapters yielded better results in 6, given the same time and computational constraints. While it is also possible for the adapters to obtain better performances by longer training, the presented results provide evidences that our proposed factorization scheme is able to outperform both the baseline and the deeper language adapter network without extensive tuning and with reasonable resources.

Table 4.1: Comparison on the 7-language dataset (WER↓). Our baseline models include the Transformers (TF), LSTM and their factorized (FTR) variations respectively. The last column is the Transformer with Adapter (ADT) [19].

Language	**TF**	**+FTR**	**LSTM**	**+FTR**	**ADT**
# Params	335M	350M	167M	172M	497M
de	15.78	14.62	15.75	15.53	14.71
es	16.06	13.47	14.66	14.09	14.81
fr	17.34	16.26	17.35	16.44	16.76
it	18.62	15.82	16.65	15.63	17.58
nl	26.61	22.33	24.18	22.57	31.84
pl	20.4	15.7	16.39	15.28	20.65
pt	25.8	19.3	23.21	19.49	25.19
Mean	20.08	**16.97**	18.31	17.00	20.2

Table 4.2: Comparison on the 27-language dataset. The models being shown include Transformers (TF), LSTM (TF) and their factorized versions (FTR). WER↓ .

Language	Hours	**TF**	**+FTR**	**LSTM**	**+FTR**	**ADT**
# Params	355M	416M	177M	194M	980M	
(ar)	49	26.2	**17.81**	28.73	20.02	16.56
(br)	7	51.85	**34.69**	71.53	40.49	40.21
(cnh)	2	52	**38.33**	62.19	36.59	55.18
(cv)	4	53.88	**33.11**	61.61	39.6	38.40
(de)	850	16.89	**15.62**	19.89	16.59	16.35
(dv)	18	71.63	**63.72**	80.18	64.82	65.23
(es)	400	16.05	**14.53**	18.41	14.82	15.27
(et)	19	33.95	30.43	39.63	34.26	**28.12**
(fr)	700	18.61	**17.24**	20.86	17.43	17.87
(ia)	6	49.86	33.24	48.39	**31.96**	42.40
(id)	9	28.78	**17.28**	32.9	20.22	22.79
(it)	250	20.76	**18**	21.99	18.07	19.60
(ja)	3	39.17	**20.44**	38.92	23.79	27.55
(lv)	6	66.17	**34.3**	66.66	37.93	43.57
(ky)	11	22.08	**17.17**	18.68	21.46	12.86
(mn)	11	42.03	**35.03**	46.42	38.5	34.12
(nl)	90	27.54	**23.75**	29.44	23.93	28.30
(pl)	120	21.81	17.8	19.92	**17.19**	18.75
(pt)	80	25.16	21.38	27.13	**21.37**	22.82
(ro)	30	39.39	32.15	34.7	**26.73**	41.71
(sah)	4	57.47	50.47	69.04	**49.2**	55.27
(sl)	5	49.73	22.01	48.92	29.66	**20.77**
(ta)	14	33.1	22.34	18.87	28	**16.36**
(tr)	20	6.04	5.16	4.99	8.29	**2.40**
(tt)	26	24.96	22.12	38.03	24.07	**21.83**
(zh)	56	24.05	**22.53**	33.01	23.54	25.99
Mean		35.4	**26.2**	38.5	28.0	27.78

4.4 Combination of factorization and unsupervised learning

Unsupervised learning in speech modality In the previous section, the multilingual architecture based on weight factorization [193] was proposed to more efficiently model the similarity as well as difference existing in a multilingual dataset.

However, the drawback of supervised learning is clearly seen in multilingual settings, as many languages are only poorly available. As a result, despite such promising potential, end-to-end multilingual models inevitably requires abundant training data because this approach combines acoustic models, language models and alignment into the same model [16, 31]. In fact, the labeled data necessary for the task is limited, and even more limited outside of English and other mainstream languages, making building a reliable speech recognizer for many languages even more challenging. One can even question if sequence-to-sequence is a sustainable approach without fully utilizing data as well as the hybrid systems.

On the other hand, the availability of unlabeled data is virtually unlimited, and more importantly they can also be available under the form of pretrained models. Since the release of Bidirectional Transformers or BERT [50] with masked language model pretraining and its success in applying for natural language processing, many Transformers [255] based architectures have been the silver bullet to tackle low-resource problems. Recently, in the speech domain, Transformers are also well adopted in supervised learning [199, 197] and unsupervised pretraining with contrastive predicting methods [226, 14]. Most importantly, these models can be effortlessly trained on a multilingual dataset and acquire the acoustic or syntactic information of many languages. Replacing the components in an e2e model with the pretrained ones is trivial and can potentially gain largely for multilingual recognition.

With the pretrained models available in our arsenal, this thesis set out to explore the possibilities of combining them for multilingual ASR. While the application wav2vec 2.0 and MBART50 individually is ubiquitous and the combination of them has been observed in speech

translation [150], to the best of our knowledge this is the first attempt in multilingual ASR, especially in a large scale with many languages with different data size.

More importantly, there are many improvements for the Transformers over the years that improve either modeling long range dependencies in self-attention [46, 197] or adaptive components for multilingual models [19, 193]. It is promising to also combine these techniques with the pretrained models for the best results.

With such motivation, we carried out experiments with 32 languages ranging from high to very low resource and explore the possibilities of using pretrained models. Our contribution is as follows:

- First, using multilingual pretrained acoustic and language models for the encoder and decoder respective brings a large improvement as a whole. However each pretrained module has a different influence on different languages, namely encoder pretraining is more impactful for languages with higher resources while the decoder counterpart is more effective for languages with medium-low resources. Surprisingly, many languages with extremely low resource (less than 5 hours) do not benefit much from this combination.
- Second, the language specific modulation techniques such as language adapters [19] and factorized adaptive weights [193] complement the two pretrained modules very well and have a strong impact on especially the low-resource languages mentioned above.

Moreover, there are different possibilities to integrate knowledge from pretrained models, and it is not necessary by simply replacing components. We also provided further analysis to the architecture, by showing that there are benefits to either improving the self-attention mechanism by adding relative positions during fine-tuning, or stacking the MBART encoders to the wav2vec counterpart.

This is the continuation of the line of work building multilingual ASR systems based on sequence-to-sequence neural networks [126, 209, 193]. The implementation is available for public at *https://github.com/quanpn90/NMTGM*

providing a highly CUDA-optimized implementation for both wav2vec and MBART which is potentially useful for the community.

Combination of weight factorization and transfer learning Transformer encoders can be used to learn useful representation from input masking and construction which is demonstrated in masked language model [50]. Following this trend, they are inevitably applied for audio signals, starting from learning to reconstruct the log-mel frequency features [120] to using quantization to learn latent variables [226].

In our work, the wav2vec 2.0 model [14] is selected to replace the typically randomly initialized encoder. It consists of three main components: a convolutional feature extractor that convolves and downsamples the raw audio input, a deep Transformer encoder that learn high level representation from the downsample sequence, and a quantization module to generate latent variables. During pre-training, the network optimizes for a contrastive loss function while masking the speech input. wav2vec 2.0 showed that it can outperform other semi-supervised learning approaches using finetuning with a CTC model.

Language Adaptive Components The development of multilingual models for either machine translation, speech translation or speech recognition often concern between versatility versus specialization [146]. The motivation comes from the assumption that there are features being shared between languages and at the same time each language requires to selectively represented, and networks are encouraged to change "modes" depending on the language being processed [88]. Since then, multingual model designers opt to use specific network components being presented for each language, ranging from weight generator [201] to adapters [19, 146] or recently adaptive weights adding scales and biases to each weight matrix in the whole architecture [193]. In this paper, the last two options are selected for investigation thanks to being computationally manageable.

On the one hand, Adapters plugged into pretrained models were introduced in computer vision [213] and later natural language process-

ing [105] and recently in Transformers for text/speech translation [19, 146]. They are materialized with a small multilayer perceptrons (MLP) with one hidden layer that acts as a *downsampler* (for parameter efficiency). This MLP is serialized at the end of each layer in the Transformer to help the network changes the feature distribution based on languages.

On the other hand, adaptive weights [193] was proposed based on the observation that neural networks evolve rapidly yet the core remains to be matrix multiplication. Therefore, it is possible to separate the weight matrix into a shared component W_S and language dependent *adaptive* scale W_{ML} and bias W_{BL}. The simple matrix multiplication $Y = WX$ becomes:

$$Y = (W_S \cdot W_{ML} + W_{BL})^\top X \tag{4.20}$$

In order to encourage the model to share parameters as well as keeping the parameters efficient, the adaptive weights are *factorized* by using the form of 1-rank matrices [269] which can be compactly represented as a dot-product between two vectors. This factorization can be established with k vectors per language so that there are k independent weight factors followed by a summation, which increases the rank of the additional weight matrices.

$$\bar{W} = \sum_{i}^{k} r_i s_i^\top \tag{4.21}$$

Equation 4.21 applies for all scale and bias matrices in the network.

Comparing two approaches, the advantage of the adapters is that they can increase the depth of representation in the network thanks for having an activation function in the downsampled layer. In contrast, the adaptive weights have the advantage to directly affect each layer function, such as the QKV-projection layer in self-attention, instead of applying a new function on the output of the layer. In the Transformers specifically this particular combination between a pretrained encoder and a decoder, the cross-attention layers in the decoder are left with weights untrained to connect two modalities audio and text. By being

able to directly alter this function, the advantage of the adaptive weights are even more considerable.

4.5 Experiments

There are 32 languages We report the error rates on the test sets of CommonVoice and Europarl. It is notable that both of the pretrained model (wav2vec) and the available supervised training data are mostly read speech, leading to the curiosity about the performance on a more spontaneous setting.

Our speech recognition experiments are conducted using the public dataset including CommonVoice [8] and Europarl [112] as training data.

For the progressive comparison, we trained a competitive supervised model using the Transformer large configuration [255] with 24 encoder layers, 8 decoder layers and relative attention [197]. For transfer learning, we used the wav2vec 2.0 model pretrained with 53 languages [13] with the large configuration that has the same hidden size with our initial model. It is notable that the data used in pretraining is heavily biased to read speech including CommonVoice and Multilingual Librispeech [210]. For pretrained language models, MBART50 [249] with the same hidden size is used. For language specific modules, we use adapters with hidden layer size 512 and adaptive weights with $k = 8$ for bias and $k = 1$ for scale matrices, so that they have the same number of additional parameters per language. For training, we used an effective batch size of around 2.84 hours of data per update, together with a linear decay learning rate that peaks at 0.001[8]. The supervised model takes $150K$ updates to converge, while the model with transfer learning takes $50K$ updates.[9]

The performance impact of the pretrained modules are fully presented in Table 4.3. The test data here is the combination of CommonVoice

[8] The decaying equation follows the same in Attention is all you need [255]
[9] Training was possible using 4 NVIDIA A100 GPUs.

and Europarl wherein the latter is available. Since the languages widely vary in terms of data size, we divide them into three groups that have less than 10 hours (very low), between 10-100 hours (low) and more than 100 hours (medium-large) of training data. Notably, our supervised model outperformed the previously reported error rates [193, 104].

Impact of acoustic pretraining Compared to the Transformer model without any pretraining (**TF**), having the encoder pretrained with XLS-R (**W**) brings a substantial improvement to the average error rates by 18%, and this enables many languages to reach 10% errors or lower. Across the three groups however, there is a clear difference in impact. While the high-medium group enjoys a 20% decrease in error rate, this figure drops to 16% in the first group, and the third group is only improved by 4%.
While this is rather surprising, compared to the previously reported of wav2vec 2.0 pretrained models on very low resource settings [14], it is explainable by the difference of the approaches used in their and our works. The pretrained acoustic model has often been used directly with the CTC loss function to generate characters which heavily requires an external language model. In our setting, we are limited in both acoustic and text resources for the languages in the third group, and data scarcity makes learning to align from attention [31] even harder.

Effects from pretrained language model With that observation, initializing the decoder with the MBART pretrained model is expected to alleviate the data scarcity problem. In fact, comparing the model with two pretrained modules (**WM**) with the previous one showed a benefit of 15% error reduction for the former.

Some languages have worse performance with the MBART decoder, such as Arabic, Turkish or Thai. This can be explained as a negative effect of the large byte-pair encoding [234] model shared between many languages originally used with MBART training. A large vocabulary size of $250K$ allow for large granularity which is far from characters or phonemes, the units that speech recognition models are

often trained upon. Nevertheless, this is apparently not a problem for most languages.

Analyzing the individual performance of each group, the medium-large group is not impressively improved with just a 5% reduction. This result shows that having an additional six layers of decoders and even with pretrained weights only has a minimal effect on the result and network depth has a clearer impact on the source side than the target side in end-to-end speech recognition [199]. Probably the model does not struggle with the test data in terms of syntax, despite the fact that the text data here is not comparable to typical language models. Nevertheless, the second group receives a clearer merit from the pretrained language model, by a 27% improvement, and totally 39% improvement compared to the original multilingual Transformer. Even with a mismatched pretrained weights for the cross-attention module, it is still a noticeable improvement coming from the pretrained self-attention, feed-forward and layer normalization weights. The languages benefiting the most are Romanian, Estonian, Czech, Turkish, Indonesian, Swedish and Ukrainian.

The effect on the third group is modest at 11% reduction and the error rates remain very high for Urdu (ur), Kazakh (kk), Finnish (fi), Latvian (lv) or Vietnamese (vi). Many of the languages are also syntactically with many morphological word forms, such as Finnish, making recognition even more challenging. The most surprising improvement, however, comes from Slovenian that massively reduces from 26% to 14.6%. The overall struggling mainly comes from data scarcity which is not adequate for the decoder cross-attention layers.

Effect of the language-specific modules As can be seen from Table 4.3, both techniques are able to help the model generalize better in all language groups. Most importantly, the very low resource group witnesses 27% and 26% improvement using adaptive weights and adapters respectively, compared to the baseline model with two pretrained modules.

In order to quantify their impact, we proceeded to freeze all of the pretrained parameters and only fine-tuned the language-specific pa-

rameters. There is a difference in how each technique handles this situation. With the presence of only adapters, the errors in all languages deteriorate rapidly in all language groups. Many languages in the low-group experience very bad results including Romanian, Arabic, Chinese, Lithuanian and especially many members within the very-low group exceed 90% error rates. While this is unexpected, we can see that the main problem here is the cross-attention layer which is not familiar with the inputs coming from two modalities. The adapters are not able to drive the bad context vectors (weighted-sum of the encoder inputs) into meaningful representation in the low resource condition.

The adaptive factorized weights do not have this problem because they directly alter cross-attention. As a result, the performance is much better than the adapters even though they still fall behind the baseline without language-specific modules.

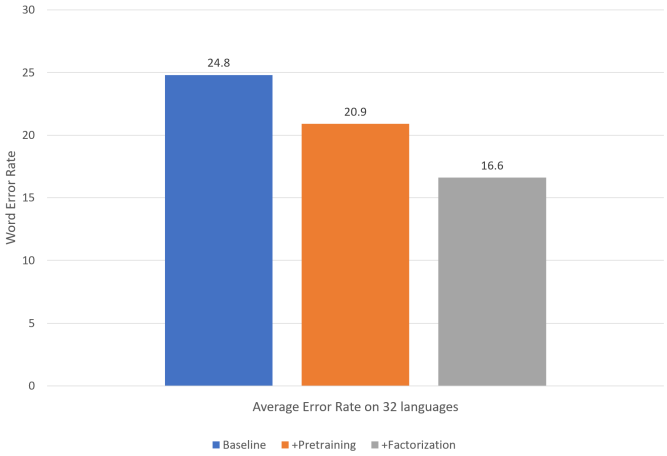
Further analysis In the previous sections, we presented the most important enhancement for our multilingual setup coming from the pretrained modules and the language adaptive components. The success of using the language adaptive components in various places, either breaking the *layer dynamics* with adapters or the *function dynamics* with the adaptive weights suggests that further improvement can be found by adding more information to the system instead of treating the pretrained model as an immovable black box.

Here we follow the implementation in [197] to add relative positions into the wav2vec model, by adding the content-position interaction together with the content bias and position bias to self-attention to all self-attention layers of wav2vec with factorized weights. The additive information allowed for the reduction of the average error rate to 16.04% (3% improvement) and especially an 12.7% on the large-medium group. This evidently helps the model learns better with adequate data.

Surprisingly, an alternative attempt to enhance the encoder simply by stacking the MBART50 encoder on top of the wav2vec encoder (without any length conversion) yields similar results, compared to the baseline **WM**, it improves the large-medium group by an impressive

18.3%, yet only 3.2% overall. Not only does stacking the encoder increase the encoder depth, it also helps the cross-attention layers because they are familiar with the output of the text encoder during training. Training a stacked model with the adaptive techniques would probably result in the best model in our experiments, however it was unfortunately beyond our computational tolerance.

Figure 4.3: Summary of the results averaged on 32 languages.



4.6 Conclusion

In this chapter, we introduce an architectural approach to improve the performance of a single end-to-end model in multilingual speech recognition. The final result can be seen in Figure **??**, in which the combination of weight factorization and unsupervised pretraining for two modalities is able to reduce the error rate by 33% relatively compared to a powerful Transformer baseline. The main reason for this improvement comes from the network being well initialized using the pre-trained weights from unsupervised pre-training techniques,

wav2vec 2.0 and MBART50 respectively. Moreover, weight factorization helps the network balance the gradient of each language during training, that especially helps the low-resource languages in the mix because their gradients are dominated by other languages during the training process.

Weight factorization is a reasonably efficient and scalable method (to the number of languages) to generate a sub-network per language. Compared to the more expensive adapter approach, weight factorization showed that it can be applied for any network and still provide a better performance, which is shown when we only train the per-language parameters.

The weakness of weight factorization lies in the fact that a certain language identification is required, in order to direct the network to the language weights. There are situations in which the language is not given, and even code-switching is used. It is still possible to use weight factorization is such cases, especially with a language predictor. The main challenge in this the code-switching situation is that the language can change within one utterance, and the network would have to process each frame differently based on the predicted language. In theory, this is no different than handling the whole utterance with one language, however it is computationally unfriendly in most modern Deep Learning frameworks such as PyTorch [185].

Table 4.3: Performance(WER↓) on the CommonVoice-Europarl dataset. Models include Transformers (**TF**), with wav2vec pre-training (**W**), with wav2vec and MBART50 (**WM**), with adapters (**WMA**) and factorized weights (**WMF**) and the version w/ frozen pt. weights.

Language	Hours	**TF**	**W**	**WM**	**WMA**	**WMF**	**FWMA**	**FWMF**
(de)	1050	10.4	8.1	7.8	7.7	**7.2**	11.8	10.0
(nl)	150	13.2	8.4	7.7	7.4	**6.8**	10.2	9.0
(fr)	800	15.2	12.7	12.12	11.62	**11.2**	16.7	15.1
(it)	325	11.5	8.7	8	7.8	**6.5**	12.8	10.5
(fa)	293	5.5	4.8	3.9	4.2	**4.0**	6.4	7.2
(pl)	145	11.5	10.5	9.2	9.1	**7.6**	14.8	12.2
(pt)	120	14.0	10.9	10.0	9.5	**6.0**	18.5	12.8
(es)	400	10.9	8.0	7.6	7.4	**6.2**	11.2	9.8
(ru)	148	10.0	8.7	5.5	5.7	**5.4**	20.1	10.1
(ta)	198	28.6	24	**20.2**	20.7	21.0	31.4	31.5
(th)	133	2.8	**2.6**	3.3	3.4	3.2	5.1	4.5
Average		12.1	10.3	9.5	9.3	**8.6**	15.8	13
(ro)	45	18.1	21.2	15.8	13.7	**10.12**	42.0	15.7
(ar)	85	21.1	**15.8**	18.7	17.6	18.2	31.2	23.2
(et)	32	30.4	22.1	14.8	15.1	**13.2**	32.5	21.5
(ja)	26	13.0	10.3	8.3	7.91	**7.9**	21.8	11.5
(zh)	63	25.9	16.7	15.2	**14.6**	14.7	37.6	18.2
(cs)	49	19.8	15.8	10.0	9.2	**8.4**	16.4	12.7
(lt)	16	43.3	37.9	31.9	26.7	**25.5**	71.3	30.0
(tr)	30	10.4	13.6	7.5	8.4	**7.5**	9.8	10.1
(id)	23	14.0	13.6	7.5	7.5	**6.6**	9	8.7
(mn)	12	49.8	35.1	26.2	26.0	**24.3**	90.4	32.0
(sv)	35	24.7	20.6	14.3	13.0	**12.3**	17.5	16.3
(uk)	56	14	13.4	7.6	8.3	**7.4**	11.4	14.8
Average		23.7	20	14.5	13.7	**12.5**	32.7	17.4
(lv)	6	41.9	57.0	41.01	**22.3**	**22.3**	79.1	25.0
(vi)	3	49.5	53.5	46.1	35.6	**34.0**	104.1	38.3
(ka)	6	58.6	48.0	48.3	33.0	**32.09**	130.3	39.0
(sl)	9	20.5	26.5	14.6	10.4	**9.1**	10.5	12.8
(fi)	6	54.2	48.5	41.0	31.4	**30.0**	109.2	35.2
(hi)	8	46.1	46.4	36.4	28.6	**27.6**	99.5	31.1
(gl)	7	26.5	15.3	15.3	11.2	**9.8**	48.8	16.0
(ur)	0.6	78.0	68.2	62.3	**56.6**	57.0	104.1	70.0
(kk)	0.73	86.5	76.8	86.4	**60.6**	61.0	104.3	70.03
Average		51.3	48.9	43.5	32.2	**31.4**	87.8	37.5
Overall		30	24.8	20.9	17.4	**16.6**	41.6	21.6

5 Continual Learning in Multilingual ASR

5.1 Catastrophic forgetting in learning new languages

With the conclusion of chapters 4 and 5, we have come to the Neural sequence-to-sequence models are applied to automatic speech recognition with a great success [31, 199, 183, 174] and this model can be easily extended for the multilingual scenario in which the training data consists of multiple languages [291, 209, 193]. These models are able to improve quality especially for the lower-resource languages since the acoustic representation can benefit from sharing with other languages.

Despite the success, most multilingual research works are limited to an assumption that all of the languages are available at once. This approach implies that having new languages to the system requires adding the new training data and retraining the model with the new data combination. Neural network systems can lose information about the previous datasets when being trained or fine-tuned on a new dataset. The weights in the networks are shifted towards minimizing the loss for the new dataset and are no longer in the region of having good performance for the older ones. This has been known as *catastrophic forgetting* [62] when the old information is abruptly lost when the model parameters are changed during the new training process. In multilingual speech recognition, fine-tuning a pretrained system on a new language can quickly deteriorate the performance of the old languages over the number of updating steps. The deterioration speed

depends on the amount of weights being changed in the process, a full update can quickly destroy the performance (by having error rates over 100%).

In practice, giving a multilingual system an ability to continually adding new languages into the capacity without performance loss in the old languages is very beneficial. For instance, it is not necessary to retrain the system with the whole amount of data including both old and new languages, saving time and effort. The demand is especially important when the training data cannot be stored after training, due to privacy reasons or simply to clear the storage for new data.

Catastrophic forgetting has often been countered by a number of approaches. Prominently *progressive neural networks* [221] were specifically designed for this purpose by allocating new network parameters when new tasks are introduced. By fulfilling the condition that the new weights do not interfere the topology for the older tasks, catastrophic forgetting would not happen. In opposite to progressive networks that linearly expand in space with respect to the number of tasks, *regularization-based* approaches look for 'empty space' in models that are often overparamaterized with redundant weights [232]. Most prominent among this approach are elastic weight consolidation [135] or synapse learning [283] looking for redundant weights based on importance metrics such as input/output magnitudes or variances of derivatives [184]. Recently, distillation based methods are widely applied when the model is also trained with the output from the counterpart from the previous approach.

Weight factorization as mentioned in the previous chapter, can be considered as a variation of progressive neural networks, in the sense that transcribing each language is a separated task for the model. Each language is provided with a set of weights for each matrix multiplication of the network, so the knowledge of each language is stored in two sections: the majority in the shared body of the network, and also in the factorized specific weights with much lower capacity than the shared weights. With that observation, when the network is exposed to a new language, it is possible allocate new factorized weights for this language. This idea, in the past, was materialized in controlling the learning capacity of different phoneme types [263, 260]. If the

shared body of the network contains language independent feature detectors, fine-tuning the new factorized weights is promising as a good start to learn new languages without catastrophic forgetting, because the weights participating in recognizing the old languages are untouched.

In continual learning for multilingual speech recognition, the desiderata is not limited in just catastrophic forgetting, however, but also other factors:

- Forward transfer: adding new languages to the current multilingual model can ideally obtain the performance similar to when having them in the initial training.
- Backward transfer: catastrophic forgetting is avoided for the previously learned languages, ideally adding the new languages should not affect the performance for the previously learned ones.
- Optimal training cost: the process of learning new languages should be economically better than re-training all languages from the beginning, in terms of training speed and storage.

In a continual learning scenario for multilingual speech recognition, the main challenge is how to use the new data of the new languages without affecting the representation of the previously learned languages. If we fine-tune the model on these data (by using gradient descent based on the loss function and updating the parameters like training a normal model), then the parameters will be shifted away from the original position which is the optimal point for the previously learned languages, as illustrated in Figure 5.1. Consequently, the model would achieve very bad performance on the previously learned languages.

When we factor in the specific Encoder-Decoder architecture for end-to-end speech recognition, certain problems arise when the models are required to learn new languages. The first problem lies at the *output layer*, which produces the likelihood of each token in the vocabulary per timestep during generation given an acoustic input. When being exposed to new languages, new words in the new languages can be added to the vocabulary, leading to the change of the weight matrix in

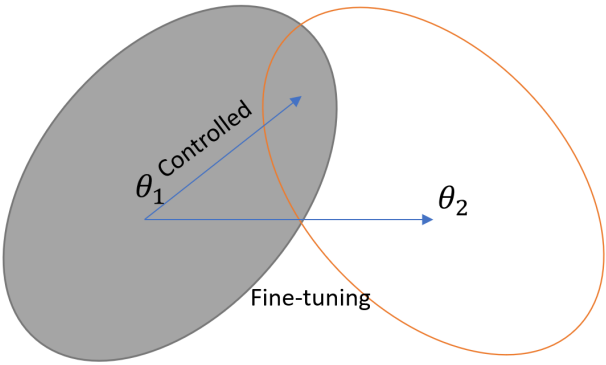


Figure 5.1: Illustration for the cause of catastrophic forgetting when the optimal parameters for the first language θ_1 are shifted towards the optimal parameters θ_2 for new languages, which unfortunately are far away from the original positions.

this layer consisting of parameters being untrained compared to the tokens in the previous languages. The second problem is a genuine architecture without any language aware mechanism (such as weight factorization) does not have any language separation mechanism in the architecture, updating the parameters for new languages would fall into the illustration as in Figure 5.1.

In order to realize the desiderata, the strategy is to combine three different techniques: *transfer learning, weight factorization and elastic weight consolidation*, by the following reasons. On the one hand, neural sequence-to-sequence models rely on a fixed vocabulary output, which is likely to be changed when the models are exposed with new languages. Multilingual pretrained language models [157] can relax this problem, by providing the syntax knowledge of the new languages, so that the output layer can be easily applied for new languages. On the other hand, Weight factorization [193] facilitates multilingual models by decomposing each matrix in the neural network into a shared component and per-language additive and multiplicative factors. Motivated

by progressive neural networks [221], this modeling scheme can prevent catastrophic forgetting by offloading information in the language specific weights. Meanwhile, elastic weight consolidation [135] can be used to find the "empty space" in the shared components of the networks to accommodate new languages.

The combination of all three techniques is applied in our experiments with 27 languages involved with the Transformer based model [199, 255]. In the process of learning these languages in different iterations, we were able to minimize the effect of catastrophic forgetting for previously learned languages, while the performance of the new languages are still comparable or potentially competitive with training all languages from scratch. Our contribution in continual learning, to the best of our knowledge, the first application in learning to transcribe languages.

5.2 Elastic Weight Consolidation

When a model with its parameters θ is trained sequentially with one task after another, $T_1, T_2, T_3, \ldots, T_N$, a single estimate of θ via gradient descent on the current task would lead to catastrophic forgetting. EWC reformulates the problem in a Bayesian fashion, in which the Bayesian posterior distribution $p(\theta|T_1, T_2, T_3, \ldots, T_N)$ over possible values is considered.

For the latest task T_N that the model learns, we use Bayes's rule to condition on the new training data of this task:

$$p(\theta \mid T_N, T_{N-1}, T_{N-2}, \ldots, T_3, T_2, T_1) \tag{5.1}$$

This full posterior is intractable since the denominator is rather unknown. However, the conditional distribution can be ignored when we optimize the weights θ to maximize the posterior because it is a constant with respect to θ. In the case of two tasks, the log posterior takes the form:

$$\log p(\theta \mid T_2, T_1) = \log p(T_2|\theta) + \log(\theta|T_1) - \log(T_2|T_1) \quad (5.2)$$

Equation 5.2 shows that in order to maximize the log posterior, we can omit the $\log(T_2|T_1)$ term. On the other hand, the first term $\log p(T_2|\theta)$ is also the log likelihood of the data for the second task, and is often represented using a neural network with parameters θ. For example, in speech recognition we model the conditional probabilities of the output sequence given the speech signal.

The main focus of EWC lies in approximating the second term practically. Even though the posterior of θ given the first task T_1 is intractable, during the training session on T_1 we obtained the optimized weights that maximize the log likelihood of T_1:

$$\theta^*_{T_1} = argmin_\theta - \log p(\theta|T_1) \quad (5.3)$$

Following the Laplace's method of approximating a log posterior with a second order Taylor approximation around the maximum [161, 163], equation 5.3 can be rewritten as:

$$-\log p(\theta|T_1) \approx \frac{1}{2}(\theta - \theta^*_{T_1})^\intercal H(\theta^*_{T_1})(\theta - \theta^*_{T_1}) \quad (5.4)$$

In which $H(\theta^*_{T_1})$ is the Hessian of the $-\log p(\theta|T_1)$ with respect to θ and evaluated at $\theta^*_{T_1}$. EWC makes a further assumption by approximating the posterior as a Gaussian distribution with mean given by $\theta^*_{T_1}$ and a diagonal precision given by the diagonal of the Fisher Information matrix $F(\theta^*_{T_1})$. Unlike Hessian, $F(\theta^*_{T_1})$ can be efficiently computed using the first-order derivatives which can be obtained with back-propagation. More importantly, at the local minimum, $F(\theta^*_{T_1})$ is equivalent to the second derivatives (the Hessian).

The loss function for EWC is then formulated as follows, based on the aforementioned approximations.

$$\log p(\theta|T_1, T_2) \approx \log p(T_2|\theta) - \frac{1}{2}\sum_i \lambda F(\theta^*_{T_1})_i(\theta_i - (\theta^*_{T_1}))^2_i \quad (5.5)$$

Intuitively, the second loss term regularizes the weights θ to stay close to the optimal weights in the previous task, but each weight is weighted by an importance denoted by the diagonal Fisher. λ is a hyper parameter weighting the involvement of the regularization term in gradient descent optimization.

Further explanation from [109] also suggests that the term $F(\theta^*_{T_1})_i$ should be weighted by the number of samples n T_1.

EWC computational cost Using EWC allows for removing the previously used dataset T_1 and EWC retains the knowledge in T_1 via the diagonal Fisher $F(\theta^*_{T_1})_i$, representing the *importance* of each weight in the model when learning on this dataset. Since each parameter θ_i requires 1 scalar in the Fisher and 1 scalar for $(\theta^*_{T_1})_i$, the maintenance cost for EWC is 2× the number of model parameters.

Extension for more than two tasks When it comes to applying EWC for more than two tasks, it is controversial in terms of formulating the loss function. In the original work [135], it was suggested to use multiple penalties centering around subsequent approximated posteriors $F(\theta^*_{T_1})$, $F(\theta^*_{T_2})$

However, this approach implies the requirement to store additional diagonal Fishers for each experienced dataset. Since these tensors are required to be simultaneously present in the computing device's memory for the loss function, this requirement quickly becomes a burden when the model experiences new languages.

The other approach [109] argues that simply accumulating the Fisher over time is adequate to build the loss function based on the weight obtained on the latest dataset.

$$\begin{aligned}\log p(\theta|T_1, T_2, \ldots, T_N) &\approx \log p(T_N|\theta) \\ &- \frac{1}{2}\lambda \sum_i (F(\theta^*_1)_i + F(\theta^*_2)_i + \cdots + F(\theta^*_{T_{N-1}})_i)(\theta_i - \theta^*_{T_{N-1}})_i)^2\end{aligned} \tag{5.6}$$

The accumulation of the Fisher information implies that the information of each parameter with respect to all datasets is the sum of the Fisher diagonal for that parameter. The algorithm can now be applied

recursively for many tasks with a constant cost of memory (because we can also discard the previous Fishers and trained weights).

Variants In practice, EWC is often considered as a regularization based continual learning approach [182], which can be intuitively interpreted from the loss function making the parameters important with the old tasks to have less discrepancy with the old parameters. The importance factor, however, can be manifested differently. For example, Synaptic Intelligence [283] computes the change in loss over an entire trajectory through parameter space by summing over all infinitesimal changes. The importance for each weight is the contribution to changes in total loss, and can be estimated online as the running sum of the product of the gradient with the parameter update. The Memory Aware Synapses [4] use the model output instead of the data to compute the gradients for importance, thus being able to apply for unsupervised learning.

It is also worth mentioning that, the regularization term in EWC can also cause learning difficulties when parameters change too slowly. In order to combat this, either an additional renormalization step [141] or Huber regularization [156] is applied to prevent the regularization term to be far larger than the main learning objective.

5.3 Combination of Weight Factorization and EWC

While EWC represents a continual learning approach relying on regularization, it is also possible to avoid catastrophic forgetting by allocating new weights when the model is exposed to a new task. There are many methods that are designed for task-based and class-based incremental learning such as Progressive Neural Networks [221]. The main criticism of such network design is the linear cost in terms of number of parameters per task. The same criticism can be also applied for the adapter based networks [19] which controls the output of the network based on blocks of feed-forward neural networks assigned for the task specifically.

In the context of multilingual speech recognition, it is possible to grow the network in an agnostic manner with respect to the network architecture and more importantly, with a sub-linear cost using weight factorization [193].

Main Idea Neural networks are composed by layers containing matrices of weights being multiplied with input vectors/matrices to generate output features for the next layers. Therefore, weight allocation per task can be manifested at this fundamental level, by factorizing a basic $Y = WX$ equation into:

$$Y = (W_S \cdot W_M + W_B)X \tag{5.7}$$

where $W_S \in \mathbb{R}^{D_{in} \times D_{out}}$ is the normal projection weight shared for all languages. $W_M \in \mathbb{R}^{D_{in} \times D_{out}}$ and $W_B \in \mathbb{R}^{D_{in} \times D_{out}}$ are the multiplicative and bias terms that are language exclusive.

In order to keep the number of parameters in check as well as encouraging the model to share more information between languages instead of partitioning into the exclusive terms, each language-dependent matrix W_M or W_B is further factorized into dot-products of vectors $r \in \mathbb{R}^{D_i n}$ and $v \in \mathbb{R}^{D_o ut}$.

$$W_M = r_m \cdot v_m \tag{5.8}$$

$$W_B = r_b \cdot v_b \tag{5.9}$$

The capacity of each factor is controlled via an additional hyperparameter k that increases the rank of W_M and W_M via adding multiple 1-rank matrices:

$$W = \sum_{i}^{k} r_i \cdot v_i \tag{5.10}$$

With the value of $k << D_{in}$ or D_{out}, the cost of adding each language is $\frac{2k}{D_out}$ number of parameters, assuming $D_{in} = D_{out}$[1].

Weight factorization for incremental learning This method can be applied for incremental learning by freezing the shared weights after the initial learning session [269]. When a new language is exposed, the system allocates new weights for this language specifically without tampering the weights of other languages. Using this approach, the network is ensured to keep the previous knowledge untouched while still having a multilingual body structure for the new language (otherwise the new weights would be insufficient), as depicted in Figure 5.2. However, it is important to note that the quality of learning greatly depends on the first learning session, when the shared weights need to capture as much global information as possible. Unlike progressive neural nets, the weights allocated to each tasks are decoupled and the ability to forward transfer is limited.

Combination with Elastic Weight Consolidation As analyzed above EWC and Weight factorization are both applicable for incremental learning, but with different approaches entailing different strengths and weaknesses. Thanks to having different natures, the methods can compensate for each other.

- During learning language L, the network is allocated with weights W_L consisting of the adding and multiplicative terms in weight factorization.
- After learning each language or a set of languages, Fisher Information diagonals for the main weight matrices in weight factorization, as well as other shared weights in the network, are computed.
- When learning new languages, we apply EWC for the main weights. The other factorized weights are naturally updated for their corresponding languages, and frozen for the previously learned counterparts.

[1] Its actually much lower than that, because the network may contain layers that do not need to be factorized, such as the output layer, or layer normalization

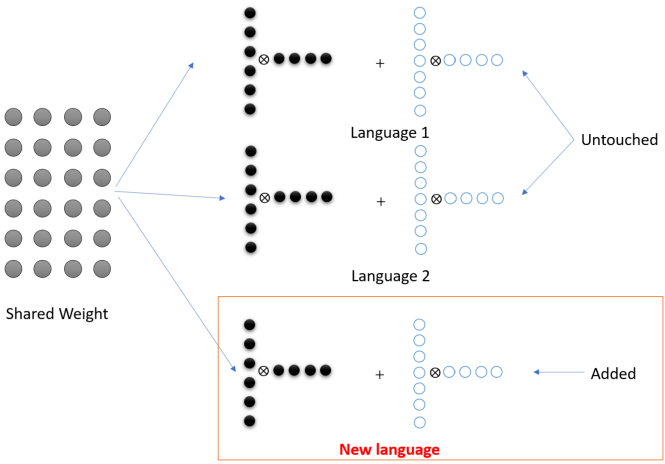


Figure 5.2: Illustration of Weight factorization for continual learning. The orange box indicates the new portion of weights being added for the new language, while the previously learned languages are kept intact.

On the one hand, the main problem of EWC is that looking for available space in the network can be challenging, allocating new weights for new tasks can alleviate this problem. On the other hand, EWC can help weight factorization in terms of transferring the knowledge between tasks.

The illustration is shown in Figure 5.3. EWC tries to find a compromised state for the weights to not stay out of the good regions, but it is also likely that it will end up having bad performance for both languages. With the intervention of per-language weights, it is possible to bias the weights back to the optimized states. While this strategy involves a cost for each new language, it is possible to rely on efficient methods such as Weight Factorization.

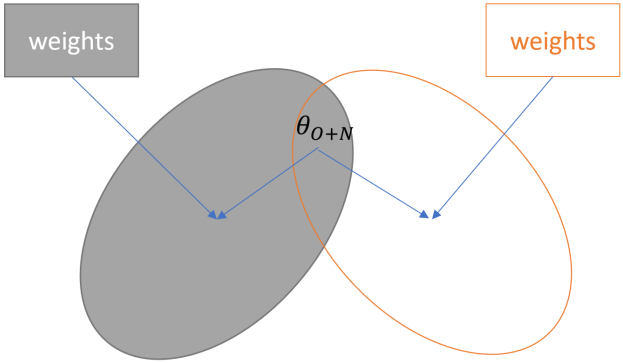


Figure 5.3: Illustration of the combination between EWC and Weight Factorization.

5.4 Continual learning experiments

Dataset Available multilingual datasets provided an ideal experimental ground for life-long learning new languages. We use 27 languages from Mozilla Commonvoice [8] and Europarl [112] for our language pools as shown in Table 4.2.
ASR Model We used the Transformer model as the base network for speech recognition. The architecture configuration is based on the wav2vec 2.0 model for encoder transfer learning [14] and MBART50 for decoder initialization [157, 249]. As mentioned before, one of the main reasons to use transfer learning, apart from achieving better performance than random initialization [190, 150], is to ensure that the output layer can contain the languages to be added without additional word embeddings. Despite the experiments being limited with the languages covered by the pretrained language models, it still remains practical thanks to the current coverage reaching nearly 200 languages [44].

There are two scenarios being considered in our experiments. In the first scenario, all of the languages in the second group are included in

the incremental learning session, while in the second one we further divided the languages into four different groups entailing four different sessions. In both scenarios, we measure the performance of the newly added languages as well as how much performance retained in the previously learned languages.

Implementation Our models are implemented in PyTorch [185]. The encoder is built upon the multilingual wav2vec 2.0 pretrained model [14, 42] and the decoder is essentially the MBART50 [157]. Such combination has been proven to be extremely effective in speech-to-text tasks [150, 190]. For the weight factorized model, the main weights are initialized with the unsupervised pretrained model, while the fast weights (multiplicative and additive terms) are randomly initialized.

For training, the minibatches are built by grouping utterances with similar length for efficiency. The model is updated every $8.2M$ speech frames with Adam algorithm [133] with weight decay 0.001. In order to speed up training, our implementation utilized Flash Attention [47] to accelerate training, together with CUDA-level optimized modules [162].

EWC control It is also important to discuss about the hyper parameter λ in EWC. This parameter has been proven to be important since a high value restricts the model to learn from a new dataset, while a small value prevents the model from retaining the previously learned knowledge [141] and several methods have been proposed to relax this rigidity [156], however to a minimal effect. In our work, we found that a value ranging from 0.001 to 0.004 is stable for the first session, however subsequent sessions requires values of a magnitude higher, from 0.01 to 0.05 to prevent catastrophic forgetting, possibly due to the difference between languages. In order to stablize training, this coefficient is *gradually decreased* during training, first the model is forced to learn with very rigid weights (with high coefficient) which are loosened over time by decreasing the coefficient. The initial coefficient is often 0.001 while the final coefficient is $0.1\times$ the initial. Averaging checkpoints [255, 189] between the best 5 checkpoints before decaying the coefficient and the best 5 checkpoints at the end of

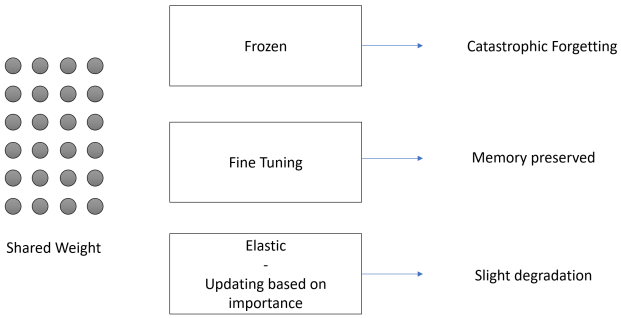


Figure 5.4: Three different scenarios for the main weights in the ASR model in continual learning.

training is found to obtain the best performance for the old and new languages.

Baselines To the best of our knowledge, this is the first attempt at learning languages incrementally. The main objective of our study is to propose a method that prevents catastrophic forgetting without a compromise on the ability to learn new languages. As such, we measure how much the model can memorize the previously learned languages after learning new ones, can compare the WER with the previous session. For the new languages, we setup three baselines:

- EWC: the model learns incrementally based on EWC.
- Weight factorization: the model carries on the weights to the next session, in which the main weights are frozen and only newly added weights are learned. These models can be considered to be the EWC with $\lambda = \infty$
- The scenario in which all weights are fine-tuned is expected to lead to catastrophic forgetting is also considered, because the performance of the new languages is expected to be the ceiling for continually learned models with regularization.

These scenarios are depicted in Figure 5.4 with three different outcomes.

Learning without 'catastrophic' forgetting The experiments show different pictures from EWC and Factorization as illustrated in Table 5.1. The main purpose of this table is to measure the short term impact of the methods, including the Factorized Weights, EWC and when all languages are present at once. As expected, the model with factorization and frozen body weights (FFr) can keep the performance of the original languages intact. However the EWC model performed extremely poor. Many languages exhibited fully catastrophic forgetting, such as (ta) or (th) when their error rates nearly reach 100%[2].

Without any regularization or weight factorization, the models trained on new languages quickly overfit to the new ones while reaching more than 100% WER for the prior languages. Having EWC can only slow down the catastrophic forgetting process and made the prior error rates 6.5 times faster, while also make learning new languages worse. Meanwhile, weight factorization with frozen shared weights, by nature, totally prevents catastrophic forgetting, so the prior error is almost the same with having all languages (the last column).

The most flexible combination between EWC and Weight Factorization shows the most interesting result. It provided a similar performance for new languages compared to fine-tuning the whole model (13.6% compared to 13.8%) and is the closest configuration compared to the Oracle setting with all languages in presence (13.1%). The performance loss in the previously learned languages is worse than fine-tuning the whole model (7.7%), however it is far away from catastrophic forgetting and stay at a reasonable level (7.8%). It can be seen that, EWC is important by regularizing the gradients for the important weights (that are measured by the variance of gradients). The regularization, however, is still too weak to prevent catastrophic forgetting. This could be explained by the residual structure in the Transformer, in which a small disturbance in the network can cause a large change in the output [155].

[2] This is also what happened with a normal model with finetuning without either EWC or weight factorization.

When using weight factorization, the stability of each network branch (whose weights are the linear combination of the shared weights and the factorized weights) is improved, as shown in the effect of EWC.

Continually learning in Multiple iterations Given the results in 1-iteration learning that demonstrated that it is possible to keep a competitive performance for the previously learned languages while effectively learn the new languages, we proceeded to extend the experiments to multiple iterations. Our observation was that forgetting still happens, despite not severely. As a result, it is possible that the degree of forgetting is higher with more iterations, also this is theoretically motivated which will be explained in the next section.

As can be seen in Table 5.2, the performance for the first set (10 languages) increases over time for the *EWCF* model and after 3 iterations, it is 52.5% higher than the original one, as kept by the *Frozen (FFr)* model, a significant amount but partly because the original error rate is quite low.

For the second set (nl, ar, zh, uk, cs) the error rate of the EWCF model is still lower than the FFr after one lifelong learning iteration (from **Iter 1** to **Iter 2**). Two iterations is required for the Frozen model to surpass this, naturally because one model is inevitably getting worse after each iteration, while the other does not forget despite a worse starting point. This story is repeated in the other two sets of languages.

5.5 Analysis

Why does EWC lose effect over time? The experiments showed that a combination of external weights per language and EWC for the shared weights is effective in the first two iterations, when the performance can strike the best spot for the compromise between forgetting and learning. Yet we can observe two problems: Forgetting is increased over time, and the role of weight factorization is much more important than EWC, which is not effective by itself.

Table 5.1: Performance(WER↓) on the CommonVoice-Europarl testsets. The models included are EWC (E), EWC combined with Factorization (E+F), Factorized with main weights frozen (FFr) and Factorized with all weights finetuned (FFt). Models are first trained on 10 languages (top) and then learned on 17 languages (bottom).

Lg	Hrs	**E**	**E+F**	**F**	**FFt**	All
(de)	1050	53.3	7.5	7.2	39.3	7.2
(fr)	800	19.7	11.6	11.4	37.4	11.2
(es)	400	14.6	7.1	6.8	19.9	6.2
(it)	325	26	7.3	6.8	24	6.5
(fa)	293	80	4.1	3.7	36.2	4
(ta)	198	91.4	20.5	18.2	44.9	21
(pt)	120	37	7.6	7	23.1	6
(ru)	148	40.5	6.4	5.3	26.7	5.4
(pl)	145	40.8	8.4	7.7	30.9	7.6
(th)	133	94.4	3.5	3.2	15.5	3.2
Avg		49.8	8.4	**7.7**	29.8	7.8
(nl)	150	10.6	7.3	7.6	7.3	6.8
(ar)	85	24.8	19	17.7	17.3	18.2
(zh)	63	23.6	14.7	15.8	14.7	14.7
(uk)	56	21.7	8.1	8.7	7.8	7.4
(cs)	49	19.8	9.2	9.6	9.4	8.4
(ro)	45	28	11.2	11.7	11.4	10.1
(sv)	35	28.3	12.4	13.2	12.5	12.3
(et)	32	31.5	13	14.3	13.6	13.2
(tr)	30	19.6	7.8	8.3	7.4	8.4
(ja)	26	20.5	7.5	8.5	7.4	7.9
(id)	23	17.6	6.8	6.9	6.7	6.7
(lt)	16	56.1	26.9	27.7	27.4	25.5
(mn)	12	51.2	24.5	25.7	25.1	24.3
(sl)	9	41.3	10.1	10.2	10.5	9.1
(hi)	8	70.5	27.9	28.9	28.5	27.6
(gl)	7	16.7	11.7	10.7	14.5	9.8
Avg		30.1	**13.6**	15	13.8	13.1

From the theoretical analysis [109], EWC originates from replacing the log posterior $\log p(\theta|T_1)$ in Equation 5.3 with its Taylor expansion form. The requirement for this expansion is that we need to find the optimal value θ^* during optimizing the model for the data T_1, such that $\log p(\theta|T_1)$ is 0 at θ^*. Factually, θ^* is found using stochastic gradient descent which does not guarantee this property and this problem is exacerbated by regularization methods such as Dropout [244] encouraging the model to not fit the data perfectly for the sake of generalisation.

The approximation is further "approximated" by the fact that the Hessian in Equation 5.2 is approximated by the diagonal of the Fisher Information matrix. Furthermore, the prior is also assumed to be a zero-mean isometric Gaussian [109] which is rather a simple assumption [134]. From such approximation, it is understandable that EWC might be effective when the new task/data is somewhat close to the original task which is unlikely in language learning.

Finally, the missing ingredient of EWC is that the model and its parameters θ is treated as a black-box. The choice of the model architecture can have an effect with EWC. Transformers and their residual connection are known to have representations strictly centering the word-embeddings [155], which can make the model difficult when adapting to a new language without changing too much weight values.

Empty space in Model We extracted the Fisher information after the initial training stage, and other two iterations in the second set of experiments. For each Fisher diagonal matrix, the number of weights with high Fisher value (with a threshold of 0.5) is counted. For all layers in the network, the number of important parameters is increased over the iterations.

To illustrate this, we counted the number of parameters with their importance greater than the threshold within all shared model parameters in the self-attention and feed-forward neural network blocks in the Transformer, for both the encoder and decoder. In Figure 5.5 showing the percentage of the important weights over approximately $380M$ weights, the model could allocate approximately 15% of the weight

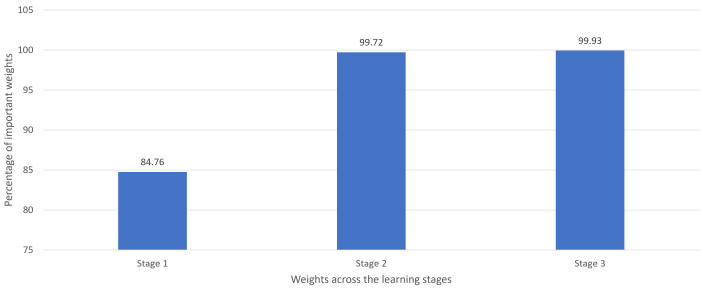


Figure 5.5: The percentage of important weights in the model over the course of learning.

space to learn from the first to the second stage. However since the second stage model is almost full, it can explain the undesirable performance in the third stage. So the question here is that: how can we allocate more weights for long term learning? Maybe it is possible to increase the size of the language dependent weights, so that the importance in the main weights are reduced. However it comes with a storage cost per language added in the system.

Related works Learning tasks consecutively without catastrophic forgetting and using the knowledge of previous tasks to facilitate learning new task is an important topic in machine learning that has been investigated in computer vision or reinforcement learning. There are three common approaches in continual learning: regularization, progressive architecture and replaying from memory. The regularization approach is model agnostic and focuses on designing objective functions that punish weights that tend to be shifted too far from the original positions, where the optimal state with respect to the previous tasks is achieved. The important weights can be identified by importance [135] or memory synapses [4]. Besides, the network can also be designed to to isolate the weights and module of each task, while allocating new weights for new tasks [221, 82]. It is also possible to store examples of previous tasks as memory replaying [158] to ensure that the gradient updates in the new tasks do not have negative effect over the previous datasets.

In Automatic Speech Recognition, continual learning or incremental learning has been explored in a number of monolingual scenarios. The hybrid HMM models were explored in continual learning by learning different datasets such as World Street Journal, Reverb, Librispeech and Chime4 consecutively [222]. In a similar manner, the sequence-to-sequence model can also be trained on different English datasets with the goal of evaluating the performance in each domain after training on another [32]. Recently, the replaying from memory approach has been applied to online continual learning [276] without a clear boundary within task.

Compared to the related works, continual learning new languages in multilingual ASR has a clear task separation due to the difference between languages, compared to monolingual setups. The weight factorization method can be classified into the architectural approach, by assigning new network parameter for new 5 languages. In our work, we combine both architectural and regularization approaches to cover forward and backward transfers in the desiderata.

The closest work related to learning new languages so far is to apply gradient episodic memory [158] in online continual learning [276]. Even though learning on new data of the same language is different than learning new languages, however the method can be conceptually applied for the latter. In our experiments, however, compared to our approach, gradient episodic memory is less memory and computational friendly when applied for the new languages. We provide an analysis, as this method can be applied in future works.

Gradient Episodic Memory As we mentioned before, the gradients obtained while learning on new tasks/languages drive the model parameters far away from the optimal states in the original states. In order to alleviate this problem, Gradient Episodic Memory (GEM) - a replay-based continual learning method aims at minimizing the distance between the gradients of the parameters in the new data and the old data.

$$\begin{aligned} &\min_{\omega} \frac{1}{2}||\omega - \mathbf{g}||_2^2 \\ \text{s.t. } &\langle \omega, \tilde{\mathbf{g}_i} \rangle \leq 0, \forall i \in (0, 1, \ldots, n-1). \end{aligned} \tag{5.11}$$

In Equation 5.11, $\mathbf{g}, \mathbf{g}_i \in \mathbb{R}^{|\Theta|}$ with $|\Theta|$ being the number of (trainable) parameters in the model. n is the number of previous tasks. $\mathbf{g}$ is the gradient of the parameters obtained from the new data (current task) and $\tilde{\mathbf{g}_i}$ is the gradient for the data from the ith task in the memory.

Solving Equation 5.11 - a quadratic programming problem - then is changed into solving its dual form:

$$\begin{aligned} &\min_{v} \frac{1}{2} v^\top \mathbf{G}\mathbf{G}^\top v + \mathbf{g}^\top \mathbf{G}^\top v \\ &\text{s.t. } v \leq 0. \end{aligned} \tag{5.12}$$

where $\mathbf{G} = (\tilde{\mathbf{g}_1}, \tilde{\mathbf{g}_2}, \ldots, \tilde{\mathbf{g}_{n-1}})$ and $v \in \mathbb{R}^{n-1}$. The solution of the primal form is recovered as $\omega = \mathbf{G}^\top v + \mathbf{g}$. In other words, the gradients required to update the model with stochastic gradient descent are a linear combination between the gradients in the previous tasks and the current gradients. The former is obtained my maintaining a subset of the previous data used as memory.

In our experiments, this approach is computational unfriendly for the following reasons:

- First, it is required to store a part of the training data in previously trained languages. The amount of data required to store subjects to the significance of the past gradients $\mathbf{G}$. Since this is the approximation of the data distribution of the previous languages, the optimal solution would be storing the whole previous training data and sampling mini-batches per forward-backward passes in the current training.
- Therefore, each training cycle in the current task/language would require a new estimation of $\mathbf{G}$ due to the fact that model has been changed previously. Previous works in GEM did not mention about a cheap estimation such as using only the initial

model, and the assumption that GEM can bring positive backward transfer assumes that $\mathbf{G}$ is re-estimated for every update. The computational requirement would increase linearly with the number of tasks.

- Even when the dual quadratic programming problem manages to change the QP on $|\Theta|$ variables into $n - 1$ variables, computing the term $\mathbf{G}\mathbf{G}^\top$ is expensive since $\mathbf{G}$ can have billions of parameters, such as our models having half billion parameters.

Nevertheless, it is worth making a theoretical connection between these methods, and GEM can be feasible when computation becomes more excessive. In our experiments, EWC and factorization runs about 20 times faster than GEM on the same model. The latter is slow partly due to the memory constraint in computing $\mathbf{G}\mathbf{G}^\top$ and required to use CPUs instead of accelerated GPUs, therefore a qualitative comparison is not available in the Thesis, pending for the future works.

5.6 Conclusion

Learning new languages without forgetting falls into the intersection between automatic speech recognition and continual learning/incremental learning areas. In this thesis, we presented a model for this specific task with the combination of Elastic Weight Consolidation and Weight factorization. This model exhibits an interesting property when learning languages sequentially, that it can learn the new languages very effectively compared to other regularized baselines, while maintaining the performance for the previously learned languages to an acceptable extent. We provided an analysis to empirically explain the effectiveness (and ineffectiveness) of EWC and aim at improving the weaknesses of the model to improve further on this task. One of the missing ingredients is to use distillation from the previous models as a compression of the data that is promising, which is shown in many concurrent work in image generation [79].

Table 5.2: Continual learning with EWC and Factorization (EWCF) or Frozen Factorized (FFr) for three iterations (the first session is with the top 10 languages and the next sessions are separated by middle rules.

Lg	E+F	F	E+F	F	E+F	F
	Iter 1		**Iter 2**		**Iter 3**	
(de)	7.4	7.23	8.7	7.23	10.5	7.23
(fr)	11.5	11.4	13.1	11.4	15	11.4
(es)	6.74	6.74	8.4	6.7	9.8	6.74
(it)	7.1	6.8	9.1	6.8	11.1	6.8
(fa)	4.1	3.7	4.9	3.7	6.5	3.7
(ta)	19.7	18.2	23.3	18.2	28.5	18.2
(pt)	7.3	7	9.2	7	11	7
(ru)	6	5.3	7.4	5.3	9.6	5.3
(pl)	8.1	7.72	9.4	7.72	11.3	7.72
(th)	3.4	3.2	4	3.2	4.6	3.2
Avg	8.1	**7.7**	9.8	**7.7**	11.7	**7.7**
(nl)	7.19	8	7.7	8	9	8
(ar)	15.9	19.4	16	19.4	17.4	19.4
(zh)	14.8	17.7	15.7	17.7	16.9	17.7
(uk)	7.9	10.4	9.1	10.4	12.6	10.4
(cs)	9.3	10.6	10.3	10.6	13.9	10.6
Avg	11	13.2	11.8	13.2	14	13.2
(ro)	-	-	11.6	12	12.8	12
(sv)	-	-	12.1	14.8	14.7	14.8
(et)	-	-	12.1	16.8	14.7	16.8
(tr)	-	-	7.5	9.4	9.5	9.4
(ja)	-	-	7.5	9.5	8.3	9.5
Avg	-	-	10.2	12.5	12	12.5
(id)	-	-	-	-	7.9	8
(lt)	-	-	-	-	28.5	29.3
(mn)	-	-	-	-	27.7	28
(sl)	-	-	-	-	11	12.3
(hi)	-	-	-	-	29.7	30.8
(gl)	-	-	-	-	12.3	10.9
Avg	-	-	-	-	19.5	19.9

6 End-to-end Speech Translation

In the previous three chapters, the Thesis explored an end-to-end architecture for speech recognition which is applied in three different scenarios:

- Monolingual speech recognition with the goal of achieving competitive results
- Multilingual speech recognition being wide applied for multiple languages with a strategic architecture to benefit the most from the multilingual datasets
- A flexible system towards continually learning new languages without catastrophic forgetting.

The potentials of sequence-to-sequence models can be expanded beyond speech recognition - the output is the transcription of the acoustic input. When the end-to-end approach became staple in speech recognition, the research scene has raised the question if the same approach can be used for direct speech translation, in which the output of the model is the translation in another language, especially after the success in machine translation [247, 255, 157]. Typically, speech translation is handled by a *cascaded system* consisting of a speech recognition model and a machine translation model. In such pipeline, the acoustic input is first transcribed into the text format which is then translated. More often than not, the transcription requires another layer of processing for punctuation insertion and realignment [33, 37]. Now the objective of a sequence-to-sequence model is to learn such a direct mapping and replace the whole pipeline.

6.1 Motivation of E2E Speech Translation

There are a number of reasons explaining the interest invested in direct speech translation, using machine learning methods to learn a direct mapping between the speech signals and the translation in another language.

- From an application perspective, the low-resource languages account for the majority of the languages in the world. These languages are more likely to come with translations than transcription, since most of the world's languages exist solely in spoken form without any orthography for transcription. Phonetic transcription is an alternative option but also an expensive one. Speech translation can be especially useful in documentation of endangered languages, because most speakers are bilingual in another common language and can provide translations for the recordings in the original languages [1, 26, 27].
- When using a cascaded system to address speech translation, it is required to build a pipeline of components. The required components are speech recognition, machine translation and often a punctuation or text normalizer [34] to add punctuation to the output of speech recognition which by nature does not handle punctuation in the transcript. When passing inputs-outputs through the pipeline, the system is exposed to an inherent effect: error propagation. Each involved component is likely to be error-prone and the errors are propagated through the cascade components leading to compounding follow-up errors. In statistical modeling, an crucial principal is to delay hard decisions as long as possible [108, 168].
- Building an end-to-end model is the best way to utilize the power of the neural networks. These machine learning models can learn representations from low level features such as audio waveforms (or the shallow features such as MFCC). Learning "end-to-end" models allows the networks to optimize all their

components jointly for one single loss function. As such, a subtle change in the input might be perceived in a smooth way by the model and does not lead to a large error in the output.

Compared to the "sisters" tasks, speech translation turned out to be more challenging especially for the end-to-end approach, due to a number of reasons. In comparison to machine translation that handles transformation between two discrete sequences, the acoustic input is continuous in nature with a high amount of variability. Different factors such as speaker characteristics, channel properties, dialects or background noise feature a challenge that does not exist in text translation. While speech recognition systems have proved to overcome this challenge, they are not required to deal with the translation specific problem: the non-monotonic alignment between source and target sequences, happening when words and phrases are re-ordered as the difference in the grammars. Additional linguistic problems such as word sense disambiguation, anaphora resolution also contributed to the translation difficulties. The combinatory difficulties make speech translation a challenging task.

Lastly, a neural network would require parallel data to train a sequence transformation model like Transformers, yet the parallel data required to train speech translation is not as abundant as text-based translation data. Naturally, in order to obtain this kind of parallel data, the data collectors have to start from transcription with the translation process follows suit. While text data can be harvested in many sources such as multilingual documents, speech data is much harder to find. Lacking data for supervision is one of the difficulties in this speech translation approach. On the other hand, the cascaded approach often has the upper hand when the ASR and MT models have more data and modeling capacities to optimize. As a result, even with the "error propagation" problem, the cascaded system would result in better translation accuracy than using one single end-to-end model.

Contribution The main contribution of the thesis is to show that our solution in end-to-end speech recognition can also be effective in speech translation. We proposed two strategies that directly address the two problems mentioned above. First, in terms of **modeling choice**, the

Deep Transformer model can effectively represent the acoustic features as well as the complex alignment and mapping between the acoustic input and the translation. Second, when it comes to the problem of data sparsity, the speech translation model can "borrow" data from machine translation using knowledge distillation [96, 132, 110]. By translating the transcription of the speech data into the target language, we can obtain the desired parallel data required for training the speech translation model, whose quality would depend on the quality of the machine translation model.

With the cascaded system as the target in mind, our aim is to integrate these two ideas in order to enable end-to-end speech translation to overcome the cascaded systems. This very goal is demonstrated in the following experimental scenarios:

- First, we showed that an end-to-end speech translation can be trained with a deep Transformer network. As such, this model was able to outperform other neural architectures specifically designed for speech translation such as LSTMs or shallower Transformers [52] while still having a distance compared to a powerful cascaded system [198]. In this stage, we observed the **segmentation error** existing in the training data, and relied on Relative Position Encodings to make the models more resilient.
- In order to further verify this observation, we conducted further experiments in speech translation scenarios in which *segmentation is given* so that we can rule out this problem. In this condition, the end-to-end Transformer is able to overcome the cascade, however with the large contribution from the *pseudo-labels* from machine translation.
- With that observation, we came back to the large scale scenario with mandatory segmentation and largely improved the end-to-end model using data augmentation, transfer learning and neural-based segmentation. This integration allowed the end-to-end model to even overcome the powerful system also receiving the same treatment (i.e transfer learning applied to speech recognition and machine translation).

6.2 Prior works

Speech is foundational to human communication and it is the main communication channel in many situations without any intervention in the written level. Normal dialogs or lectures are examples of such situation in which the access to information is done only via the speech channel without any textual presence. With the high potentials for application, speech translation receives a considerable amount of research interest in the recent years.

The neural sequence-to-sequence models [124, 247, 15] were quickly adapted in machine translation [160, 118, 119] and speech recognition [40, 16] showing the great potential to completely dominate the previous approach. Meanwhile, the speech translation applications still prefer the cascaded approach due to not only the fundamental reasons mentioned above, but also because the speech recognition and text translation components in the cascade are reliably constructed and transparent, i.e it is also possible to observe the transcription output which can be post-editted to correct the translation output.

Applying the sequence-to-sequence model with the LSTMs as the core units for direct speech translation quickly showed that this is an arduous task. Using a relatively simple dataset of trivial conversations in English and Spanish, the LSTM-based model cannot operate on word level and can only yield meaningful alignment between the acoustic input and the phoneme-level units [55]. An attempt to use the same structure yet to overcome the learning difficulty included the transcription in the architecture, leading to a *two stage model* [127, 254] that performs inference in two steps: first transcribing into the original language and then translating into the target language and thus mimicking a cascaded system in one neural network and eventually re-introducing the error propagation problem. By continuously connecting the hidden states of the transcript representation and the translation representation via an attention-passing mechanism [242], the resulting model not only obtained higher translation quality but is also more *data efficient* than the cascade.

Transformers materialize the sequence-to-sequence concept using attention as the main neural connectors in the network. Such a model

has less inductive bias and is more suitable than LSTMs to learn complicated mapping functions given enough training data [115]. Adapting the Transformer for direct speech translation for lecture-style talks [216] has shown an initial potential, by greatly outperforming the LSTMs [52].

In the prior works, it is notable that the successfully trained models would require assistance from the transcription via an ASR task. As such, the model is designed to be multi-tasked, with two output layers to learn to transcribe and translate at the same time. In some situations, an additional text encoder can be included for the machine translation sub-task. In contrary, the next sections would demonstrate that it is possible to keep the original sequence-to-sequence model with a single stage instead of a complicated two-staged hierarchy. In turn, the training process is also simplified without multi-tasking. Simplification is advantageous because the training process can be applied for new models and it is often industrially faster and more efficient. Through out the chain of experiments in the next three sections, our models are incrementally improved but do not deviate from this objective.

6.3 Transformers for E2E Speech Translation

With the goal of using a single-stage model (directly transforming the acoustic input to the translation), we applied the deep stochastic Transformer architecture introduced in Chapter 4 which has been shown to be capable of learning the complex mapping between acoustic inputs and textual output together with the alignment between the sequences via attention [31].

Training Approach One of the main problems with speech translation is the learning difficulty, which has been treated by using the transcription and guide the acoustic encoder of the model via learning to transcribe. Multi-task learning, however, introduces different problems in practice. The encoder would receive gradients from two

loss functions: one for ASR optimization and one for ST optimization. Despite the fact that the ASR gradient can be useful for ST learning, they are still fundamentally different (possibly the ST model requires more information in the encoder than the ASR model), so it is still necessary to carefully control the gradient flow during the learning process. Moreover, since most implementations use two decoders for ASR and ST respectively [127, 243] training such a model can be practically slow because the model has to perceive each task differently. This is ironic because end-to-end speech translation initially started from using the same model with speech recognition, which is what we are doing in this Thesis.

In order to make the training process simple, the model is first trained with the ASR task using the same process as in Chapter 3. After convergence, the decoder is discarded and then the encoder parameters are used to initialize the counterpart in the next training phase with the speech translation data. At first sight, this approach might be the same with multi-task training due to the fact that there are two decoders involved with the transcription in the model. In practice, this is much easier to implement because it is not necessary to change the model architecture as well as the training procedure for multi-task training. Moreover, since the objective of the whole training process is translation, the quality of the by-product ASR is not important.

Relative Attention Here, we would like to mention that one of the crucial components in the network is the absolute positional encoding that assigns a fixed vector for each position in the sequence. The intention of this positional encoding is to allow the self-attention mechanism to differentiate between positions, since content-based information retrieval [74] - fundamentally attention, is position agnostic. The presence of the relative position encoding scheme as described in Chapter 3.3 would help the model to have the advantages of relative position encoding applied for the Deep Transformer [199], especially on the encoder side. The resulting novel model maintains the trigonometric position encodings to better scale with longer speech sequences, and is able to model bidirectional positions as well.

Experimental Setup The speech translation task is divided into two different subtasks. Many SLT datasets require an auto-segmentation component to splits the audio into sentence-like segments.[1] Even though end-to-end models are not theoretically limited in generating beyond sentences, longer sequences would pose more troubles in terms of memory complexity and training time. Due to the fact that attention requires to store an attention matrix A containing the scoring values of each pair of positions, the space complexity is $O(n^2)$ with respect to sequence length [136] and would require exotic strategies to approximate the values in A such as sparsity [286] or Orthogonal Random features [39]. Moreover, learning to align in long sequences is more difficult than in shorter sequences.

Therefore, for end-to-end models, the quality of this segmentation process is crucially important due to the lack of incremental decoding and higher GPU memory requirements. This problem leads to a mismatch during training and evaluation. During training. speech translation corpora such as *MuST-C* [53] contain segmentations for both the training and test sets, requiring no extra segmentation component, and so we use its English-German pair serves as our first experimental benchmark. We further carry out experiments on the IWSLT 2019 evaluation campaign data, a superset of MuST-C, where segmentation is not given; here we can compare the effects of variable-quality segmentation on different end2end models, and also compare models to highly competitive tuned cascades. We use the MuST-C validation data for both tasks since it is necessary to use a segmented test set instead of a long talk for validation. The preprocessing step is simple, in which the target sentences are tokenized using the Moses tokenizers [138]. In order to handle out of vocabulary words, byte-pair encodings [234] with 40000 units is used to split the words. For this reason, the vocabulary size of the model is roughly 40000. Unlike speech recognition, in translation we also need to keep the punctuations in the sentences. Since punctuation might not be aligned with any audio input (possibly silence, but not necessary), it is a challenge for the decoder to learn the syntax patterns in the data.

[1] This is commonly seen in IWSLT evaluation campaigns [179].

The main characters in all experiments are the Deep Stochastic Transformer [199] and the relative version using this encoding scheme for both encoder and decoder to enhance the representation of these two components. For SLT, the models and the training process are identical to ASR, with the exception that we use 32 encoder layers and 12 decoder layers. The SLT data sequences are longer and thus need more memory. Following the curriculum learning intuition that SLT models benefit from pre-training the speech encoder with ASR [18], we first pre-trained the model for ASR with the parallel English transcripts from MuST-C, and then fine-tune the encoder weights and re-initialize the decoder for SLT. This approach enabled us to consistently train our SLT models without divergence (which may happen when the learning rate is too aggressive or the half-precision GPU mode is used).

For all models, the batch size is set to fit the models to a single GPU [2] and accumulate gradients to update every 12000 target tokens. We used the same learning rate schedule as the Transformer translation model [255] with 4096 warmup steps for the Adam [133] optimizer.

Speech Translation Results Our first SLT models were trained only on the MuST-C training data and the results are reported on the COMMON testset[3], using the provided with the segmentation of each utterance which has a corresponding translation. For each utterance, we can directly translate with the end2end model, and the final score can be obtained using standard BLEU scorers such as SacreBLEU [203] because the output and the reference are already sentence-aligned in a standardized way.

As shown in Table 6.1, our Deep Transformer baseline achieves an impressive 24.2 BLEU score compared to the ST-Transformer [52], which is a Transformer model specifically adapted for speech translation. There are two reasons for this large gap in performance. Training the Transformers on speech translation was unstable unless the encoder was pre-trained with the recognition task and the stability is even worse

[2] Titan V and Titan RTX with 12 and 24 GB respectively

[3] MuST-C is a multilingual dataset and this testset is the commonly shared utterances between the languages.

with a deep encoder, leading to a bad result reported by [52]. Our results also confirmed the discovery in the subsequent publication [18]. Moreover, speech datasets are often much smaller compared to text datasets in terms of number of words. For this reason, a strong regularization is required to obtain a good result. The stochastic mechanism in the Deep Transformer allowed a large model with 32 layers to get a strong performance.

Here, using relative position information makes self-attention more robust and effective still, as our BLEU score increases to 25.2 with a 1 BLEU point improvement coming from improving the attention mechanism. However it is notable that the test samples are pre-segmented and the dataset creation process guarantees that each segment corresponds to one grammatically reasonable sentence. For this reason,

For better performance, we also add the Speech-Translation TED corpus [4] and follow the method from [52] to add synthetic data for speech translation, where a cascaded system is used to generate translations for the TEDLIUM-3 data [94]. Our cascade system is built based on the procedure from the winning system in the 2019 IWSLT ST evaluation campaign [198].

With these additional corpora, we observe a considerable boost in translation performance (similarly observed in [52]). More importantly, the relative model further enlarges the performance gap between two models to now 1.4 BLEU points. We hypothesize that the model is able to more effectively use the additional data, with data patterns more easily captured when the model considers *relative* rather than absolute distance between speech features. More concretely, each training corpus has a different segmentation method, which leads to large variation in spoken patterns, which is difficult to capture using absolute position encodings.

To verify our hypothesis, we compare these two models and the cascaded system on the TEDTalk testsets without a provided segmentation. These talks are available as long audio files and require an external

[4] Available from the evaluation campaign at https://sites.google.com/view/iwslt-evaluation-2019/speech-translation

audio segmentation step to make translation feasible. It is important to note that the cascaded model has a separate text re-segmentation component [34] which takes ASR output and reorganizes it into logical sentences, which is a considerable advantage compared to the end2end models. We experimented with several audio segmentation methods and see that the cascade is less affected by the segmentation quality than the end-to-end models.

The results in Table 6.10 compare two different segmentation methods, LIUM [218] and VAD [271], and four different testsets. The relative Transformer unsurprisingly consistently outperforms the Transformer, regardless of segmentation. Moreover, comparing between the segmenters, the relative model more effectively uses higher segmentation quality, yielding a larger BLEU difference. While the base Transformer only increases up to 0.5 BLEU with better segmentation, this figure becomes up to 2.4 BLEU points for the relative counterpart. In the end, the cascade model still shows that heavily tuned separated components, together with an explicit text segmentation module, is an advantage over end-to-end models, but this gap is closing with more efficient architectures.

Table 6.1: ST: Translation performance in BLEU↑ on the COMMON testset (no segmentation required)

Models	BLEU
LSTM [52]	12.9
ST-Transformer [52]	18.0
+SpecAugment	19.3
+Additional Data [51]	23.0
Deep Transformer (w/ SpecAugment)	24.2
+Additional Data	29.4
Deep Relative Transformer (w/ SpecAugment)	**25.2**
+Additional Data	**30.6**

Despite the large improvement that we obtained by using Transformers and additional data, improving the translation performance by an impressive 7.6 points compared to the previous works, there is still

Table 6.2: ST: Translation performance in BLEU↑ on IWSLT testsets (re-segmentation required)

Testset →	**Relative**		**Cascade**			
Segmenter →	LIUM	VAD	LIUM	VAD	LIUM	VAD
tst2010	22.04	22.53	23.29	24.27	25.92	26.68
tst2013	25.74	26.00	27.33	28.13	27.67	28.60
tst2014	22.23	22.39	23.00	25.46	24.53	25.64
tst2015	20.20	20.77	21.00	21.82	23.55	24.95

a gap between this model and the cascaded system. Ironically, the number of parameters of the ST model is only a half of the cascaded, counting from the number of parameters of the speech recognizer and translator but just increasing the number of parameters probably is not the solution. Towards the end of the chapter, the end-to-end model would gradually shorten the bridge between the two approaches by targeting the weaknesses shown in this work.

6.4 Data augmentation for E2E Speech Translation

In the previous section, the Transformer was described as an alternative solution, in which the gap of performance between itself and the well-established cascaded system has become shortened, compared to the situation before our solution [241]. However, it is notable that direct speech translation research and benchmarks have been conducted in conditions such that there is sufficient training data for supervised learning, for example the English→Spanish Fisher dataset [204] or the English→German datasets having been built for many years [216, 53, 30]. For many language pairs, it is often difficult to gather data at a similar level compared to these two datasets.

Multilingual translation has been considered as the solution for data scarcity in machine translation [85, 123]. In this setting, the training data consists of various datasets in different languages. Unlike speech

recognition in which each individual data is monolingual and training one single multilingual model might focus on the *encoder side* of the Encoder-Decoder architecture by helping it learning better representation, as reflected in different works [193, 191, 42, 209]. In translation, due to the presence of all languages in both sides of the architecture, it is possible to establish a connection between two languages unseen in the dataset. The so-called zero-shot translation ability is enabled because the encoder representation between languages can bear certain level of similarities, and especially they can be trained to minimize the difference [194, 10].

Can the same method be applied for speech translation? In such organization, the language organization graph stays the same with the translation setting, only the modality of the source side is changed to speech data. How does an end-to-end system perform compared to a fully fledged cascaded trained on a multilingual setting (ASR + MT) and especially on zero-shot translation? If the performance of the end-to-end system is subpar, can we fill the gap in the dataset using pseudo-labels? In order to answer those questions, we trained speech translation systems using both approaches on the Multilingual TEDX dataset [223].

As to provide the answers for such research questions, we discovered that the quality of the end-to-end translation model depends heavily on the sister tasks: ASR and MT. Specifically, the ASR model is required to bootstrap the encoder of the end-to-end model, while the MT model can be used to translate the transcripts of the speech data into different languages to be used as pseudo-labels for the training the end-to-end model. Contextually, various benchmarks have shown a fierce competition between traditional cascade systems and end-to-end counterparts [116, 117, 7]. The competition without a doubt would continue in multilingual speech translation especially in a low-resource condition. However, the competition between two modeling schemes suggests that each of them possesses its own strengths and advantages. Notably the cascade models can easily benefit from the separated optimized architectures of each sub-task and enjoy the larger available datasets, while the end-to-end models can theoretically avoid *error propagation*. In this Thesis, we can also show that it is possible

to combine via *ensembling* the outputs of end-to-end and cascaded systems to maximally utilize the strengths of each approach. The work has been conducted under the scope of the IWSLT2021 evaluation campaign [5].

Dataset overview The Multilingual TEDx corpus [223] provided us with the 5 languages Spanish (es), French (fr), Italian (it), Portuguese (pt) and English (en). While speech audio is available for the first 4 languages, text translation is available for all 20 language pairs, and the speech translation parallel data is largely more scarce than the other two. The data statistics is shown in Table 6.3 and 6.4.

Source → Target	**en**	**es**	**fr**	**it**	**pt**
es	36K	102K	3.6K	5.6K	21K
fr	30K	20K	116K	-	-
it	-	-	-	50K	-
pt	-	30K	-	-	90K

Table 6.3: Data statistics for speech recognition/translation in the number of utterances.

Source → Target	**en**	**es**	**fr**	**it**	**pt**
en	-	36.2K	30.5K	-	30.8K
es	36.2K	-	24.4K	5.6K	21.1K
fr	30.1K	24.4K	-	-	13.2K
it	-	5.6K	-	-	-
pt	30.8K	21.1K	13.2K	-	

Table 6.4: Data statistics for machine translation in the number of sentence pairs.

It is noticeable that the training data is severely lacking for speech translation when the number of sentences is only a fraction of the ASR or MT resources. As a result, our initial plan was to generate synthetic translations from the available transcripts, that can effectively increase the data size for training end-to-end SLT models.

Enhancement for Transformers Due to the need of building all three systems, the Transformers are going to be the protagonists in all tasks: machine translation, speech recognition and speech translation. In this specific works, we experimented with several model improvements for the Tranformers. As we know, they are constructed with blocks of transformation functions including self-attention and feed-forward neural networks.

Self-attention transforms a sequence of states using themselves as queries, keys and values, building up hierarchical representational powers since the output states are the weighted-sum of the input states that can be flexibly learned during training. Relative attention [236] further improves the interaction between states by assigning learnable weights for each relative position. [197] incorporated this mechanism into speech models by extending the partially learnable relative positions in [46] to attend to all positions in the sequence bidirectionally.

Furthermore, the Transformer models are strengthened by using dual feed-forward (FFN) layers per block instead of one [159]. As such, one feed-forward network block precedes the initial self-attention in either encoder and decoder. The outputs of both FFN layers are scaled by 0.5. Besides, it is possible to help training deep Transformer better by using RELU-inspired activation functions that do not suffer from dead neurons. GELU [93] and SiLU [58] are combined with gated linear units [48], as used in our activation functions.

In most of our experiments and in the eventual submission, all of the above enhancements were incorporated. Ablation studies are unfortunately not fully possible because of the time constraint, but will be provided to depict the improvement of each addition.

Speech Recognition Our speech recognition models are built based on both the LSTM and the Speech Deep Transformer [198] enhanced with bidirectional relative attention [197]. While LSTM models have been intensively experimented for the best results [176, 183], Transformers have been recently adopted to this task with strong results [198, 197].

For the four languages in the Multilingual TEDx, we trained both multilingual Transformers and LSTM models on the combination of the datasets, using the factorization scheme. The LSTM has 6 encoder layers and 2 decoder layers with 1024 hidden units in each layer. The sole attention layer between encoder and decoder is an 8-head dot-product attention. On the other hand, we experimented the Transformers with the "Large" models having 16 encoder layers and 6 decoder layers with 1024 units in the hidden layers.

The models are trained with Adam and an inverse square-root learning rate schedules with 4096 warm-up steps following the same setting as [255] for up-to 120K steps or early-stopping on the development set. In order to facilitate training, layers are randomly dropped with the highest rate of 0.5 and linearly reducing from top to bottom [198]. Due to the relatively small size of the dataset, regularization is added with dropout probability 0.35 in all layers, and spec augmentation with dropped frequency range is $F = 16$ and the maximum dropped time $T = 64$ which is relatively aggressive.

Language	LSTM	bTF	eTF	Ensemble
es	16.9	16.4	15.25	14.37
fr	16.5	16.8	15.39	14.44
pt	18.3	19.5	17.1	16.79
it	19.5	16.4	17.24	15.47

Table 6.5: Comparison on Multilingual TEDx dataset (WER↓). Our baseline models include the baseline (b) and enhanced (e) Transformers (TF) and the LSTM.

Table 6.5 shows the experimental result of speech recognition, in which we can see that the Transformer with only Relative attention is as good as the LSTM, while using all enhancements allowed us to improve the result further. It is notable that those results are obtained using our own word error rate measuring method that does not remove punctuations, which are retained in ASR to be compatible with the subsequent MT models.

Removing the punctuations and using the evaluation scripts in the same repo with [223] gave us 11.0, 13.88, 13.38 and 14.14 error rates

for Spanish, Italian, French and Portuguese respectively, which are significantly lower than the Hybrid LF-MMI provided.

Machine Translation Our multilingual machine translation is built based on the universal multilingual framework [84, 122, 194], in which the vocabulary is shared between languages using a BPE size of 16000 merging units.

Thanks to the relatively small data size, the translation task was used to measure the incremental improvement of various features, including the relative attention and the Macaron feed-forward layers. Therefore, experiments were carried out using the base-setting of Transformer as the starting point. Dropout was increased to 0.35 together with word dropout [66] at both encoder and decoder to help the models counter overfitting. The output language is controlled by the language embedding vectors added directly to the word embedding at every timestep [83, 194]. The language pairs are randomly sampled based on the training size of each pair (no temperature was used). Training is done using the adaptive learning rate for Adam, with maximum learning rate at 0.7 achieved after 4096 warming-up steps, and is often early-stopped after 60000 training steps, each is approximately 48000 words.

Regularization is further improved via data diversification [177]. Carrying a similar idea of back-translation [233] that generates synthetic labels for untranslated monolingual data, the main idea of data diversification is to popularize the available training data with synthetic translation of both source sentences and target sentences.

According to the algorithm presented in [177], the training process is divided into rounds in which the training data is incrementally added with synthetic data coming from the refining models themselves. Starting from the original training data in round 0, we use the best settings in round n to translate the source and target sentences in the training to the counterpart language and add the synthetic translation pairs to the current training data, proceeding to round $n + 1$. Each synthetic pair consists of one original sentence and one synthetic sentence. The

idea is the combination of backtranslation, model distillation [132] and data augmentation [268] without any additional data.

Interestingly, thanks to the multilingual property, it is also possible to translate one sentence to a range of languages after each round, leading to different options and a massive amount of sentences to be added. However, it was empirically found out that the method did not scale after 1 round, and massively translating to all languages did not improve the training data. Therefore, after round 0, the best configuration which is an ensemble is used to generate synthetic parallel data for round 1 by just translating each sentence to the same language in the original dataset.

The translation result is seen in Table 6.6. We showed the progressive results as a result of adding each empirical feature, and measured the change in average over 14 language pairs. Even though the training data also contains language pairs that are not included for the SLT task, we found that adding those "reverse" language pairs is beneficial for the others.

In terms of improvement, it can be seen that even though in this extreme low-resource scenario, using more complicated architecture obtained better translations. A combination of relative attention, macaron FFN and 16 layers of depth allowed us to improve the baseline by 0.95 BLEU points, in which the relative attention seems to be the most useful. Ensembling multiple models is, as expected but costly to improve the results further.

Data diversification was very effective after the first round, by improving the average score by nearly 1 BLEU point. Italian-related language pairs enjoyed up to 2 BLEU points, due to the lowest amount of original sentences. This result somewhat went against the initial expectation, because by not changing the sampling method, the data ratio for those languages was even lower than in round 0.

We obtained the best configuration for text translation with ensembles on round 1. Proceeding to round 2 unfortunately did not produce any further improvement, which might be reasoned by the dominance of synthetic sentences in terms of quantity.

Pair/Model	TF	+Rel	+MCR	+16L	+ESB	+DSF	+ESB	+DSF2
es-en	33.48	33.98	34.94	34.93	35.16	35.88	**36.14**	35.83
en-es	30.87	31.34	31.88	31.72	32.76	33.42	**33.97**	33.56
es-fr	40.65	41.40	41.19	41.26	42.06	42.87	**43.57**	43.12
fr-es	38.48	38.59	38.98	38.85	39.87	40.82	**41.09**	40.88
es-it	28.82	29.07	30.24	31.29	31.27	32.50	**33.80**	32.93
it-es	34.74	35.27	35.25	35.31	36.58	38.41	**39.01**	38.50
es-pt	43.04	43.40	43.65	43.53	44.30	44.96	**45.40**	45.03
pt-es	46.95	47.01	46.63	46.59	47.70	48.74	**48.95**	48.41
fr-en	38.29	38.62	39.64	39.53	40.32	41.09	**41.65**	40.93
en-fr	39.88	40.47	40.85	41.18	41.51	42.40	**43.17**	42.14
fr-pt	40.61	41.31	41.71	42.52	42.50	43.94	**44.25**	43.52
pt-fr	46.14	46.42	46.57	47.02	47.76	48.90	**49.66**	48.76
fr-pt	37.67	38.49	38.73	39.81	39.57	40.23	**40.55**	39.52
pt-fr	34.60	34.53	35.07	35.43	35.58	36.59	**37.05**	36.51
avg	38.16	38.56	38.95	39.21	39.78	40.76	41.3	40.68
		+0.4	+0.29	+0.26	+0.57	+0.98	+0.54	-0.62

Table 6.6: IWSLT 2021 machine translation progressive results. The features including Relative Attention (REL), Macaron FFN (MCR), 16 layer-deep (16L), ensembling (ESB) and diversification (DSF) are additive from left to right, starting from the base model. The last row shows the improvement compared to the previous increment.

End-to-end Speech Translation Naturally, end-to-end speech translation is developed at the last stage to benefit from the previous stages. The ASR models serve as providing the SLT with the pretrained encoder, while we used the MT model to fill the gaps, i.e translate all available ASR data. This allows us to increase the amount of training data for SLT significantly, especially for languages such as Italian and French.

Architecture wise, we only used Transformers for SLT, that followed the same training procedure with ASR due to the fact that the encoders are transferred from the Transformer ASR models.

The results are shown in Table 6.7. Unfortunately the results without ASR pre-training are not available because training was unstable and likely to diverge in such harsh data condition. It is not unexpected that the end-to-end model (E2E) trained with only the initially limited amount of data falls behind the performance of the cascade models. With distillation from machine translation, the performance is largely boosted to be on par with the cascade. The 0.2 differential in average

mostly comes from Portuguese-Spanish, Italian-English and Italian-Spanish.

Compared with pre-distillation, a lot of language-pairs enjoyed a significant improvement of up to 26 BLEU points, such as the sample Italian audio inputs, thanks to the distillation models changing zero-shot to supervised settings. The supervised language pair that was mostly improved is Spanish-French (12 BLEU points).

Finally, in this particular SLT setup, we found that it is useful to ensemble cascade and SLT models in a multi-modal manner. In the literature, it has been observed that each approach has its own strength. While the components of the cascade can be easily tuned individually because ASR and MT have lower mapping complexity than SLT, the end-to-end models can avoid error-propagation that plagues cascade systems. An ensemble suggests that we can combine the strengths of two approach, yet only available in certain experimental settings that leaves *audio segmentation* out of the scope. Here the ensemble is done by simply using the same bpe vocabulary for the MT and SLT models, and average the output probabilities of the MT and SLT models for every timestep. The result showed that this intuition can help improve the results further.

In the final results, we can see that the ensemble quality depends on the ASR performance, which can be seen in test sets with Spanish audio and French audio. At the relatively low error rate, combining two approaches provides a significant boost to the translation quality. However, for French samples the deterioration of the cascade makes the combination worse than the sole end-to-end solution. This experiment shows that error propagation is a serious problem and end-to-end SLT systems can be more robust than cascades with sufficient data and training efficiency improvement. The evaluation also suggests us to investigate into zero-shot translation for multilingual SLT, which is extremely difficult because of the modality difference between the source and target sequences.

Eventually, our systems achieved the highest position in the evaluation campaign [5].

Model Pair	Cascade	E2E	+Syn	+ESB
es-en	30.44	25.58	30.27	**31.02**
es-fr	31.64	18.81	31.32	**32.25**
es-it	26.07	22.94	26.22	**26.21**
es-pt	39.33	34.73	39.53	**40.04**
fr-en	35.41	29.73	35.19	**36.06**
fr-es	37.71	30.13	38.48	**38.96**
fr-pt	38.21	30.98	37.97	**38.44**
pt-en	33.63	28.16	33.25	**34.15**
pt-es	37.53	25.55	38.41	**38.43**
it-en	24.28	5.37	24.92	**25.29**
it-es	32.29	7.20	33.67	**33.90**
avg	33.32	23.56	33.56	**34.06**

Table 6.7: End-to-end speech translation results on progressive testsets.

6.5 State-of-the-art integration

The experiments in the previous section have shown that, under a perfect condition in terms of segmentation, the end-to-end model can outperform the cascade, despite the fact that it desperately needs the distillation effect from the cascade model to reach the performance, especially in the zero-shot condition, as in the results in Italian→English and Italian→Spanish when none of the pairs appear in the training data. Coming back to the large condition, it has been seen that the performance gap is caused by two factors: modeling capacity in the end-to-end model and segmentation quality. The final contribution in speech translation in this Thesis is to eventually close this gap between the two approaches. These final touches are achieved by using two main factors: **transfer learning** and **data-driven data segmentor** that can provide a better segmentation result than acoustic-based segmentation tools.

To remind the difference between an end-to-end model and a cascaded system, the latter is able to divide the complicated ST to smaller

sub-problems: automatic recognition, (often) re-segmentation [34] and machine translation and have the advantage of using more data to separately optimize the components. The former, on the other hand, relies on a single network architecture that requires an explicit speech-translation dataset, potentially alleviating error propagation and having a faster generation speed (due to the reduction in the number of involved models).

Evaluation campaigns [179, 7, 5, 6] around the time of the thesis showed that the performance gap between E2E and cascade is gradually reduced, and there are three negative factors that outweigh the advantages of having a single architecture without the problem of error propagation [241].

- Data utilization: the end2end model can only be directly trained on parallel speech translation data, which is often lacking compared to speech-transcription or text translation data. Previously the SLT models would require a necessary pre-training step with ASR in order to have comparable results with cascade [18, 195].
- Modeling capacity. The transition from shallow LSTM-based models [242] to Transformer-based models [197] resulted in a big leap in model capacity and showed the potential of the E2E approach.
- Better audio segmentation. Decoding directly from long audio files is infeasible due to the expensive memory requirement and the presence of other distractions such as breaks, noise or music. Applying either cascade or E2E models absolutely requires an audio segmentation step performed by a voice activity detection system. While the cascade systems can handle imprecise cuts based on a re-segmentation process [34], the E2E lacks this ability to recover from this training-testing condition mismatch.

Here, the integrations in the thesis improved our end-to-end SLT systems for English→German with up to 6 BLEU points by directly addressing the aforementioned weaknesses:

- Pretrained acoustic [14] and language models [249] are incorporated in modeling. This allowed for transferring the knowledge during the pretraining processes which contain a massive amount of data. This effect is further enhanced when combined with the pseudo labels generated by machine translation.
- By using the pretrained models, we fully utilized the large architectures that improved the results further. More importantly, the pretrained acoustic model directly extracts features from audio waveforms which is potentially an advantage compared to the manually extracted features in the previous systems.
- The audio segmentation component is changed into a full neural-based solution combined with pretraining [253]. The new solution is not only more accurate, but also directly optimized on TED Talks giving the translation model more precise and complete segmentations compared to the generic voice activity detectors.

In contrary to a cascaded system that benefit from those factors only to a limited extent, the end-to-end speech translator received a massive gain and is able to fairly overcome the performance of the cascade.

Speech Corpora. For training and evaluation of our ASR models, we used Mozilla Common Voice v7.0 [8], Europarl [112], How2 [224], Librispeech [180], MuST-C v1 [53], MuST-C v2 [30] and Tedlium v3 [94] dataset. The data split is presented in the following table 6.8.

Modeling In order to fully utilize the pretrained acoustic and language models, the models are constructed with the encoder based on the wav2vec 2.0 [14] and the decoder based on the autoregressive language model pretrained with mBART50 [249]. The parameters are initialized based on the pre-trained paramaters.

Table 6.8: Summary of the English data-sets used for speech recognition

Corpus	Utterances	Speech data [h]
A: Training Data		
Common Voice	1225k	1667
Europarl	33k	85
How2	217k	356
Librispeech	281k	963
MuST-C v1	230k	407
MuST-C v2	251k	482
TEDLIUM	268k	482
B: Test Data		
Tedlium	1155	2.6
Librispeech	2620	5.4

wav2vec 2.0 is a Transformer encoder model which receives raw waveforms as input and generates high level representations. The architecture consists of two main components: first a convolution-based *feature extractor* downsamples long audio waveforms into features that have similar lengths with spectrograms. After that, a deep Transformer encoder uses self-attention and feed-forward neural network blocks to transform the features without further downsampling. During the self-supervised training process, the network is trained with a constrastive learning strategy [14], in which the features (after being downsampled) are randomly masked and the model learns to predict the quantized latent representation of the masked time step as well as encouraging the model to diversify the quantization codebooks by maximizing their entropies.

During the supervised learning stage, the weights in the convolutional based feature extracted can be frozen to save memory since the first layers are among the largest ones and all of the weights in the Transformer encoders are fine-tuned for the task, either speech recognition or translation. Moreover, in order to make the model more robust against the fluctuation in absolute positions when it comes to audio signals, as well as the training-testing mismatched condition happening when we have to use a segmentation model to find audio segments during testing, we added the relative position encodings [46, 197] to

alleviate this problem. Similar to the relative wav2vec model in the previous chapters, the parameters in the relative model contain the original content based attention-based parameters plus the position biases. As a result this model can be initialized with the parameters from the original wav2vec model, while the position bias parameters are randomly initialized and become data-specific. Here we used the same pretrained model with the speech recognizer, with the large architecture pretrained with $53k$ hours of unlabeled data.

mBART50 is an Encoder-Decoder Transformer-based language model. During training, instead of the typical language modeling setting of predicting the next word in the sequence, this model is trained to reconstruct a sequence from its noisy version [148] and later extended to a multilingual version [157, 249] in which the corpora from multiple languages are combined during training. mBART50 is the version that is pretrained on 50 languages.

Architecture wise, this model follows the Transformer encoder and decoder [255]. During fine-tuning, we can combine the mBART50 decoder with encoder pretrained with the wav2vec 2.0 so that each component contains the knowledge of one modality. The cross-attention layers connecting the decoder with the encoder are the parts that require extensive fine-tuning in this case, due to the modality mismatch between pretraining and finetuning.

Speech segmentation As pointed out in [253], the quality of audio segmentation has a big impact on the performance of the speech translation models, which are trained on utterances corresponding to full sentences, often manually aligned, and this rarely happens with an automatic segmentation system.

With the advantage of neural architectures and pretrained models, we follow the SHAS method [253] to train a Transformer-based audio segmentation model on the MuST-C v2 corpus. Based on the high-level audio features generated by wav2vec 2.0, the model predicts the probability of each frame belonging to an utterance or not with cross-entropy. Afterwards, given the probabilities of the frames in an audio

sequence (which are actually averaged over several rolls for more consistent accuracy), a segmentation algorithm called probabilistic DAC is used to aggressively cut the segments at the points with lowest probabilities, and then trim the segments to get probabilities higher than a set threshold.

We found this method to be much more effective than other voice activity detectors such as WebRTCVAD [271]. In the next experimental part, it will be shown that the audio segmentation quality is one of the most important factors helping the E2E system. Here we closely followed the original implementations and parameters to obtain the neural segmenter.

End-to-end Offline Speech Translation Given two new factors coming into play for the end-to-end models, namely pretrained models and audio segmentation, the models are first tested on the static test which is the tst-COMMON set from the MuST-C corpus [53] with the pre-segmented utterances and labels. This testset is available for all three languages. The whole system is tested on the IWSLT testsets without utterance boundaries and labels are only provided in paragraphs (each talk is contained in one paragraph). In this condition, only English→German tests are available.

The results on this test for all three languages are presented in Table 6.9. On English-German, overall we managed to improve the purely supervised model with Transformers [197] by 2.6 BLEU points. Using the pretrained weights from wav2vec and mBART is very effective for an additional 1.6 BLEU points, while we found that the relative attention also contributed for a 0.7 BLEU points, and training the model in the multilingual setting is also slightly better.

The final results on previous IWSLT testsets are presented in Table 6.10. First of all, the new segmentation method SHAS managed to improve the translation results of our previous year's submission by up to 4.4 BLEU points (as can be see on **tst2015** and **tst2019**). By using a stronger model with wav2vec and mBART pretrained modules, the results are vastly improved by 2.2 and 3.1 BLEU points on **tst2019** and **tst2020**. The performance is incrementally improved even further, by

Table 6.9: BLEU scores on tst-COMMON from MuST-C

Model	BLEU
English-German	
E2E 2021	30.6
wav2vec + mBART	32.2
wav2vec + rel + mBART	32.9
wav2vec + rel + mBART + multi	33.2

Table 6.10: ST: Translation performance in BLEU↑ on IWSLT testsets (re-segmentation required). Progressive results from previous evaluation campaignes with end-to-end (E2E) and cascade (CD) are provided for comparison.

Testset →	**tst2015**	**tst2019**	**tst2020**
E2E2021	22.13	20.43	23.20
CD2021	24.95	21.07	25.4
E2E2021 + SHAS	26.66	24.55	25.58
+W2V-MBART	26.64	26.31	28.66
+REL	27.27	26.58	29.11
+MULTI	27.65	26.84	29.2
+ENSEMBLE	**27.87**	**27.61**	**30.05**
CD2022	26.84	25.91	28.35

combining different techniques including relative attention, multilingual training and ensemble. Eventually, we obtain a result which is 7.8 BLEU points better than the last year's end-to-end submission.

The cascade system is also improved this year, by using the pretrained ASR, MT and better segmentation. On **tst2020**, we managed to improve the BLEU score by 3 points. However this enhancement pales against the E2E, and this is our first participation in which the E2E convincingly outperformed the Cascade system.

6.6 Conclusion

The main purpose of this chapter is to present the application of the deep Transformer networks in end-to-end speech translation. Modeling enhancement, data augmentation, transfer learning and segmentation were the main factors in improving the quality of the approach, from being far behind from cascaded speech translation, to finally overcoming it. It is notable, however, that despite the end result being a single neural network that can competitively translate from speech, the effort of training a cascaded model is inevitable, due to the requirement for weight transfer (in the acoustic encoder) or data augmentation with machine translation or even potentially with expensive audio synthesis systems. This requirement means that there is much work to do in the future to reduce the training cost for speech translation systems. Moreover, since the cascaded components (ASR and MT) can stand alone in their own applications, it is more industrially efficient to construct a multi-modal and multilingual speech translation system, in which the system can simultaneously generate into different languages.

Besides, the focus of this work is on *offline speech translation* in which the system is not time-limited in processing. In an online environment, there are further modifications to be done, either as a post-processing step [202, 178, 154] after the speech translation models are trained, or a fundamental change to the training process. In such case, the large model size of current systems remain to be a burden as well unless compression techniques or distillation can help us maintain the same performance with a smaller model.

7 Conclusion and Future Works

7.1 Achievement in the thesis

Having being developed for several decades, the statistical approach using a Hidden Markov Model structure was considered a cornerstone for speech recognition. This approach is still reliable nowadays for constructing one recognition system for one certain language, and in certain cases can still enjoy a higher performance than an end-to-end system depending on the implementation [187]. In some cases, the inductive bias that exists in the manually designed components of the pipeline can be an advantage when setting up a new system. The design of such systems, however, suffers in the application towards other languages in which linguistic resources are limited. Moreover, the rigid structure can hardly benefit from learning shared and language specific features from the languages. Developing a multilingual system is important thanks to the performance factor as well as the economic factor. There are assumptions that can be transferred between languages, and using acoustic features from high-resourced languages and help the low-resourced languages [209].

With the goal of designing a flexible and extensible learning strategy for multilingual speech recognition, the thesis explored and proposed three main contributions. First, a deep architecture with stochastic Transformer layers was discovered to be able to achieve state-of-the-art level recognition performance on conversational speech benchmarks [199]. This very architecture became the backbone of later important works that further explored speech recognition [80, 186] and unsupervised

feature learning [14] by scaling on large data and computation. Notably the architectural depth is one of the important improvements for industrial solution, such as Whisper [211]. As an extension, the architecture is also empirically proven to be able to replace complicated cascaded systems in multilingual speech translation [196, 192].

Second, a novel weight factorization strategy was devised for multilingual learning in speech recognition [193]. This weight factorization is two-staged: factorizing each weight matrix in the network into shared and language specific factors, and then factorizing the language specific factors into low-rank matrix forms. The latter is to ensure that the capacity of the shared network is larger to prioritize sharing features, while allocating language specific features to the factorized components. This method was experimentally proven in both normal supervised learning and combined with transfer learning [191]. One notable advantage here is that the method is neural network driven but not architecture dependent, since matrix multiplication is the basis of neural feature learning.

Third, the current approach in supervised learning is limited in designing a flexible language learning system. An intention to add new languages to a trained multilingual system is limited in choices. Retraining the whole network with all languages is a costly option, while fine-tuning the network with new languages only suffers from catastrophic forgetting, a phenomenon happening when the weights in the network are trained towards minimizing loss on a different data, and deviated from the optimized states for the previous data. In the thesis, weight factorization became the key to combat catastrophic forgetting in language learning, due to the ability to maintain previous knowledge in the separated network branch. This property allowed us to add languages in an existed system in a more relaxing manner. On the one hand, using weight factorization can totally prevent catastrophic forgetting for all previous languages, while having a competitive performance for the new languages. On the other hand, a flexible learning algorithm such as Elastic Weight Consolidation [135] can be combined with weight factorization to compromise the quality of the old languages, but achieving almost the same effect as fine-tuning for the

new ones. This achievement is the pioneer for *continual learning* for multilingual speech recognition.

7.2 Future Work

Speech recognition has been developed to an extent that applications for mainstream languages such as English/Chinese have been reliable. The core of such applications is often a Transformer based models trained with hundred of thousands of hours of audio data. The Factorization technique not only allowed the models to scale better in a multilingual environment, but also enabled continual learning by adding new languages to a trained system to reduce cost and complexity in training a full new system. From such contributions of this thesis, new research directions can be further pursued.

First, the batch learning method widely applied in deep learning might be too costly in practice, when new training data is available everyday and new systems demand to have access to all data at once. Therefore, lifelong learning strategies are important to help these systems incrementally learn from new data while achieving the same effect of batch learning.

Second, the continual learning research investigated in the thesis is still in the early development stage, in which catastrophic forgetting is the main concern while it is also necessary to design new learning algorithms for backward transfer - learning new languages can also potentially improve the previously learned ones.

Third, there is undiscovered information in the speech data from the perspective a pure speech recognition system. When combined with other applications such as translation, information such as gender or emotion of the speaker based on fundamental frequencies is often lost in the transcription while can potentially improve the accuracy in translation. Similarly, the information is also important for synthesizing the speech in another language, preserving the original intention in the original speech.

Finally, neural networks are going beyond the typical linear transformation coupling with activation functions. New neural architectures with different properties can potentially improve speech processing. For example, linear state space models are constructed based on the linear state equation in signal processing and can generate high quality speech [72].

Bibliography

[1] Steven Abney and Steven Bird. “The human language project: Building a universal corpus of the world’s languages”. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. 2010, pp. 88–97.

[2] Oliver Adams et al. “Massively Multilingual Adversarial Speech Recognition”. In: *Proceedings of the Conference of the NAACL: Human Language Technologies*. 2019.

[3] Roee Aharoni, Melvin Johnson, and Orhan Firat. “Massively Multilingual Neural Machine Translation”. 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 3874–3884. DOI: 10.18653/v1/N19-1388. URL: https://aclanthology.org/N19-1388.

[4] Rahaf Aljundi et al. “Memory aware synapses: Learning what (not) to forget”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 139–154.

[5] Antonios Anastasopoulos et al. “FINDINGS OF THE IWSLT 2021 EVALUATION CAMPAIGN”. In: *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*. Bangkok, Thailand (online): Association for Computational Linguistics, Aug. 2021, pp. 1–29. DOI: 10.18653/v1/2021.iwslt-1.1. URL: https://aclanthology.org/2021.iwslt-1.1.

[6] Antonios Anastasopoulos et al. “Findings of the IWSLT 2022 Evaluation Campaign”. In: *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*. Dublin, Ireland (in-person and online): Association for Computational Linguistics, May 2022, pp. 98–157. DOI: 10.18653/v1/2022.iwslt-1.10. URL: https://aclanthology.org/2022.iwslt-1.10.

[7] Ebrahim Ansari et al. “Findings of the IWSLT 2020 evaluation campaign”. In: *Proceedings of the 17th International Conference on Spoken Language Translation*. 2020, pp. 1–34.

[8] Rosana Ardila et al. “Common voice: A massively-multilingual speech corpus”. In: *arXiv preprint arXiv:1912.06670* (2019).

[9] Naveen Arivazhagan et al. “Massively multilingual neural machine translation in the wild: Findings and challenges”. In: *arXiv preprint arXiv:1907.05019* (2019).

[10] Naveen Arivazhagan et al. “The missing ingredient in zero-shot neural machine translation”. In: *arXiv preprint arXiv:1903.07091* (2019).

[11] Kartik Audhkhasi et al. “Building competitive direct acoustics-to-word models for english conversational speech recognition”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 4759–4763.

[12] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).

[13] Arun Babu et al. “Xls-r: Self-supervised cross-lingual speech representation learning at scale”. In: *arXiv preprint arXiv:2111.09296* (2021).

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

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint* (2014).

[16] Dzmitry Bahdanau et al. “End-to-end attention-based large vocabulary speech recognition”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016.

[17] Lalit Bahl et al. “Maximum mutual information estimation of hidden Markov model parameters for speech recognition”. In: *ICASSP’86. IEEE international conference on acoustics, speech, and signal processing*. Vol. 11. IEEE. 1986, pp. 49–52.

[18] Sameer Bansal et al. “Pre-training on high-resource speech recognition improves low-resource speech-to-text translation”. 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 58–68. DOI: 10.18653/v1/N19-1006. URL: https://aclanthology.org/N19-1006.

[19] Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. “Simple, scalable adaptation for neural machine translation”. In: *arXiv preprint arXiv:1909.08478* (2019).

[20] Jim Barnett et al. “Multilingual speech recognition at dragon systems”. In: *Fourth International Conference on Spoken Language Processing*. 1996.

[21] Samy Bengio et al. “Scheduled sampling for sequence prediction with recurrent neural networks”. In: *Advances in neural information processing systems* 28 (2015).

[22] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. “Advances in optimizing recurrent networks”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 8624–8628.

[23] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. “A neural probabilistic language model”. In: *Advances in neural information processing systems* 13 (2000).

[24] Yoshua Bengio et al. “A neural probabilistic language model”. In: *journal of machine learning research* 3.Feb (2003), pp. 1137–1155.

[25] Jayadev Billa et al. “Multilingual speech recognition: The 1996 byblos callhome system”. In: *Fifth European Conference on Speech Communication and Technology*. 1997.

[26] Steven Bird. “A scalable method for preserving oral literature from small languages”. In: *International Conference on Asian Digital Libraries*. Springer. 2010, pp. 5–14.

[27] Steven Bird et al. “Collecting bilingual audio in remote indigenous communities”. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 2014, pp. 1015–1024.

[28] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[29] Lukáš Burget et al. “Multilingual acoustic modeling for speech recognition based on subspace Gaussian mixture models”. In: *ICASSP*. 2010.

[30] Roldano Cattoni et al. “MuST-C: A multilingual corpus for end-to-end speech translation”. In: *Computer Speech & Language* 66 (2021), p. 101155.

[31] William Chan et al. “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.

[32] Heng-Jui Chang, Hung-yi Lee, and Lin-shan Lee. “Towards Lifelong Learning of End-to-End ASR”. In: *Proc. Interspeech 2021*. 2021, pp. 2551–2555. DOI: 10.21437/Interspeech.2021-563.

[33] Eunah Cho, Jan Niehues, and Alex Waibel. “NMT-Based Segmentation and Punctuation Insertion for Real-Time Spoken Language Translation.” In: *Interspeech*. 2017, pp. 2645–2649.

[34] Eunah Cho, Jan Niehues, and Alex Waibel. “NMT-Based Segmentation and Punctuation Insertion for Real-Time Spoken Language Translation.” In: *INTERSPEECH*. 2017.

[35] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[36] Kyunghyun Cho et al. “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches”. In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://aclanthology.org/W14-4012.

[37] Kyunghyun Cho et al. “On the properties of neural machine translation: Encoder-decoder approaches”. In: *arXiv preprint arXiv:1409.1259* (2014).

[38] Krzysztof Choromanski et al. “Rethinking attention with performers”. In: *arXiv preprint arXiv:2009.14794* (2020).

[39] Krzysztof Marcin Choromanski et al. “Rethinking Attention with Performers”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=Ua6zuk0WRH.

[40] Jan K Chorowski et al. “Attention-based models for speech recognition”. In: *Advances in neural information processing systems* 28 (2015).

[41] P Cohen et al. “Towards a universal speech recognizer for multiple languages”. In: *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*. IEEE. 1997, pp. 591–598.

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

[43] Alexis Conneau et al. “What you can cram into a single vector: Probing sentence embeddings for linguistic properties”. In: *arXiv preprint arXiv:1805.01070* (2018).

[44] Marta R Costa-jussà et al. “No language left behind: Scaling human-centered machine translation”. In: *arXiv preprint arXiv:2207.04672* (2022).

[45] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. “Improving deep neural networks for LVCSR using rectified linear units and dropout”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 8609–8613.

[46] Zihang Dai et al. “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[47] Tri Dao et al. “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness”. In: *arXiv preprint arXiv:2205.14135* (2022).

[48] Yann N Dauphin et al. “Language modeling with gated convolutional networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 933–941.

[49] Jacob Devlin et al. “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.

[50] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[51] M Di Gangi et al. “Data augmentation for end-to-end speech translation: FBK@IWSLT’19”. In: *Proceedings of the 16th International Workshop on Spoken Language Translation (IWSLT)*. 2019.

[52] Mattia A Di Gangi, Matteo Negri, and Marco Turchi. “Adapting transformer to end-to-end spoken language translation”. In: *INTERSPEECH 2019*. 2019.

[53] Mattia A. Di Gangi et al. “MuST-C: a Multilingual Speech Translation Corpus”. In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. June 2019.

[54] Christian Dugast, Xavier Aubert, and Reinhard Kneser. “The Philips large-vocabulary recognition system for American English, French, and German”. In: *Fourth European Conference on Speech Communication and Technology*. 1995.

[55] Long Duong et al. “An attentional model for speech translation without transcription”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 949–959.

[56] Nadir Durrani et al. “Edinburgh’s phrase-based machine translation systems for WMT-14”. In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. 2014, pp. 97–104.

[57] Michael Dusenberry et al. “Efficient and scalable bayesian neural nets with rank-1 factors”. In: *International conference on machine learning*. PMLR. 2020, pp. 2782–2792.

[58] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning”. In: *Neural Networks* 107 (2018), pp. 3–11.

[59] Jeffrey L Elman. “Finding structure in time”. In: *Cognitive science* 14.2 (1990), pp. 179–211.

[60] Jeffrey L Elman and David Zipser. “Learning the hidden structure of speech”. In: *The Journal of the Acoustical Society of America* 83.4 (1988), pp. 1615–1626.

[61] Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. “Multi-way, multilingual neural machine translation with a shared attention mechanism”. In: *arXiv preprint arXiv:1601.01073* (2016).

[62] Robert M French. “Catastrophic forgetting in connectionist networks”. In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[63] Jürgen Fritsch, Michael Finke, and Alex Waibel. “Adaptively growing hierarchical mixtures of experts”. In: *Advances in Neural Informfaceation Processing Systems* 9 (1996).

[64] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[65] Yarin Gal and Zoubin Ghahramani. “A theoretically grounded application of dropout in recurrent neural networks”. In: *Advances in neural information processing systems*. 2016, pp. 1019–1027.

[66] Yarin Gal and Zoubin Ghahramani. “Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference”. In: *4th International Conference on Learning Representations (ICLR) workshop track*. 2016.

[67] Jonas Gehring et al. “A convolutional encoder model for neural machine translation”. In: *arXiv preprint arXiv:1611.02344* (2016).

[68] Jonas Gehring et al. “Convolutional sequence to sequence learning”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017.

[69] Jonas Gehring et al. “Extracting deep bottleneck features using stacked auto-encoders”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 3377–3381.

[70] James Glass et al. “Multilingual spoken-language understanding in the MIT Voyager system”. In: *Speech communication* 17.1-2 (1995), pp. 1–18.

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

[72] Karan Goel et al. “It’s Raw! Audio Generation with State-Space Models”. In: *arXiv preprint arXiv:2202.09729* (2022).

[73] Alex Graves and Jürgen Schmidhuber. “Framewise phoneme classification with bidirectional LSTM and other neural network architectures”. In: *Neural Networks* 18.5 (2005), pp. 602–610.

[74] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural turing machines”. In: *arXiv preprint arXiv:1410.5401* (2014).

[75] Alex Graves et al. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks”. In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 369–376.

[76] Roberto Gretter. “Euronews: a multilingual speech corpus for ASR.” In: *LREC*. 2014, pp. 2635–2638.

[77] Frantisek Grezl and Petr Fousek. “Optimizing bottle-neck features for LVCSR”. In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2008, pp. 4729–4732.

[78] Frantisek Grézl et al. “Probabilistic and bottle-neck features for LVCSR of meetings”. In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP’07*. Vol. 4. IEEE. 2007, pp. IV–757.

[79] Xiuye Gu et al. “Open-vocabulary Object Detection via Vision and Language Knowledge Distillation”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=lL3lnMbR4WU.

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

[81] Caglar Gulcehre et al. “On integrating a language model into neural machine translation”. In: *Computer Speech & Language* 45 (2017), pp. 137–148.

[82] Xin et al. Guo. “Continual Learning Long Short Term Memory”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 1817–1822.

[83] Thanh-Le Ha, Jan Niehues, and Alexander Waibel. “Effective Strategies in Zero-shot Neural Machine Translation”. In: *arXiv preprint arXiv:1711.07893* (2017).

[84] Thanh-Le Ha, Jan Niehues, and Alexander Waibel. “Toward Multilingual Neural Machine Translation with Universal Encoder and Decoder”. In: *Proceedings of the 13th International Workshop on Spoken Language Translation (IWSLT 2016)*. Seattle, USA, 2016.

[85] Thanh-Le Ha, Jan Niehues, and Alexander Waibel. “Toward multilingual neural machine translation with universal encoder and decoder”. In: *arXiv preprint arXiv:1611.04798* (2016).

[86] John Hampshire and Alex Waibel. “Connectionist architectures for multi-speaker phoneme recognition”. In: *Advances in neural information processing systems* 2 (1989).

[87] John B Hampshire and Alex H Waibel. “The meta-pi network: Connectionist rapid adaptation for high-performance multi-speaker phoneme recognition”. In: *International Conference on Acoustics, Speech, and Signal Processing*. IEEE. 1990, pp. 165–168.

[88] John B Hampshire II and Alex Waibel. “The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition”. In: *IEEE Computer Architecture Letters* (1992).

[89] Kyu J Han et al. “The CAPIO 2017 conversational speech recognition system”. In: *arXiv preprint arXiv:1801.00059* (2017).

[90] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[91] Kaiming He et al. “Identity mappings in deep residual networks”. In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[92] Georg Heigold et al. “Multilingual acoustic models using distributed deep neural networks”. In: *ICASSP*. 2013.

[93] Dan Hendrycks and Kevin Gimpel. “Gaussian error linear units (gelus)”. In: *arXiv preprint arXiv:1606.08415* (2016).

[94] François Hernandez et al. “TED-LIUM 3: twice as much data and corpus repartition for experiments on speaker adaptation”. In: *International Conference on Speech and Computer*. Springer. 2018, pp. 198–208.

[95] James L Hieronymus. “ASCII phonetic symbols for the world’s languages: Worldbet”. In: *Journal of the International Phonetic Association* 23 (1993), p. 72.

[96] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* 2.7 (2015).

[97] Geoffrey Hinton et al. “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. In: *IEEE Signal processing magazine* 29.6 (2012), pp. 82–97.

[98] Geoffrey E Hinton. “A practical guide to training restricted Boltzmann machines”. In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 599–619.

[99] Geoffrey E Hinton. “Training products of experts by minimizing contrastive divergence”. In: *Neural computation* 14.8 (2002), pp. 1771–1800.

[100] Geoffrey E Hinton and Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *science* 313.5786 (2006), pp. 504–507.

[101] Akira Hirose. *Complex-valued neural networks*. Vol. 400. Springer Science & Business Media, 2012.

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

[103] Takaaki Hori et al. “Advances in Joint CTC-Attention Based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM”. In: *Proc. Interspeech 2017*. 2017, pp. 949–953. DOI: 10.21437/Interspeech.2017-1296.

[104] Wenxin Hou et al. “Exploiting adapters for cross-lingual low-resource speech recognition”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2021), pp. 317–329.

[105] Neil Houlsby et al. “Parameter-efficient transfer learning for NLP”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2790–2799.

[106] Wei-Ning Hsu et al. “Hubert: Self-supervised speech representation learning by masked prediction of hidden units”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3451–3460.

[107] Gao Huang et al. “Deep networks with stochastic depth”. In: *European conference on computer vision*. Springer. 2016, pp. 646–661.

[108] Melvyn J Hunt. “Delayed decisions in speech recognition—the case of formants”. In: *Pattern Recognition Letters* 6.2 (1987), pp. 121–137.

[109] Ferenc Huszár. “On quadratic penalties in elastic weight consolidation”. In: *arXiv preprint arXiv:1712.03847* (2017).

[110] Hirofumi Inaguma, Tatsuya Kawahara, and Shinji Watanabe. “Source and Target Bidirectional Knowledge Distillation for End-to-end Speech Translation”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, pp. 1872–1881. DOI: 10.18653/v1/2021.naacl-main.150. URL: https://aclanthology.org/2021.naacl-main.150.

[111] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[112] Javier Iranzo-Sánchez et al. “Europarl-ST: A multilingual corpus for speech translation of parliamentary debates”. In: *ICASSP*. 2020.

[113] Kazuki Irie et al. “Language modeling with deep transformers”. In: *arXiv preprint arXiv:1905.04226* (2019).

[114] Kazuki Irie et al. “Model unit exploration for sequence-to-sequence speech recognition”. In: *arXiv preprint arXiv:1902.01955* (2019).

[115] Andrew Jaegle et al. “Perceiver: General perception with iterative attention”. In: *International conference on machine learning*. PMLR. 2021, pp. 4651–4664.

[116] Niehues Jan et al. “The IWSLT 2018 evaluation campaign”. In: *15th International Workshop on Spoken Language Translation 2018*. 2018.

[117] Niehues Jan et al. “The IWSLT 2019 evaluation campaign”. In: *16th International Workshop on Spoken Language Translation 2019*. 2019.

[118] Sébastien Jean et al. “Montreal neural machine translation systems for WMT’15”. In: *Proceedings of the tenth workshop on statistical machine translation*. 2015, pp. 134–140.

[119] Sébastien Jean et al. “On Using Very Large Target Vocabulary for Neural Machine Translation”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 1–10. DOI: 10.3115/v1/P15-1001. URL: https://aclanthology.org/P15-1001.

[120] Dongwei Jiang et al. “Improving transformer-based speech recognition using unsupervised pre-training”. In: *arXiv preprint arXiv:1910.09932* (2019).

[121] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. “Densecap: Fully convolutional localization networks for dense captioning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4565–4574.

[122] M. Johnson et al. “Google’s Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation”. In: *CoRR* abs/1611.04558 (2016).

[123] Melvin Johnson et al. “Google’s multilingual neural machine translation system: Enabling zero-shot translation”. In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 339–351.

[124] Nal Kalchbrenner and Phil Blunsom. “Recurrent continuous translation models”. In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1700–1709.

[125] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. “A convolutional neural network for modelling sentences”. In: *arXiv preprint arXiv:1404.2188* (2014).

[126] Anjuli Kannan et al. “Large-scale multilingual speech recognition with a streaming end-to-end model”. In: *arXiv preprint arXiv:1909.05330* (2019).

[127] Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. “Structured-Based Curriculum Learning for End-to-End English-Japanese Speech Translation”. In: *Proc. Interspeech 2017*. 2017, pp. 2630–2634. DOI: 10.21437/Interspeech.2017-944.

[128] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. “Visualizing and understanding recurrent networks”. In: *arXiv preprint arXiv:1506.02078* (2015).

[129] Angelos Katharopoulos et al. “Transformers are rnns: Fast autoregressive transformers with linear attention”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5156–5165.

[130] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. “Joint CTC-attention based end-to-end speech recognition using multi-task learning”. In: *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2017, pp. 4835–4839.

[131] Suyoun Kim and Michael L Seltzer. “Towards language-universal end-to-end speech recognition”. In: *ICASSP*. 2018.

[132] Yoon Kim and Alexander M Rush. “Sequence-level knowledge distillation”. In: *arXiv preprint arXiv:1606.07947* (2016).

[133] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *Proc. of ICLR* (2015). arXiv:1412.6980.

[134] Durk P Kingma et al. “Improved variational inference with inverse autoregressive flow”. In: *Advances in neural information processing systems* ().

[135] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[136] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. “Reformer: The Efficient Transformer”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=rkgNKkHtvB.

[137] Tom Ko et al. “Audio augmentation for speech recognition”. In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[138] Philipp Koehn et al. “Moses: Open source toolkit for statistical machine translation”. In: *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*. 2007, pp. 177–180.

[139] Alexander Kolesnikov et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: 2021.

[140] Jan Koutnik et al. “A clockwork rnn”. In: *International Conference on Machine Learning*. PMLR. 2014, pp. 1863–1871.

[141] Alexey Kutalev and Alisa Lapina. “Stabilizing Elastic Weight Consolidation method in practical ML tasks and using weight importances for neural network pruning”. In: *arXiv preprint arXiv:2109.10021* (2021).

[142] Peter Ladefoged and Morris Halle. “Some major features of the International Phonetic Alphabet”. In: *Language* 64.3 (1988), pp. 577–582.

[143] Lori Lamel, Martine Adda-Decker, and Jean-Luc Gauvain. “Issues in Large Vocabulary, Multilingual Speech Recognition.” In: *EUROSPEECH*. 1995.

[144] Hugo Larochelle et al. “Exploring strategies for training deep neural networks.” In: *Journal of machine learning research* 10.1 (2009).

[145] Hai-Son Le et al. “Structured output layer neural network language model”. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2011, pp. 5524–5527.

[146] Hang Le et al. “Lightweight Adapter Tuning for Multilingual Speech Translation”. In: *arXiv preprint arXiv:2106.01463* (2021).

[147] Quoc V Le. “Building high-level features using large scale unsupervised learning”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 8595–8598.

[148] Mike Lewis et al. “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension”. In: *arXiv preprint arXiv:1910.13461* (2019).

[149] Bo Li et al. “Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes”. In: *ICASSP*. 2019.

[150] Xian Li et al. “Multilingual speech translation with efficient finetuning of pretrained models”. In: *arXiv preprint arXiv:2010.12829* (2020).

[151] Xiangang Li and Xihong Wu. “Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition”. In: *2015 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE. 2015, pp. 4520–4524.

[152] Hui Lin et al. “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.

[153] Richard Lippmann. “An introduction to computing with neural nets”. In: *IEEE Assp magazine* 4.2 (1987), pp. 4–22.

[154] Danni Liu, Gerasimos Spanakis, and Jan Niehues. “Low-Latency Sequence-to-Sequence Speech Recognition and Translation by Partial Hypothesis Selection”. In: *Proc. Interspeech 2020*. 2020, pp. 3620–3624. DOI: 10.21437/Interspeech.2020-2897. URL: http://dx.doi.org/10.21437/Interspeech.2020-2897.

[155] Danni Liu et al. “Improving Zero-Shot Translation by Disentangling Positional Information”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 1259–1273. DOI: 10.18653/v1/2021.acl-long.101. URL: https://aclanthology.org/2021.acl-long.101.

[156] Liyang Liu et al. “Incdet: In defense of elastic weight consolidation for incremental object detection”. In: *IEEE transactions on neural networks and learning systems* 32.6 (2020), pp. 2306–2319.

[157] Yinhan Liu et al. “Multilingual denoising pre-training for neural machine translation”. In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 726–742.

[158] David Lopez-Paz and Marc'Aurelio Ranzato. “Gradient episodic memory for continual learning”. In: *Advances in neural information processing systems* 30 (2017).

[159] Yiping Lu et al. “Understanding and improving transformer from a multi-particle dynamic system point of view”. In: *arXiv preprint arXiv:1906.02762* (2019).

[160] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. “Effective approaches to attention-based neural machine translation”. In: *arXiv* (2015).

[161] David JC MacKay, David JC Mac Kay, et al. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[162] Stefano Markidis et al. “Nvidia tensor core programmability, performance & precision”. In: *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*. IEEE. 2018, pp. 522–531.

[163] James Martens. “New insights and perspectives on the natural gradient method”. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5776–5851.

[164] Markus Miiller, Sebastian Stiiker, and Alex Waibel. “Multilingual adaptation of RNN based ASR systems”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 5219–5223.

[165] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *Proceedings of ICLR Workshops Track*. 2013.

[166] Tomas Mikolov et al. “Recurrent neural network based language model.” In: *Interspeech*. Vol. 2. 3. Makuhari. 2010, pp. 1045–1048.

[167] Tomáš Mikolov et al. “Extensions of recurrent neural network language model”. In: *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2011, pp. 5528–5531.

[168] James L Morgan and Katherine Demuth. *Signal to syntax: Bootstrapping from speech to grammar in early acquisition*. Psychology Press, 2014.

[169] Markus Müller, Sebastian Stüker, and Alex Waibel. “Neural Codes to Factor Language in Multilingual Speech Recognition”. In: *ICASSP*. 2019.

[170] Markus Müller, Sebastian Stüker, and Alex Waibel. “Neural language codes for multilingual acoustic models”. In: *arXiv preprint* (2018).

[171] Markus Müller, Sebastian Stüker, and Alex Waibel. “Phonemic and graphemic multilingual ctc based speech recognition”. In: *arXiv preprint arXiv:1711.04564* (2017).

[172] Hermann Ney, Ute Essen, and Reinhard Kneser. “On structuring probabilistic dependences in stochastic language modelling”. In: *Computer Speech & Language* 8.1 (1994), pp. 1–38.

[173] Nathan Ng et al. “Facebook FAIR’s WMT19 News Translation Task Submission”. In: Florence, Italy: Association for Computational Linguistics, Aug. 2019.

[174] Thai-Son Nguyen, Sebastian Stüker, and Alex Waibel. “Superhuman performance in online low-latency recognition of conversational speech”. In: *arXiv preprint arXiv:2010.03449* (2020).

[175] Thai-Son Nguyen et al. “Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation”. In: *arXiv preprint arXiv:1910.13296* (2019).

[176] Thai-Son Nguyen et al. “Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation”. In: *arXiv preprint arXiv:1910.13296* (2019).

[177] Xuan-Phi Nguyen et al. “Data diversification: A simple strategy for neural machine translation”. In: *arXiv preprint arXiv:1911.01986* (2019).

[178] Jan Niehues et al. “Low-Latency Neural Speech Translation”. In: *Proc. Interspeech 2018*. 2018, pp. 1293–1297. DOI: 10.21437/Interspeech.2018-1055.

[179] Jan Niehues et al. “The IWSLT 2019 Evaluation Campaign”. In: *16th International Workshop on Spoken Language Translation 2019*. Zenodo, Nov. 2019. DOI: 10.5281/zenodo.3525578.

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

[181] Kishore Papineni et al. “BLEU: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318.

[182] German I Parisi et al. “Continual lifelong learning with neural networks: A review”. In: *Neural Networks* 113 (2019), pp. 54–71.

[183] Daniel S Park et al. “SpecAugment: A simple data augmentation method for automatic speech recognition”. In: *arXiv* (2019).

[184] Razvan Pascanu and Yoshua Bengio. “Revisiting natural gradient for deep networks”. In: *arXiv preprint arXiv:1301.3584* (2013).

[185] Adam Paszke et al. “Automatic differentiation in pytorch”. In: (2017).

[186] Yifan Peng et al. “Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 17627–17643.

[187] Juan M Perero-Codosero, Fernando M Espinoza-Cuadros, and Luis A Hernández-Gómez. “A Comparison of Hybrid and End-to-End ASR Systems for the IberSpeech-RTVE 2020 Speech-to-Text Transcription Challenge”. In: *Applied Sciences* 12.2 (2022), p. 903.

[188] Ngoc-Quan Pham, German Kruszewski, and Gemma Boleda. “Convolutional neural network language models”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 1153–1162.

[189] Ngoc-Quan Pham, Jan Niehues, and Alexander Waibel. “The Karlsruhe Institute of Technology Systems for the News Translation Task in WMT 2018”. In: *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*. Belgium, Brussels: Association for Computational Linguistics, 2018, pp. 467–472. URL: http://aclweb.org/anthology/W18-6422.

[190] Ngoc-Quan Pham, Alex Waibel, and Jan Niehues. “Adaptive multilingual speech recognition with pretrained models”. In: *Proc. Interspeech 2022*. 2022.

[191] Ngoc-Quan Pham, Alexander Waibel, and Jan Niehues. “Adaptive multilingual speech recognition with pretrained models”. In: *Proc. Interspeech 2022*. 2022, pp. 3879–3883. DOI: 10.21437/Interspeech.2022-872.

[192] Ngoc-Quan Pham et al. “Effective combination of pretrained models-KIT@ IWSLT2022”. In: *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*. 2022, pp. 190–197.

[193] Ngoc-Quan Pham et al. “Efficient Weight Factorization for Multilingual Speech Recognition”. In: *Proc. Interspeech 2021*. 2021, pp. 2421–2425. DOI: 10.21437/Interspeech.2021-216.

[194] Ngoc-Quan Pham et al. “Improving Zero-shot Translation with Language-Independent Constraints”. In: *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 13–23. DOI: 10.18653/v1/W19-5202. URL: https://aclanthology.org/W19-5202.

[195] Ngoc-Quan Pham et al. “Kit’s iwslt 2020 slt translation system”. In: *Proceedings of the 17th International Conference on Spoken Language Translation*. 2020, pp. 55–61.

[196] Ngoc-Quan Pham et al. “Multilingual Speech Translation KIT@ IWSLT2021”. In: *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*. 2021, pp. 154–159.

[197] Ngoc-Quan Pham et al. “Relative Positional Encoding for Speech Recognition and Direct Translation”. In: *Proc. Interspeech 2020*.

[198] Ngoc-Quan Pham et al. “The IWSLT 2019 KIT Speech Translation System”. In: *16th International Workshop on Spoken Language Translation 2019*. Zenodo, Nov. 2019. DOI: 10.5281/zenodo.3525564.

[199] Ngoc-Quan Pham et al. “Very Deep Self-Attention Networks for End-to-End Speech Recognition”. In: *Proc. Interspeech 2019*. 2019, pp. 66–70. DOI: 10.21437/Interspeech.2019-2702.

[200] Jerin Philip et al. “Language Adapters for Zero Shot Neural Machine Translation”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 4465–4470.

[201] Emmanouil Antonios Platanios et al. “Contextual Parameter Generation for Universal Neural Machine Translation”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, 2018.

[202] Peter Polák et al. “CUNI-KIT System for Simultaneous Speech Translation Task at IWSLT 2022”. In: *arXiv preprint arXiv:2204.06028* (2022).

[203] Matt Post. “A Call for Clarity in Reporting BLEU Scores”. In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Belgium, Brussels: Association for Computational Linguistics, Oct. 2018, pp. 186–191. URL: https://www.aclweb.org/anthology/W18-6319.

[204] Matt Post et al. “Fisher and CALLHOME Spanish–English speech translation”. In: *LDC2014T23. Web Download. Philadelphia: Linguistic Data Consortium* (2014).

[205] Daniel Povey. “A tutorial-style introduction to subspace Gaussian mixture models for speech recognition”. In: *Microsoft Research, Redmond, WA* (2009).

[206] Daniel Povey et al. “Boosted MMI for model and feature-space discriminative training”. In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2008, pp. 4057–4060.

[207] Daniel Povey et al. “Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI.” In: *Interspeech*. 2016, pp. 2751–2755.

[208] Daniel Povey et al. “The Kaldi speech recognition toolkit”. In: *IEEE 2011 workshop on automatic speech recognition and understanding*. CONF. IEEE Signal Processing Society. 2011.

[209] Vineel Pratap et al. “Massively multilingual asr: 50 languages, 1 model, 1 billion parameters”. In: *arXiv preprint arXiv:2007.03001* (2020).

[210] Vineel Pratap et al. “Mls: A large-scale multilingual dataset for speech research”. In: *arXiv preprint arXiv:2012.03411* (2020).

[211] Alec Radford et al. “Robust speech recognition via large-scale weak supervision”. In: *arXiv preprint arXiv:2212.04356* (2022).

[212] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. “Searching for Activation Functions”. In: *CoRR* abs/1710.05941 (2017). arXiv: 1710.05941. URL: http://arxiv.org/abs/1710.05941.

[213] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. “Learning multiple visual domains with residual adapters”. In: *Advances in neural information processing systems* 30 (2017).

[214] Steve Renals et al. “Connectionist probability estimators in HMM speech recognition”. In: *IEEE transactions on speech and audio processing* 2.1 (1994), pp. 161–174.

[215] Frank Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[216] Anthony Rousseau, Paul Deléglise, and Yannick Esteve. “TED-LIUM: an Automatic Speech Recognition dedicated corpus.” In: *LREC*. 2012, pp. 125–129.

[217] James Route et al. “Multimodal, multilingual grapheme-to-phoneme conversion for low-resource languages”. In: *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*. 2019, pp. 192–201.

[218] Mickael Rouvier et al. “An open-source state-of-the-art toolbox for broadcast news diarization”. In: 2013.

[219] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[220] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[221] Andrei A Rusu et al. "Progressive neural networks". In: *arXiv preprint arXiv:1606.04671* (2016).

[222] Samik Sadhu and Hynek Hermansky. "Continual Learning in Automatic Speech Recognition." In: *Interspeech*. 2020, pp. 1246–1250.

[223] Elizabeth Salesky et al. "The multilingual tedx corpus for speech recognition and translation". In: *arXiv preprint arXiv:2102.01757* (2021).

[224] Ramon Sanabria et al. "How2: a large-scale dataset for multimodal language understanding". In: *arXiv preprint arXiv:1811.00347* (2018).

[225] George Saon et al. "English conversational telephone speech recognition by humans and machines". In: *arXiv preprint arXiv:1703.02136* (2017).

[226] Steffen Schneider et al. "wav2vec: Unsupervised pre-training for speech recognition". In: *arXiv preprint arXiv:1904.05862* (2019).

[227] Tanja Schultz and Alex Waibel. "Adaptation of pronunciation dictionaries for recognition of unseen languages". In: *Workshop on Speech and Communication (SPECOM-1998)*. Citeseer. 1998, pp. 207–210.

[228] Tanja Schultz and Alex Waibel. "Experiments on cross-language acoustic modeling." In: *INTERSPEECH*. 2001, pp. 2721–2724.

[229] Tanja Schultz and Alex Waibel. "Fast bootstrapping of LVCSR systems with multilingual phoneme sets". In: *Fifth European Conference on Speech Communication and Technology*. Citeseer. 1997.

[230] Tanja Schultz and Alex Waibel. “Language-independent and language-adaptive acoustic modeling for speech recognition”. In: *Speech Communication* 35.1-2 (2001), pp. 31–51.

[231] Holger Schwenk. “Continuous space language models”. In: *Computer Speech & Language* 21.3 (2007), pp. 492–518.

[232] Abigail See, Minh-Thang Luong, and Christopher D Manning. “Compression of neural machine translation models via pruning”. In: *arXiv preprint arXiv:1606.09274* (2016).

[233] R. Sennrich, B. Haddow, and A. Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54st Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. Berlin, Germany, 2016.

[234] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2016.

[235] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. “Self-Attention with Relative Position Representations”. In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. June 2018.

[236] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. “Self-Attention with Relative Position Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 464–468. DOI: 10.18653/v1/N18-2074. URL: https://www.aclweb.org/anthology/N18-2074.

[237] Noam Shazeer. “Glu variants improve transformer”. In: *arXiv preprint arXiv:2002.05202* (2020).

[238] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[239] Hagen Soltau, Hank Liao, and Haşim Sak. “Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition”. In: *Proc. Interspeech 2017*. 2017, pp. 3707–3711. DOI: 10.21437/Interspeech.2017-1566.

[240] Hagen Soltau, George Saon, and Tara N Sainath. “Joint training of convolutional and non-convolutional neural networks”. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 5572–5576.

[241] Matthias Sperber and Matthias Paulik. “Speech translation and the end-to-end promise: Taking stock of where we are”. In: *arXiv preprint arXiv:2004.06358* (2020).

[242] Matthias Sperber et al. “Attention-passing models for robust and data-efficient end-to-end speech translation”. In: *Transactions of the Association for Computational Linguistics (TACL)* (2019).

[243] Matthias Sperber et al. “Self-attentional acoustic models”. In: *arXiv preprint arXiv:1803.09519* (2018).

[244] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[245] Sebastian Stüker et al. “Training time reduction and performance improvements from multilingual techniques on the BABEL ASR task”. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 6374–6378.

[246] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. “End-to-end memory networks”. In: *Advances in neural information processing systems*. 2015, pp. 2440–2448.

[247] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[248] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[249] Yuqing Tang et al. “Multilingual translation with extensible multilingual pretraining and finetuning”. In: *arXiv preprint arXiv:2008.00401* (2020).

[250] Samuel Thomas, Sriram Ganapathy, and Hynek Hermansky. “Cross-lingual and multi-stream posterior features for low resource LVCSR systems”. In: *Eleventh Annual Conference of the International Speech Communication Association*. 2010.

[251] Tijmen Tieleman and Geoffrey Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural Networks for Machine Learning* 4.2 (2012).

[252] Shubham Toshniwal et al. “Multilingual speech recognition with a single end-to-end model”. In: *ICASSP*. 2018.

[253] Ioannis Tsiamas et al. “SHAS: Approaching optimal Segmentation for End-to-End Speech Translation”. In: *arXiv preprint arXiv:2202.04774* (2022).

[254] Zhaopeng Tu et al. “Neural machine translation with reconstruction”. In: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[255] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems*. 2017.

[256] Andreas Veit, Michael J Wilber, and Serge Belongie. “Residual networks behave like ensembles of relatively shallow networks”. In: *Advances in neural information processing systems*. 2016, pp. 550–558.

[257] Petar Veličković et al. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).

[258] Andrew Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE transactions on Information Theory* 13.2 (1967), pp. 260–269.

[259] A. Waibel et al. “Phoneme recognition using time-delay neural networks”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3 (Mar. 1989), pp. 328–339. ISSN: 0096-3518. DOI: 10.1109/29.21701.

[260] Alex Waibel. “Consonant recognition by modular construction of large phonemic time-delay neural networks”. In: *Advances in neural information processing systems* 1 (1988).

[261] Alex Waibel. “Modular construction of time-delay neural networks for speech recognition”. In: *Neural computation* 1.1 (1989), pp. 39–46.

[262] Alex Waibel. “Phoneme recognition using time-delay neural networks”. In: *Meeting of the Institute of Electrical, Information and Communication Engineers (IEICE), Tokyo, Japan* (1987).

[263] Alex Waibel, Hidefumi Sawai, and Kiyohiro Shikano. “Modularity and scaling in large phonemic neural networks”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.12 (1989), pp. 1888–1898.

[264] Alex Waibel and B Yegnanrayana. *Comparative study of nonlinear time warping techniques in isolated word speech recognition systems*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1981.

[265] Alex Waibel et al. “Multilingual Speech Recognition”. In: *Verbmobil: Foundations of Speech-to-Speech Translation*. Ed. by Wolfgang Wahlster. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[266] L Wang and PC Woodland. “Discriminative adaptive training using the MPE criterion”. In: *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No. 03EX721)*. IEEE. 2003, pp. 279–284.

[267] Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. “Balancing Training for Multilingual Neural Machine Translation”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.

[268] Xinyi Wang et al. “SwitchOut: an Efficient Data Augmentation Algorithm for Neural Machine Translation”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 856–861. URL: https://www.aclweb.org/anthology/D18-1100.

[269] Yeming Wen, Dustin Tran, and Jimmy Ba. “BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning”. In: *arXiv preprint* (2020).

[270] Chao Weng et al. “Improving Attention Based Sequence-to-Sequence Models for End-to-End English Conversational Speech Recognition”. In: *Proc. Interspeech 2018* (2018), pp. 761–765.

[271] John Wiseman. *python-webrtcvad*. https://github.com/wiseman/py-webrtcvad. 2016.

[272] Matthias Wolfel and John McDonough. “Minimum variance distortionless response spectral estimation”. In: *IEEE Signal Processing Magazine* 22.5 (2005), pp. 117–126.

[273] Wayne Xiong et al. “Achieving human parity in conversational speech recognition”. In: *arXiv preprint arXiv:1610.05256* (2016).

[274] Kelvin Xu et al. “Show, attend and tell: Neural image caption generation with visual attention”. In: *International conference on machine learning*. PMLR. 2015, pp. 2048–2057.

[275] Brian Yan et al. “CTC Alignments Improve Autoregressive Translation”. In: *arXiv preprint arXiv:2210.05200* (2022).

[276] Muqiao Yang, Ian Lane, and Shinji Watanabe. “Online Continual Learning of End-to-End Speech Recognition Models”. In: *Proc. Interspeech 2022*. 2022, pp. 2668–2672. DOI: 10.21437/Interspeech.2022-11093.

[277] Jaehong Yoon et al. “Scalable and order-robust continual learning with additive parameter decomposition”. In: *arXiv preprint arXiv:1902.09432* (2019).

[278] Steve J Young et al. “Multilingual large vocabulary speech recognition: the European SQALE project”. In: *Computer Speech & Language* 11.1 (1997), pp. 73–89.

[279] Chengzhu Yu et al. “A multistage training framework for acoustic-to-word model”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2018).

[280] Mingzhi Yu et al. “Multilingual grapheme-to-phoneme conversion with byte representation”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 8234–8238.

[281] Wojciech Zaremba. “An empirical exploration of recurrent network architectures”. In: (2015).

[282] Matthew D Zeiler. “ADADELTA: an adaptive learning rate method”. In: *arXiv preprint arXiv:1212.5701* (2012).

[283] Friedemann Zenke, Ben Poole, and Surya Ganguli. “Continual learning through synaptic intelligence”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3987–3995.

[284] Albert Zeyer et al. “A comparison of transformer and lstm encoder decoder models for asr”. In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE. 2019, pp. 8–15.

[285] Albert Zeyer et al. “Improved Training of End-to-end Attention Models for Speech Recognition”. In: *Proc. Interspeech 2018* (2018), pp. 7–11.

[286] Biao Zhang, Ivan Titov, and Rico Sennrich. “Sparse Attention with Linear Units”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6507–6520. DOI: 10.18653/v1/2021.emnlp-main.523. URL: https://aclanthology.org/2021.emnlp-main.523.

[287] Biao Zhang et al. “Improving Massively Multilingual Neural Machine Translation and Zero-Shot Translation”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. July 2020.

[288] Ying Zhang et al. “Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks”. In: *Proc. Interspeech 2016*. 2016, pp. 410–414. DOI: 10.21437/Interspeech.2016-1446.

[289] Yu Zhang, William Chan, and Navdeep Jaitly. “Very deep convolutional networks for end-to-end speech recognition”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 4845–4849.

[290] Shiyu Zhou, Shuang Xu, and Bo Xu. “Multilingual end-to-end speech recognition with a single transformer on low-resource languages”. In: *arXiv preprint arXiv:1806.05059* (2018).

[291] Yun Zhu et al. “Multilingual Speech Recognition with Self-Attention Structured Parameterization”. In: *Interspeech*. 2020.

[292] Julian Georg Zilly et al. “Recurrent highway networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 4189–4198.

[293] Geoffrey Zweig et al. “Advances in all-neural speech recognition”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 4805–4809.