# Exploiting Compositionality in Sequence Models

Siddharth Dalmia

CMU-LTI-22-015

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

**Thesis Committee:**

Florian Metze, CMU (co-chair)
Alan W Black, CMU (co-chair)
Shinji Watanabe, CMU
Abdelrahman Mohamed, FAIR

*Submitted in partial fulfillment of the requirements*
*for the degree of Doctor of Philosophy*
*In Language and Information Technologies*

**Keywords:** compositionality, modularity, reusability, performance monitoring, search and retrieval, sequence models for speech and language

*To my family*

## Abstract

Compositionality is the principle behind building complex systems by composing together simpler sub-systems. First, a complex task is broken down into a pipeline of simpler, more straightforward tasks. Once systems are built for the simpler tasks, intricate details are abstracted away and components are pipelined together to reason about the overall task. The knowledge and resources spent building the systems for the sub-task are then reused towards building various complex systems. This divide-and-conquer approach to compositionality promotes the flexibility, efficiency, and overall practicality of these systems.

While traditional sequence systems such as cascade models leveraged compositionality, contemporary end-to-end models such as encoder-decoder models fail to satisfy even the basic compositionality requirements, i.e., having a clear understanding of the function of each component. The lack thereof hinders the practical use of end-to-end systems, despite these approaches having advanced the state-of-the-art in a wide range of sequence tasks.

In this thesis, with a focus on sequence tasks for speech and language, we identify four characteristics of compositionality that facilitate the practical deployment of end-to-end models, i.e., having component or sub-task level (1) Performance Monitoring, (2) Search and Retrieval, (3) Resource Pooling, and (4) Reusability. We present three models with the above characteristics, which exhibit different levels of reusability behavior, from direct plug-and-play ability to the ability for further fine-tuning towards the end task. The first is the *CTC Hybrid* model, which creates a hybrid of models by decomposing a sequence task into alignment using a CTC model and language generation using a language model. For a fully differentiable alternative, we present *LegoNN* modular encoder-decoder models, which build reusable encoder and decoder modules across various sequence tasks, with the ability for further fine-tuning. Lastly, we present our *Compositional E2E* model with searchable hidden intermediates that allows using the sub-task formulations to build an end-to-end task model. It also allows reusing pre-trained sub-task models for retrieving better intermediate representations in the fully-differentiable model.

Finally, we discuss practical implications on the evaluation of end-to-end models, where we show how to make their evaluation more reliable and informative by testing their generalizability towards each sub-task in a complex sequence task.

## Acknowledgments

First and foremost, I would like to thank my advisors Florian Metze and Alan W Black for their continued support, involvement, and help during this journey. They spent countless hours discussing research and problems in the current systems with me. I enjoyed hearing about their experiences with building sequence systems before the deep learning era. Learning about its flexibility and practicality inspired this thesis. Thank you for always taking a practitioners' standpoint during our brainstorming sessions. Thank you for all the support during internships and job search, and for guiding me to grow into a better researcher and individual over the last six years.

I also want to thank my mentors and the members of my committee, Shinji Watanabe and Abdelrahman Mohamed. Shinji, I cannot thank you enough for your involvement in my research. It has been a great experience working with you and all of the members of your lab. Thank you for the support during the final years of my PhD. Thank you believing in my abilities and giving me numerous opportunities to mentor students, involving me in research discussions and helping me build new connections and collaborations. Abdo, your research work has been my inspiration since I began my career in speech technologies. Thank you for giving me the opportunity to work with you. I really enjoyed our brainstorming sessions. You helped expand the horizons of my research and helped me realize a more general view of my interests. Thank you for your mentorship and I hope we get to work more in the future.

I am fortunate to have had the opportunity to work with many collaborators throughout my PhD, many of whom have also become my close friends since. I have had a lot of fun discussing research ideas, traveling to conferences, and debugging compute infrastructure with our research group (sinbad), in particular Xinjian Li, Ramon Sanabria, Suyoun Kim, Elizabeth Salesky, Shruti Palaskar, Juncheng Li, Eric Riebling, Yun Wang, Vikas Raunak, Roshan Sharma, and others. I still miss the nice collaborative space we had at 407 S Craig St. I was also lucky to be part of the WAVLab, where I had the opportunity to interact, mentor and collaborate with many students such as Brian Yan, Siddhant Arora, Yifan Peng, Xuankai Chang, Jiatong Shi, Dan Berrebbi, Aswin Subramanian, Matthew Maciejewski, and others. I also would like to thank my internship mentors and collaborators at FAIR, AWS AI and Google. To name a few, Dmytro Okhonko, Abdo, Mike Lewis, Sergey Edunov, Luke Zettlemoyer at FAIR; Katrin Kirchhoff, Yuzong Liu, Srikant Ronanki, Julian Salazar at AWS AI; Ron Weiss, Tara Sainath, Yu Zhang, Alexis Conneau, Ankur Bapna, Simran Khanuja at Google. They introduced me to many new challenges that

exist in practical sequence models, and my interactions with them always involved deep discussions that left me enlightened at the end. I also had the opportunity to work with many brilliant researchers in my department, like Graham Neubig, David Mortensen, Bhiksha Raj, Abhilasha Ravichander, Maria Ryskina, Patrick Fernandes, Vijay Viswanathan, Alissa Ostapenko and the members of the DARPA LORELEI and KAIROS program. I think the work presented here would not have been possible without the contribution and support of all my collaborators. Of many important software libraries that made this thesis possible, I want to particularly thank the maintainers of EESEN and ESPNet for actively helping me with my requests and questions. I would also like to thank Stacey Young, Kate Schaich, Jessica Maguire, and Mary Jo Bensasi for helping simplify our daily life in LTI.

I must also thank the members of the MultiSpeech Lab in INRIA Nancy, specifically, Emmanuel Vincent, Irina Illina, Juan Cordovilla, Imran Sheikh and Sunit Sivasankaran for giving me the first taste of speech research, and my undergraduate professors Aruna Malapati, Tathagata Ray and Bhanu Murthy for getting me excited about research and helping me build a strong foundation to pursue my interests.

I could not do justice to my life in Pittsburgh without thanking the friends I have made during these six years. To name a few, Alex, Dheeraj, Chaitanya, Anant, Shivani, AVK, Devendra, Tejas, Soumya, Saksham, "gedi", Will, Chirag, Khyathi, Aakanksha, Pallu, Paul, Hai, Danish, Sai, Stas, Rohit, Raphael, Arti, Andrew and many others. You guys made this journey much more fun than it would have been otherwise. Thank you for keeping me sane, providing support, and all the laughs and fun we had along the way. My friends from undergrad and school – Varun, Naveen, Bala, Rohan, Shalaka, Tarana, "goodfellas", my hostel wing-mates: I am glad we are still connected across the globe! Brian Yan and Patrick Fernandes – I will really miss our climbing and gym sessions, thank you! Karan Ahuja, Arnav Choudhary, Deepak Gopinath, Rajat Kulsreshtha, Sujeath Pareddy (may his soul rest in peace), Abhilasha Ravichander, Shruti Rijhwani, Maria Ryskina, Samridhi Shree, Eva Spiliopoulou – I think there is no better way to describe them other than as my family in Pittsburgh. Thank you for sticking around by my side during all the difficult and the good times!

Finally, I would like to thank my parents, Sarla and Sharad Dalmia, for their unparalleled emphasis on my education and for the support during my PhD. Thank you for believing in me and giving me every opportunity I needed to go as far as I could in my career. None of this would have been possible without you.

# Contents

# Chapter 1

# Introduction

Compositionality is the principle behind building complex systems by composing together simpler sub-systems using only the high-level understanding of each sub-system. It is studied and applied extensively in the building of various complex systems, from physical machinery (Levis et al., 1994) to software systems (Baldwin and Clark, 1999). Compositionality takes a hierarchical approach towards building systems by reasoning about the problem recursively. First, a complex task is broken down into a pipeline of simpler, more straightforward tasks. Once systems are built for the simpler tasks, intricate details are abstracted away and components are pipelined together to reason about the overall task. The knowledge and resources spent building the systems for the sub-task can be reused in the construction of other complex systems (Zwiers, 1989). A clear interface or understanding of the high-level functionality of each component makes it possible to reuse them and also upgrade them in a modular fashion (Baldwin and Clark, 1999). For example, if a more efficient or a better performing component is developed, all systems using this component can simply upgrade provided the interface remains the same. Furthermore, the breakdown into simpler tasks allows leveraging additional domain knowledge, expertise, and other resources, if available, for each component (Levis et al., 1994). Overall, this divide-and-conquer approach of compositionality promotes the overall practicality of system building, going from creating stand-alone systems to their large-scale development.

While maintaining the holistic view, compositionality takes many forms in each field. A language is compositional if the meaning of every complex expression $E$ in that system depends on, and depends only on, the syntactic structure of $E$ and the meanings of the simple parts of $E$ (Johnson, 2020). Compositionality plays an essential role in understanding languages; for example, paragraph coherence and discourse analysis rely on decomposition into sentences (Johnson, 1992; Kuo, 1995) and sentence-level semantics relies on decomposition into lexical units (Liu et al., 2020c). In sequence modeling, a class of machine learning tasks that work on structured sequences such as speech and text, the models broke down the combinatorial space of sentences by decomposing them into words or other linguistic units such as phonemes or characters, or

byte-pair encoding, aiding in generalization to new sentences while capturing meaning both in the form of representations (Mikolov et al., 2013) and during sentence prediction (Mohri et al., 2002).

The application of compositionality in sequence models does not end there; traditional approaches relied heavily on the principle of compositionality in order to tackle these sequence processing tasks. Cascade models carefully decomposed a complex sequence task into a pipeline of systems that individually solve sub-tasks required to solve the overall problem. For example, speech translation systems that seek to process speech in one language and output text in another language were built by pipelining together speech recognition (ASR) and machine translation (MT) systems. Such chaining of sub-systems is used extensively to build practical systems for various sequence tasks such as hybrid ASR (Hinton et al., 2012), phrase-based MT (Koehn et al., 2007), and cascaded ASR-MT systems for speech translation (ST) (Pham et al., 2019).

With the recent explosion in computing power, fully differentiable end-to-end approaches, such as encoder-decoder models, have become popular and effective for sequence tasks. They have played a critical role in advancing the state-of-the-art in a wide range of sequence tasks, such as ASR (Park et al., 2019) and MT (Vaswani et al., 2017). However, these end-to-end models are constructed and trained as an atomic unit, where all components are tightly coupled to each other, preventing a clear understanding of the function of each part (Dalmia et al., 2019b). Thereby, they fail to satisfy even the basic requirement for compositionality, i.e. having a clear interface between components. This incapability to follow the principles of compositionality limits the practicality of these systems in several ways. Specifically, such models are unable to:

1. **Measure the quality of the individual components of the end-to-end model** - This prevents diagnosis of individual components, causing practitioners to invest more time and compute resources in selecting hyperparameters, discovering bugs and reasoning about the architecture of the model.
2. **Reuse/transfer components across different tasks** - This leads to practitioners building individual stand-alone systems for each new task they would like to support, wasting compute on components that share logical functions with components from an already trained model for a different task.
3. **Perform search and retrieval over the components for each sub-task** - This prevents practitioners from adapting individual components to other domains using external models. For example, while adapting an end-to-end speech translation model on a different speech domain but the same translation style.
4. **Utilize additional resources for training individual components** - There is still a significant lack of datasets for complex sequence tasks, as needed by end-to-end models, representing real-world scenarios that require a comprehensive coverage of different domains and languages. Practitioners would benefit from having the ability to make use of the more

comprehensive datasets available for simpler tasks to mitigate this issue.

Therefore, despite the advantages of end-to-end approaches in simplicity (Bahdanau et al., 2015; Chan et al., 2016) and effectiveness (Vaswani et al., 2017; Park et al., 2019), traditional cascade models, which followed the principles of compositionality, are still a more popular choice for practical deployment among practitioners. It is important to consider how to incorporate compositional properties into the next generation of end-to-end models.

In this thesis, we identify four characteristics of compositionality, which facilitate the practical deployment of end-to-end models: component or sub-task level (1) Performance Monitoring, (2) Search and Retrieval, (3) Resource Pooling, and (4) Reusability. We present three models that have the above characteristics, which exhibit different levels of reusability behavior, from direct plug-and-play ability to the ability for further fine-tuning towards the end task. Our first model is the *CTC Hybrid* model, which creates a hybrid of models by decomposing a sequence task into alignment using a CTC model and language generation using a language model. For a fully differentiable alternative, we present *LegoNN* modular encoder-decoder models, which build reusable encoder and decoder modules across various sequence tasks, with the ability for further fine-tuning. Lastly, we present our *Compositional E2E* model with searchable hidden intermediates that allows using the sub-task formulations to build an end-to-end task model. It also allows reusing pre-trained sub-task models for retrieving better intermediate representations in the fully-differentiable model.

To round out the thesis, we discuss practical implications on the evaluation of end-to-end models, where we show how to make their evaluation more reliable and informative by testing their generalizability towards each sub-task in a complex sequence task.

## 1.1 Thesis Outline and Contributions

A more detailed outline of this thesis is as follows:

1. Chapter 2 discusses the past, present, and our intentions for the future of sequence models. This chapter would provide a background on sequence models, covering both the traditional sequence models that exploited compositionality and the current state of end-to-end models. We then discuss the practical implications that we would like to consider for the next generation of end-to-end models.
2. Chapter 3 discusses the CTC Hybrid Models. These models create a hybrid system by decomposing a sequence task into alignment using a CTC model and language generation using a language model. For the speech recognition task, we show that by decomposing words into language-independent phone sequences we can pool resources from other languages to learn better CTC models (Dalmia et al., 2018). This work has also been the basis for our larger efforts on pooling resources across languages to build multilingual

speech recognition models (Li et al., 2019, 2020b) and language models (Dalmia et al., 2019a). This work was applied to our DARPA LORELEI 2018 and 2019 speech challenge submissions for bootstrapping speech recognition systems under extremely low-resource conditions (Chaudhary et al., 2019).

3. Chapter 4 discusses our LegoNN framework for building modular encoder-decoder models (Dalmia et al., 2022). These are fully-differentiable end-to-end encoder-decoder models that have a defined interface between the encoder and the decoder components. Without any fine-tuning, the decoder from one task, say MT, can be used towards another like ASR. LegoNN models composed from multiple tasks and domains can also be extended to have more than two modular components by further decomposing the task. We also studied the effectiveness of enforcing modularity in ASR systems in (Dalmia et al., 2019b).

4. Chapter 5 builds upon the LegoNN encoder systems to extend the CTC formulation to non-monotonic sequence tasks such as machine and speech translation and shows its efficacy towards joint CTC and Attention modeling (Yan et al., 2022a). We also discuss two techniques of decoding models that are jointly modeling both the CTC and Attention based formulations for sequence tasks. This work intends to strengthen Chapter 3 by showing how the CTC monotonic and length assumptions can be relaxed by performing reordering and length-adjustment in the encoder, making it ideal for non-monotonic sequence tasks. It also strengthens Chapter 4 by allowing a more advanced one-pass beam search that considers both the CTC and Attention distributions.

5. Chapter 6 and Chapter 7 discusses our Compositional E2E model with searchable hidden intermediates. We present an end-to-end framework that exploits compositionality to learn searchable hidden representations at intermediate stages of a sequence model using decomposed sub-tasks. These hidden intermediates can be improved using beam search to enhance overall performance and can also incorporate external models at intermediate stages of the network to search for better representations. We present two instances of our Compositional E2E model, (1) Speech Translation (Dalmia et al., 2021) and (2) Spoken Language Understanding (Arora et al., 2022) that extracts hidden intermediates from a speech recognition sub-task. This work was crucial in our submissions to IWSLT 2021 (Inaguma et al., 2021b) and IWSLT 2022 (Yan et al., 2022b). We also show its effectiveness in translating endangered languages such as Nahutl (Shi et al., 2021).

6. Chapter 8 discusses practical implications for the evaluation of end-to-end sequence models. We show that as sequence tasks become complex testing their generalizability on input variations are not enough and it is important to test generalizability towards each sub-task of the overall task. Given a dataset for a decomposable task, like Spoken Language Understanding, our method optimally creates a test set for each sub-task to individually assess sub-components of the end-to-end model (Arora et al., 2021).

# Chapter 2

# Sequence Modeling Systems: Past, Current and Future

## 2.1 Past: Cascade Systems

Our work takes inspiration from the traditional cascaded systems (Koehn et al., 2007; Povey et al., 2011; Pham et al., 2019), which use several sub-components that are trained separately. These systems successfully exploited search capabilities of the cascaded systems to compose the final task output from individual system predictions using beam-search (Inaguma et al., 2020b; Pham et al., 2019) and more sophisticated lattice based search (Koehn et al., 2007; Povey et al., 2011; Zhang et al., 2019; Beck et al., 2019). They have the ability to incorporate external models to re-score each individual system (Och and Ney, 2002; Huang and Chiang, 2007), the ability to easily adapt individual components towards out-of-domain data (Koehn and Schroeder, 2007; Peddinti et al., 2015), and finally the ability to monitor performance of the individual systems towards the decomposed sub-task (Tillmann and Ney, 2003; Meyer et al., 2016).

## 2.2 Current: E2E Encoder-Decoder Systems

Recently, there has been a growing interest in training a single end-to-end (E2E) system to perform these sequence-to-sequence tasks using methods such as the encoder-decoder models (Bahdanau et al., 2015; Chan et al., 2015; Vaswani et al., 2017), the non-autoregressive models (Gu et al., 2018; Chan et al., 2020; Saharia et al., 2020) and the tranducer based models (Graves, 2012; Zhang et al., 2020). These end-to-end approaches are attractive in part due to their simplistic design and the reduced need for hand-crafted features; however, studies have shown mixed results compared to cascaded systems particularly for complex sequence tasks like like speech translation (Inaguma et al., 2020b) and spoken language understanding (Coucke et al., 2018). Among these, we are particularly interested in popular end-to-end encoder-decoder models (Bahdanau

Table 2.1: Encoder swap of the attention based encoder-decoder model trained with a different random seed tested on two different sequence tasks, ASR, measured in % word error rate (% WER (↓)), and MT, measured in BLEU (↑).

| | Sequence Model | Eval 2000 (↓) SWB | CH | newstest2014 (↑) En → De |
|---|---|---|---|---|
| Original | Attention Based Enc-Dec | 8.5 | 17.7 | 29.7 |
| **After Encoder Swap** | Attention Based Enc-Dec | 569.0 | 892.0 | 0.2 |

et al., 2015; Chan et al., 2016; Vaswani et al., 2017) that have played a critical role in a wide range of NLP and speech tasks.

Conditioned on previously generated output tokens and the full input sequence, encoder-decoder models (Sutskever et al., 2014) factorize the joint target sequence probability into a product of individual time steps. They are trained by minimizing the token-level cross-entropy (CE) loss between the true and the decoder predicted distributions. Input sequence information is encoded into the decoder output through an attention mechanism (Bahdanau et al., 2015) which is conditioned on current decoder states, and run over the encoder output representations.

The problem of tight coupling of the encoder and decoder components in the end-to-end sequence model is highlighted in Table 2.1. The decoder cross-attention over the encoder hidden representation makes it not only conditioned on the encoder outputs but also dependent on the encoder architectural decisions and internal hidden representations. The whole ASR or MT system fall apart under a simple test of re-usability, i.e. switching an encoder with another similar one that is only different in its initial random seed, which brings our point about the lack of compositionality in encoder-decoder models.

## 2.3 Future of E2E Systems

In this thesis, we would like to bridge the practical considerations of the cascaded systems along with the strong modeling capabilities of the end-to-end models. We envision a future of end-to-end models that have the practical considerations that go towards building cascaded systems. By exploiting the principles of compositionality, we believe that we can build end-to-end systems that are practical for real-world scenarios while avoiding the issues of pipelining systems, like error propagation. The practical considerations that we consider in our thesis is growing interest and are being studied individually (Andreas et al., 2016; Raunak et al., 2019; Lake, 2019), however a holistic view with a unified end-to-end framework for such practical considerations is still missing. We will now introduce the properties of compositionality that we would like to bring to end-to-end systems and present some related works.

### 2.3.1 Performance Monitoring

We would like to have the ability to monitor the performances of the individual components of an end-to-end model. This would allow practitioners to diagnose and reason on which components of the end-to-end model is working better or worse. Such capabilities were used extensively in cascaded approaches (Tillmann and Ney, 2003; Meyer et al., 2016), where the community would question the quality of individual components of the neural network along with the quality of doing the overall task. In the current state of end-to-end models, it is possible to only evaluate the model towards the performance of the end-task and the decoders components for encoder-decoder models in terms of perplexity (Bahdanau et al., 2015). There has been some work on post-hoc analysis the representations learned by the encoders by evaluating them on an auxiliary task (Palaskar et al., 2019), however these task does not necessarily tests the quality with respect to the overall task and an improvement in these auxiliary task may not translate towards improvement towards the performance on the final task.

### 2.3.2 Reusability

We would like to have end-to-end models that have reusable components. Where a component trained on one task can be re-used towards a component for another task or domain. For example, components that share the same functionality across tasks should be reusable. Cascaded systems (Sperber and Paulik, 2020) have used this elegant property to reduce wastage of compute and time spend in building these systems by re-using systems that share a common functionality, like ASR and MT systems being re-used for building speech translation systems or language models being reused for both ASR and MT systems. In the end-to-end research previous works have considered pre-training using trained components (Zheng et al., 2021; Inaguma et al., 2020b). There have been many also proposals for inducing a modular structure on the space of learned concepts either through hierarchically gating information flow or via high-level concept blueprints (Andreas et al., 2016; Devin et al., 2017; Purushwalkam et al., 2019) to enable zero- and few-shot transfer learning (Andreas et al., 2017; Socher et al., 2013; Gupta et al., 2020; Pathak et al., 2019).

### 2.3.3 Search and Retrieval

We would like to have the ability of cascaded systems to compose a final output by using the prediction from individual components of an end-to-end model with sophisticated search techniques like beam search (Inaguma et al., 2020b; Pham et al., 2019) or lattice based search (Koehn et al., 2007; Povey et al., 2011; Zhang et al., 2019; Beck et al., 2019). In an end-to-end model, the input sequence undergoes many transformations from representations in one modality to representations in another. For instance, an end-to-end build for spoken language understanding task (Lugosch et al., 2019) first learns a sequence of speech representations which is then transformed

into a sequence of word representations, which is followed by learning a sentence representation before the final task of predicting intents. The ability for search and retrieval would allow practitioners to adapt individual components of the neural network towards new domains and style intended for each modality transformation leading to the final task.

### 2.3.4 Resource Pooling

We would like the ability to be able to pool resources from the decomposed tasks to build an end-to-end system towards the complex task. Traditional cascade models are conveniently exploiting this benefit by training individual systems on the decomposed task (Sperber and Paulik, 2020). With the advancement in pre-training and multi-task learning this aspect of compositionality is gaining interest among the end-to-end sequence modeling community. Due to the limited availability of the end-to-end datasets, recent works have considered using self-supervised techniques to learn better representation models in both speech (Baevski et al., 2020) and language (Devlin et al., 2019). We consider this work to be complimentary to our work, as the representation learning framework is task agnostic and requires adaptation towards the task at hand. There has also been many works to pool resources across languages for a variety of sequence tasks like machine translation (Liu et al., 2020a), speech recognition (Dalmia et al., 2018, 2019a; Conneau et al., 2020) and speech translation (Li et al., 2020a)

# Part I

## CTC Hybrid and LegoNN Modular E2E Systems

# Chapter 3

# CTC Hybrid Systems

The following Chapters 3, 4 and 5 introduces the building blocks for our CTC Hybrid Systems and LegoNN Modular Encoder-Decoder Systems. Chapter 3 gives a background on CTC Hybrid Systems built for speech recognition (Miao et al., 2015) and presents our work that utilizes the task decomposition in CTC Hybrid Systems to pool resources across languages to build strong ASR systems for low resource speech recognition. Chapter 4 then discusses how to build our end-to-end Modular Encoder-Decoder Systems using the LegoNN framework. In this work, we also propose building modality agnostic CTC encoders showing how we can extend the CTC Hybrid Systems to non-monotonic sequence prediction tasks. Chapter 5 shows how we can build joint CTC/Attn models (Kim et al., 2017) for non-monotonic sequence tasks like speech and machine translation. Although the joint model does not have the properties of our intended future for sequence models §2.3, this work is motivated through principles of task compositionality and further strengthens our work on CTC encoders for non-monotonic sequence task. These CTC encoders built for speech and machine translation can then be used to build CTC Hybrid Systems as it was shown in Chapter 3 for building ASR systems.

## 3.1 Connectionist Temporal Classification

The Connectionist Temporal Classification (CTC) algorithm is an alignment-free sequence prediction approach which marginalizes over the likelihoods of all possible alignments of an input $\mathbf{X}$ of length $T$ and an output $\mathbf{Y}$ of length $L$, where $L \leq T$ (Graves et al., 2006). CTC resolves the input-output length mismatch by introducing a blank token $\varnothing$ to represent a null emission, defining an output vocabulary of $\mathcal{V} + \varnothing$ where $\mathcal{V}$ is the original vocabulary of $\mathbf{Y}$. Now a possible output alignment $\mathbf{A} = [a_1, ..., a_T]$ for $a_t \in \{\mathcal{V} + \varnothing\}$ can be produced with the same length as $\mathbf{X}$. With the set of all possible alignments $\mathcal{A}_{\mathbf{X},\mathbf{Y}}$, the overall likelihood of an output $\mathbf{Y}$ is:

$$p(\mathbf{Y}|\mathbf{X}) = \sum_{\mathbf{A}\in\mathcal{A}_{\mathbf{X},\mathbf{Y}}} \prod_{t=1}^{T} p_t(a_t|\mathbf{X}) \tag{3.1}$$

$$\log p(\mathbf{Y}|\mathbf{X}) = \underset{\mathbf{A}\in\mathcal{A}_{\mathbf{X},\mathbf{Y}}}{\textsc{LogSumExp}}\Big(\sum_{t=1}^{T} \log p_t(a_t|\mathbf{X})\Big) \tag{3.2}$$

$$= \sum_{t=1}^{T} \underset{\mathbf{A}\in\mathcal{A}_{\mathbf{X},\mathbf{Y}}}{\textsc{LogSumExp}}\Big(\log p_t(a_t|\mathbf{X})\Big) \tag{3.3}$$

Note that due to the conditional independence assumption in CTC, we can write Equation (3.3) where the summand corresponds to the combined probability densities from all possible alignments for emitting tokens $\mathcal{V} + \varnothing$ at a particular time-step $t$.

In a non-autoregressive CTC network an Encoder models a sequence of hidden representations $\mathbf{h} = [\mathbf{h}_i, ..., \mathbf{h}_T]$, where $\mathbf{h_t} \in \mathbb{R}^d$. The output emission log-probabilities is given by the affine projection of $\mathbf{h}$ followed by the log-softmax function, denoted as SoftmaxOut. These emission log-probabilities are equivalently represented as a weighted finite-state transducer (WFST) $\mathcal{E}$ with $T$ states, where each state $t$ in $\mathcal{E}$ is connected to state $t+1$ by $|\mathcal{V} + \varnothing|$ arcs whose weights are defined as the emission log-probabilities at $t$, $\textsc{SoftmaxOut}(\mathbf{h}_t)$.

$$\mathbf{h} = \textsc{Encoder}(\mathbf{X}) \tag{3.4}$$

$$\mathcal{E}_{t\to t+1} = \textsc{SoftmaxOut}(\mathbf{h_t}) \tag{3.5}$$

During training, we seek to maximize the overall log-likelihood of $\mathcal{A}_{\mathbf{X},\hat{\mathbf{Y}}}$ for the target sequence $\hat{\mathbf{Y}}$ by constructing the target-constrained emissions WFST, $\mathcal{C}_{\mathbf{X},\hat{\mathbf{Y}}}$, which is the intersection of the log-probabilistic emission graph $\mathcal{E}$ with an unweighted target graph $\mathcal{Y}$. $\mathcal{Y}$ is constructed with the possible representations of $\hat{\mathbf{Y}}$ under the CTC rules for blank tokens and repeated tokens.

Rather than producing a soft alignment between the input and target sequences, the CTC loss (Graves et al., 2006) maximizes the log conditional likelihood of the output, by integrating over all possible monotonic alignments between both sequences. During training, the forward-backward algorithm is used for efficient computation of the marginalization sum. For inference, only one forward pass through the encoder-only model is needed to generate output probability distributions over the full target sequence in a non-autoregressive fashion.

### 3.1.1 Results using Transformer Encoders and SpecAugment

Since the introduction of CTC based models, there has been great strides towards improving encoders using Transformers (Vaswani et al., 2017) and stabilizing training using SpecAugmen-

Table 3.1: Results, measured in % WER (↓), presenting the CTC model trained using transformer encoder blocks (Vaswani et al., 2017) and SpecAugmentation (Park et al., 2019). We provide an LSTM CTC model trained using a similar sized BPE units presented in (Sanabria and Metze, 2018) for comparision.

| Model | SWBD (↓) | CallHome (↓) |
|---|---|---|
| LSTM-CTC (Sanabria and Metze, 2018) | 20.6 | 33.5 |
| Transformer-CTC | **11.5** | **24.1** |

tation (Park et al., 2019). They have enabled training larger models and also towards target units that are larger than character or phoneme level. To compare with the LSTM-CTC based models like in (Sanabria and Metze, 2018) we trained a Trasnformer-CTC based model on the 300h Switchboard Conversational data and tested them on the eval2000 test sets. Table 3.1 shows that we can build stronger CTC based models with transformer based encoders and SpecAugmentation. Details about the model and the dataset preparations are provided in §4.5.1.

## 3.2 CTC Hybrid Systems

Context independent models trained using the CTC loss can be elegantly combined with a language model to generate output sequences that are conditioned on the context. Thereby, behaving like a decoder from an encoder-decoder model while retaining the traditional separation of cascaded systems. In the context of speech recognition, the CTC model behaves like an acoustic model and the separation from language model allows for domain independence and adaptation or re-use of individual components. Having this hybrid approach also allows training of the language model on larger amounts of text data available (Zenkel et al., 2017). The ability to modify individual components also allow practitioners to modify the decoding strategies based on the scenario, like a WFST based decoding (Miao et al., 2015) allows for a stronger search capability but have a fixed vocabulary decoding. On the other hand, an RNN-LM based decoding (Zenkel et al., 2017; Dalmia et al., 2019a) allows for modelling longer contexts and open vocabulary decoding but are limited to left-to-right beam search based re-scoring.

Table 3.2 presents the state-of-the-art CTC Hybrid ASR model that we developed using transformer encoders (Vaswani et al., 2017) and SpecAugmentation (Park et al., 2019) on the 300 hour Switchboard dataset. These models are comparable to the current encoder decoder baselines while maintaining the flexibility that we desire from a sequence system. Additional details on the model and the dataset preparation is provided in §4.5.1. We use the WFST based decoding (Miao et al., 2015) with a trigram language model and the vocabulary from the training data. We do not use any additional data for a fair comparision with the encoder-decoder baseline. However, unlike encoder-decoder models, this performance can be improved further by simply adding

Table 3.2: Results, measured in % WER (↓), presenting the CTC Hybrid models for the speech recognition task. The models are trained on the 300h Switchboard corpora and tested on the eval2000 test sets. We provide a baseline encoder-decoder model for comparision.

| Model | SWBD (↓) | CallHome (↓) |
|---|---|---|
| Baseline Enc-Dec | 8.5 | 18.0 |
| Transformer-CTC | 11.5 | 24.1 |
| + WFST Decoding | **9.1** | **19.4** |

more in-domain text to train the language model (Sanabria and Metze, 2018).

Such hybrid models with non-autoregressive CTC encoders and LM based decoding are becoming increasingly popular as self-supervised models are improving (Baevski et al., 2020) as the CTC models allow for easy tuning of the self-supervised models. Additionally, with the adaptation of CTC models for other non-monotonic sequence tasks (Saharia et al., 2020; Libovický and Helcl, 2018; Inaguma et al., 2020a) we expect that these hybrid models would start to re-appear as the future of sequence modeling.

## 3.3 Resource Pooling across languages through decomposition: Multilingual Low Resource Speech Recognition

Techniques for multilingual and crosslingual speech recognition can help in low resource scenarios, to bootstrap systems and enable analysis of new languages and domains. End-to-end approaches, in particular sequence-based techniques, are attractive because of their simplicity and elegance. While it is possible to integrate traditional multilingual bottleneck feature extractors as front-ends, we show that end-to-end multilingual training of sequence models is effective on context independent models trained using Connectionist Temporal Classification (CTC) loss. We show that our model improves performance on Babel languages by over 6% absolute in terms of word/phoneme error rate when compared to monolingual systems built in the same setting for these languages. We also show that the trained model can be adapted crosslingually to an unseen language using just 25% of the target data. We show that training on multiple languages is important for very low resource crosslingual target scenarios, but not for multilingual testing scenarios. Here, it appears beneficial to include large well prepared datasets.

## 3.4 Introduction

State-of-the-art speech recognition systems with human-like performance (Saon et al., 2017; Xiong et al., 2016) are trained on hundreds of hours of well-annotated speech. Since annotation is an expensive and time-consuming task, similar performance is typically unattainable on low resource languages. Multilingual or crosslingual techniques allow transfer of models or features from well-trained scenarios to those where large amounts of training data may not be available, cannot be transcribed, or are otherwise hard to come by (Stolcke et al., 2006; Grézl et al., 2016).

The standard approach is to train a context dependent Hidden Markov Model based Deep Neural Network acoustic model with a "bottleneck" layer using a frame based criterion on a large multilingual corpus (Veselỳ et al., 2012; Knill et al., 2013; Vu et al., 2012). The network up to the bottleneck layer can be used as a language-independent feature extractor while adapting to a new language. Generating such a model requires the preparation of frame level segmentation in each language, which is usually achieved by training separate monolingual systems first. This is a cumbersome multi-step process. Moreover, if the speaking style, acoustic quality, or linguistic properties of the recordings are very different across a set of languages, the segmentations may be inconsistent across languages and thus sub-optimal for generating features in a new language.

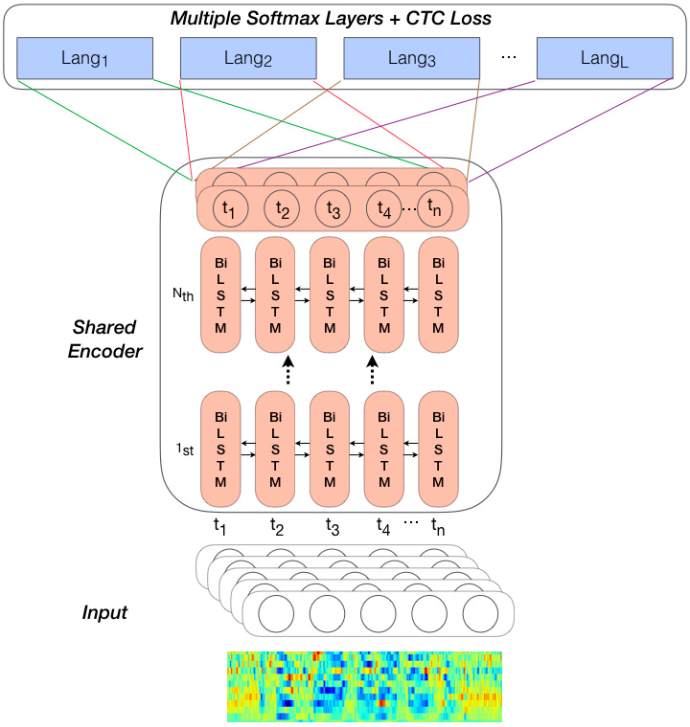



Figure 3.1: Multilingual CTC model following the "shared hidden layer" approach.

On the other hand, end-to-end training approaches which directly model context independent phones are elegant, and greatly facilitate speech recognition training. Most do not require an

explicit alignment of transcriptions with the training data, and there are typically fewer hyper-parameters to tune. We show that sequence training in multilingual settings can create feature extractors, which can directly be ported to new languages using a linear transformation (on very limited data), or re-trained on more data, opening a door to end-to-end language universal speech recognition.

## 3.5 Related Work and Babel Dataset

Some of the early works in multilingual and crosslingual speech recognition involved the use of language independent features like articulatory features (Stuker et al., 2003) to train HMM based systems. Authors in (Burget et al., 2010) used subspace Gaussian mixture model to map phonemes of different languages together. Authors in (Schultz and Waibel, 2001) introduce the use of a shared phone set to build HMM based language independent acoustic models and show the adaptation of pre-existing models towards a new language.
With the on-set of deep learning the focus of the models shifted to learning features across languages which can be mapped to the same space (Stolcke et al., 2006; Ghoshal et al., 2013). Authors in (Swietojanski et al., 2012) looked at unsupervised pretraining on different languages for a cross lingual recognition. The dominant architecture for multilingual or crosslingual speech recognition has been the so-called "shared hidden layer" model, in which data is passed through a series of shared feed-forward layers, before being separated into multiple language-specific softmax layers, which are trained using cross-entropy (Scanzio et al., 2008; Veselỳ et al., 2012; Heigold et al., 2013). This architecture can also be used as a "bottleneck" feature extractor, from which "language independent" features are extracted, on top of which a target-language acoustic model can be built. Authors in (Grézl et al., 2014) showed that these multilingual models can be adapted to the specific language to improve performance further. The work by (Veselỳ et al., 2012; Grézl et al., 2011) presented bottleneck features for multilingual systems where they showed feature porting is possible and gave competitive results when compared to systems with monolingual features. Other approaches (Tong et al., 2017; Vu et al., 2014) constructed a shared language independent phone set, which could then also be adapted to the target language. Our proposed model is inspired by the former approach which tries to learn latent features by sharing hidden layers across languages.

Connectionist Temporal Classification (CTC, (Graves et al., 2006)) lends itself to low-resource multilingual experiments, because systems built on CTC tend to be significantly easier to train than those that have been trained using hidden Markov models (Miao et al.,

[1]This work used releases IARPA-babel105b-v0.4, IARPA-babel201b-v0.2b, IARPA-babel401b-v2.0b, IARPA-babel302b-v1.0a (these 4 languages will be called the "MLing" set), and IARPA-babel106b-v0.2g, IARPA-babel307b-v1.0b, IARPA-babel204b-v1.1b, IARPA-babel104b-v0.4bY (these 4 languages will be called the "BAB300" set), and IARPA-babel202b-v1.0d and IARPA-babel205b-v1.0 for testing.

2015, 2016). Müller et al. (2017) shows that multilingual CTC systems with shared phones can improve performance in a limited data setting. As per our knowledge there has not been any prior work that have looked into learning "bottleneck" like features for a CTC based model and seen how it performs multilingually and crosslingually with adaptation.

For this work we use several languages from IARPA's Babel[1] project to test our model. These are mostly telephony (8kHz) conversational speech data in a low resource language. These were accompanied by a lexicon and dictionary in Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) format. Table 3.3 summarizes the amount of training data in hours along with the number of phonemes (including the CTC blank symbol) present for the languages we used in our experiments on the "Full Language Pack" (FLP) condition.

Table 3.3: Overview of the FLP Babel Corpora used in this work.

| Subset | Language | # Phones + $\emptyset$ | Training Data |
|---|---|---|---|
| MLing | Turkish | 50 | 79 hrs |
| | Haitian | 40 | 67 hrs |
| | Kazakh | 70 | 39 hrs |
| | Mongolian | 61 | 46 hrs |
| Bab300 | Amharic | 67 | 43 hrs |
| | Tamil | 41 | 69 hrs |
| | Tagalog | 48 | 85 hrs |
| | Pashto | 54 | 78 hrs |
| For testing | Kurmanji | 45 | 42 hrs |
| | Swahili | 40 | 44 hrs |

Table 3.4: Word (% WER) and phoneme error rate (% PER) for each of the test languages, on the Babel conversational development test sets.

| Model | Kazakh | | Turkish | | Haitian | | Mongolian | |
|---|---|---|---|---|---|---|---|---|
| | WER | PER | WER | PER | WER | PER | WER | PER |
| Monolingual | 55.9 | 40.9 | 53.1 | 36.2 | 49.0 | 36.9 | 58.2 | 45.2 |
| Multilingual (MLing) | 53.2 | 36.5 | 52.8 | 34.4 | 47.8 | 34.9 | 55.9 | 41.1 |
| MLing & FineTuning (FT) | 50.6 | 35.1 | 49.0 | 32.2 | 46.6 | 33.2 | 53.4 | 39.6 |
| MLing + SWBD | 52.3 | 36.6 | 51.3 | 33.0 | 45.8 | 33.9 | 54.5 | 40.2 |
| MLing + SWBD & FT | **48.2** | **33.5** | **48.7** | **31.9** | **44.3** | **31.9** | **51.5** | **37.8** |

## 3.6 Multilingual CTC Model

A model trained with CTC loss is a sequence based model which automatically learns alignment between input and output by introducing an additional label called the blank symbol ($\emptyset$), which

corresponds to 'no output' prediction. Given a sequence of acoustic features $\mathbf{X} = (\mathbf{x_1}, \ldots, \mathbf{x_n})$ with the label sequence $\mathbf{z} = (\mathbf{z_1}, \ldots, \mathbf{z_u})$, the model tries to maximize the likelihood of all possible CTC paths $\mathbf{p} = (\mathbf{p_1}, \ldots, \mathbf{p_n})$ which lead to the correct label sequence $\mathbf{z}$ after reduction. A reduced CTC path is obtained by grouping the duplicates and removing the $\emptyset$ (e.g. $\mathcal{B}(AA\emptyset AABBC) = AABC$).

$$P(\mathbf{z}|\mathbf{X}) = \sum_{\mathbf{p} \in \texttt{CTC\_Path(z)}} P(\mathbf{p}|\mathbf{X})$$

Like in (Miao et al., 2015) we use this loss along with stacked Bidirectional LSTM layers to encode the acoustic information and make frame-wise predictions.
In our CTC multilingual model, we share the bidirectional LSTM encoding layer till the final layer and project the learned embedding layer to the phones of the respective target languages. The intuition behind this model is that training on more than one language will help in better regularization of weights and learning a better representation of features, as it will be trained on more data. We hypothesize that the final phoneme discrimination can be learned in a linear projection of the last layer. Figure 3.1 shows the schematic diagram of our multilingual model. Mathematically this can be written as,

$$\mathbf{X} = \{\mathbf{X}_{L1} \cup \mathbf{X}_{L2} \cup \mathbf{X}_{L3} \ldots \mathbf{X}_{Ln}\} \qquad \mathbf{X}_{Li} = (x^1_{Li}, \ldots, x^n_{Li})$$
$$\mathbf{e} = \texttt{Encoder}_{BiLSTM}(X) \qquad \mathbf{e} \in \mathbb{R}^{n \times 2 * h_{dim}}$$

$$P(\mathbf{p}|\mathbf{X}) = \begin{cases} \texttt{softmax}(\mathbf{W}_{L1}\mathbf{e} + \mathbf{b}_{L1}) & \text{if } \mathbf{X} \in \mathbf{X}_{L1} \\ \texttt{softmax}(\mathbf{W}_{L2}\mathbf{e} + \mathbf{b}_{L2}) & \text{if } \mathbf{X} \in \mathbf{X}_{L2} \\ \ldots & \\ \texttt{softmax}(\mathbf{W}_{Ln}\mathbf{e} + \mathbf{b}_{Ln}) & \text{if } \mathbf{X} \in \mathbf{X}_{Ln} \end{cases}$$

Unlike (Veselỳ et al., 2012), we do not have any bottleneck layer, and the whole model is sequence trained based on CTC loss.

## 3.7 Experiments and Observations

### 3.7.1 Multilingual CTC model

To align with project goals, we chose to perform experiments on a set of four languages which are the closest/ have maximum phone overlap with Kurmanji – Kazakh, Turkish, Mongolian and Haitian. We used a 6-layer bidirectional LSTM network with 360 cells in each direction,

which performed best on average across the majority of Babel languages in a systematic search experiment. Table 3.4 shows the results. For consistency, we used absolutely identical settings across all languages, and did not perform any language-specific tuning, other than choosing the lowest perplexity language model between 3-gram and 4-gram models for WFST-based decoding. Techniques such as blank scaling and applying a softmax temperature can often improve results significantly, but we did not apply any of them here for consistency.

In our multilingual experiments, we use the same 6-layer Bi-LSTM network with 360 cells (per direction) in each layer as our shared encoded representation[2]. Again, this setup performed best on average on a larger set of languages. Multilingual training on the "MLing" set (the four languages shown in Table 3.4) improves WER by 1.7% (absolute) on average, while keeping the LSTM layers shared across all languages. If we fine-tune the entire model towards each language specifically, performance improves further, by 4.4% on average over the baseline. If we roughly double the amount of training data by adding the Switchboard 300h training set to the "MLing" training data, performance improves yet again, for both the universal (MLing+SBWD) and language-specific (MLing+SWBD & FT) case. Overall, WER and PER improve by about 6% absolute (>10% relative), which is in line with other results reported on comparable tasks discussed in section 3.5.

As expected, reductions in the error rates tend to be higher for the lower resource languages, like Kazakh and Mongolian.

### 3.7.2 Data Selection

Given that adding a seemingly unrelated, but high resource language improved the performance of the model on four low resource languages, we further studied the impact of varying the source(s) of the extra data. Specifically, we replaced the 300 h Switchboard corpus with four more unrelated Babel languages, "BAB300" composed of Tamil, Amharic, Pashto, and Tagalog. The results on the test data are summarized in Table 3.5. We can see that adding Switchboard data outperforms adding more unrelated Babel languages.

Table 3.5: Word error rate (% WER) on the test languages when switching the SWBD data with 300 hrs equivalent of Babel.

| Model | Kazakh | Turkish | Haitian | Mongolian |
|---|---|---|---|---|
| MLing + BAB300 | 57.5 | 52.0 | 47.8 | 56.7 |
| MLing + SWBD | **52.3** | **51.3** | **45.8** | **54.5** |

While our main goal here has been the creation of a multilingual recognizer, we verified that models that have been trained on a single Babel language plus 300 h of Switchboard do

[2]The code to train the multilingual model is released as part of EESEN (Miao et al., 2015).

not outperform the fine-tuned MLing+SWBD system, while there is no clear pattern on other languages. This indicates that it is generally beneficial to train (sequence-based) multilingual systems on closely related languages, and/or on large amounts of well-prepared but unrelated monolingual data, but that adding a large number of languages may in fact prevent the model from training well.

### 3.7.3 Representation Learning

In order to study to what extent the CTC sequence models have learned useful bottleneck like discriminatory audio features that are independent of the input language, we attempt to port a model to an unseen language. We aim to use the trained model as a language-independent feature extractor that can linearly separate any language into a phoneme sequence. To do this, we replace the softmax layer (or "layers" in the multilingual case) of a "donor" CTC model with a single softmax, which we then train with varying amounts of data from the target language, Kurmanji in our case. Figure 3.2 shows how different "donor" models behave in this situation. In the crosslingual case, it becomes beneficial to train the LSTM layers with as many different languages as possible ("MLing+BAB300" outperforms "MLing+SWBD" and "MLing"), while a single related language (Turkish) outperforms adaptation on a larger amount of data from an unrelated language (SWBD). There is a large gap between monolingual systems and multilingual systems. Improvements become smaller once training is performed on 4 h (10%) of data or more, but even then the re-estimation of the softmax layer (with ca. 32k parameters) benefits from more data.

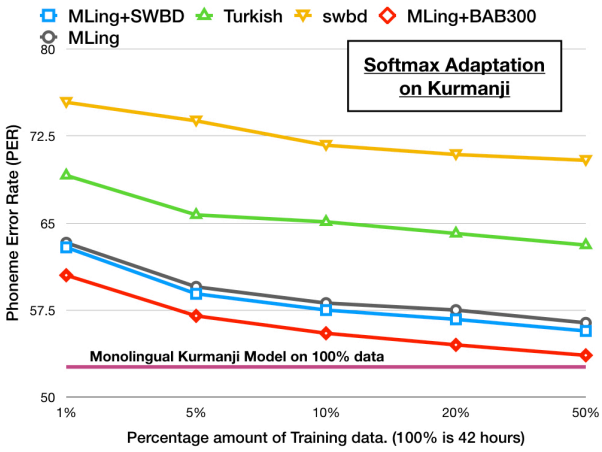
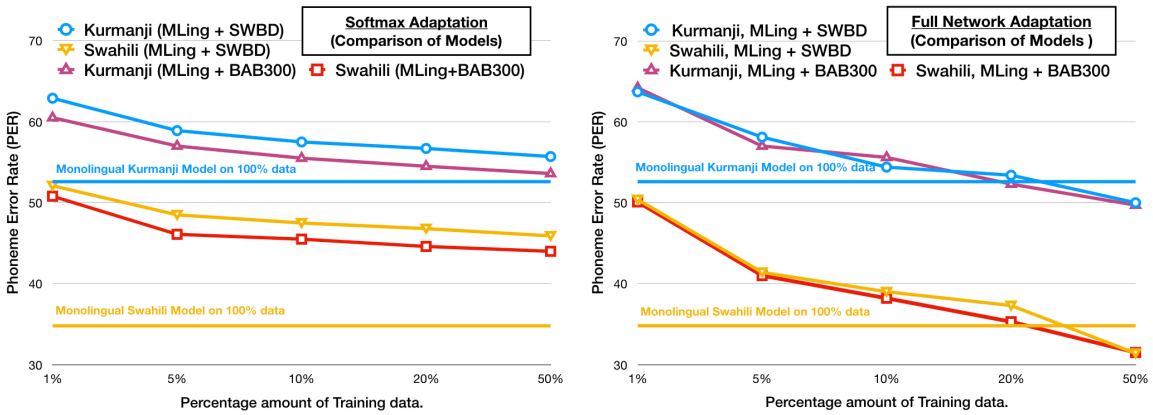



Figure 3.2: Crosslingual training of CTC softmax layer only on top of different "donor" models.

It thus seems that multilingual systems do indeed learn a portable, language independent representation, which is useful when porting to a new language, while the sheer amount of data is less beneficial.




(a) Adaptation of softmax layer only for Kurmanji and Swahili targets. Kurmanji performs well, because the language is similar to some training languages.

(b) Adaptation of entire network (re-training) to target languages. This outperforms softmax adaptation (on the left) as soon as 2-4 h of data become available.

Figure 3.3: Crosslingual training of Kurmanji and Swahili systems.

### 3.7.4 Crosslingual Explorations

Figure 3.4 shows that for both related and unrelated languages, a multilingual system surpasses the monolingual baseline once about 25% of the original data has been seen. The behavior of retraining ("full network adaptation") seems independent of the original trained languages.

To further investigate how multilingual models can be used in crosslingual settings, and with varying amounts of training data, we compare "softmax" adaptation and full network adaptation (retraining) on Kurmanji and Swahili, two languages which we did not see in training. We use the (MLing + SWBD) and (MLing + BAB300) "donor" models. Figure 3.3 shows that for small amounts of adaptation data, and a target language that is related to the pre-trained languages (Kurmanji), "softmax adaptation" is competitive, and an initialization with many languages is beneficial.

When the entire network can be retrained ("full network adaptation", shown on the right side of Figure 3.3), there is very little difference between the "donor" systems' performance.

### 3.7.5 Extending this approach with transformer based CTC models

The above study was done in 2018 and since then there have been many advances in the community which help make better speech recognition models. In particular, Transformers (Vaswani et al., 2017) and SpecAugment (Park et al., 2019) have led to better encoders and training stability respectively. In this section, we apply the above approach, but replacing LSTMs with Transformers and applying SpecAugmentation during training. We also added more languages and build a bigger model. The results are presented in Table 3.6.

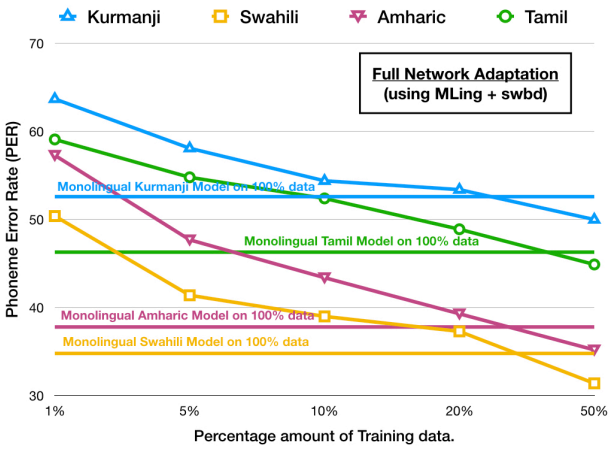



Figure 3.4: PER on different amounts of crosslingual data using a full network end-to-end adaptation (retraining).

Table 3.6: Results presenting the techniques developed in this chapter with the more recent Transformer encoders and SpecAugmentation. The table shows the phoneme error rate (% PER) for each of the test languages, on the Babel conversational development test sets. We have added the monolingual CTC phoneme error rates for comparison.

| | | Phoneme Error Rate % | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Model Type | Model Name | Eng | Tur | Tgl | Vie | Kaz | Amh | Jav | Total |
| LSTM-CTC | Monolingual-CTC (Mortensen et al., 2018) | - | 34.4 | 32.0 | - | 37.8 | 33.3 | 42.1 | 35.9 |
| Transformer-CTC | Multilingual-CTC (Dalmia et al., 2018) | 25.3 | 27.7 | 28.5 | 31.9 | 31.5 | 28.6 | 35.2 | **29.8** |

We use the English LDC Switchboard Dataset (Godfrey and Holliman, 1993; Consortium, 2002a,b) and 6 languages from the IARPA BABEL Program: Turkish, Tagalog, Vietnamese, Kazakh, Amharic and Javanese.[3] These datasets contain 8kHz recordings of conversational speech each containing around 50 to 80 hours of training data, with an exception of around 300 hours for English.

The model was trained using the ESPnet toolkit (Watanabe et al., 2018). To prepare our speech input features we first upsample the audio to 16kHz, augment it by applying a speed perturbation of 0.9 and 1.1, and then extract global mean-variance normalized 83 log-mel filterbank and pitch features. Input frames are processed by an audio encoder with convolutional blocks to subsample by 4 (Watanabe et al., 2018) before feeding to 12 transformer-encoder blocks with a feed-forward dim of 2048, attention dim of 256, and 4 attention heads. We augment our data with the Switchboard Strong (SS) augmentation policy of SpecAugment (Park et al., 2019) and apply a dropout of 0.1 for the entire network. We use the Adam optimizer to train 100 epochs

[3]We use the Full Language Packs (FLP) released by the IARPA Babel Research Program (IARPA-BAA-11-02): IARPA-babel105b-v0.4, IARPA-babel106-v0.2g, IARPA-babel107b-v0.7, IARPAbabel302b-v1.0a, IARPAbabel307b-v1.0b, IARPAbabel402b-v1.0b

with an inverse square root decay schedule, a transformer-lr scale (Watanabe et al., 2018) of 5, 25k warmup steps, and an effective batchsize of 768.

## 3.8 Conclusion

In this chapter, we present the CTC hybrid models which decompose a sequence task into alignment using the CTC loss and language generation using a language model. In order to show the resource pooling aspects for the CTC component of the hybrid model, we demonstrate that it is possible to train multilingual and crosslingual acoustic models by decomposing words from different languages into universal phone sequences. In multilingual settings, it seems beneficial to train on related languages only, or on large amounts of clean data; there is no benefit simply from training on many languages. It is thus possible to combine e.g. Switchboard and Babel data. In very low resource crosslingual scenarios, it is possible to adapt a model to a previously unseen language by re-training the softmax layer only. CTC models can learn a language independent representation at the input to the softmax layer. We find that training the models trained on related languages help, as does training on many languages, rather than large amounts of data. As more and more data is available, and the whole network can be retrained, and the effect of the choice of language for the multilingual training disappears.

# Chapter 4

# Modular Encoder-Decoder Systems

In Chapter 3, we build sequence prediction models using CTC Hybrid Systems that decompose the overall task into alignment using the CTC encoder and language generation using a language model. These systems exhibit all the properties of compositionality but suffer from the inability to tune the model on end task data.

This chapter aims to build an end-to-end alternative based on the encoder-decoder architecture (Bahdanau et al., 2015; Chan et al., 2016) that is also compositional in nature. We present LegoNN a framework for building modular encoder-decoder models that defines a clear interface between the encoder and decoder modules and allows re-using decoder modules trained from different seeds, architectures, or even different tasks into an encoder module that is trained separately. These models also have the ability to further fine-tune the model on end-to-end task data.

## 4.1 LegoNN: Building Modular Encoder-Decoder Models

State-of-the-art encoder-decoder models (e.g. for machine translation (MT) or speech recognition (ASR)) are constructed and trained end-to-end as an atomic unit. No component of the model can be (re-)used without the others. We describe LegoNN, a procedure for building encoder-decoder architectures with decoder modules that can be reused across various MT and ASR tasks, without the need for any fine-tuning. To achieve reusability, the interface between each encoder and decoder modules is grounded to a sequence of marginal distributions over a discrete vocabulary pre-defined by the model designer. We present two approaches for ingesting these marginals; one is differentiable, allowing the flow of gradients across the entire network, and the other is gradient-isolating. To enable portability of decoder modules between MT tasks for different source languages and across other tasks like ASR, we introduce a modality agnostic encoder which consists of a length control mechanism to dynamically adapt encoders' output lengths in order to match the expected input length range of pre-trained decoders. We present several experiments to demonstrate the effectiveness of LegoNN models: a trained language generation

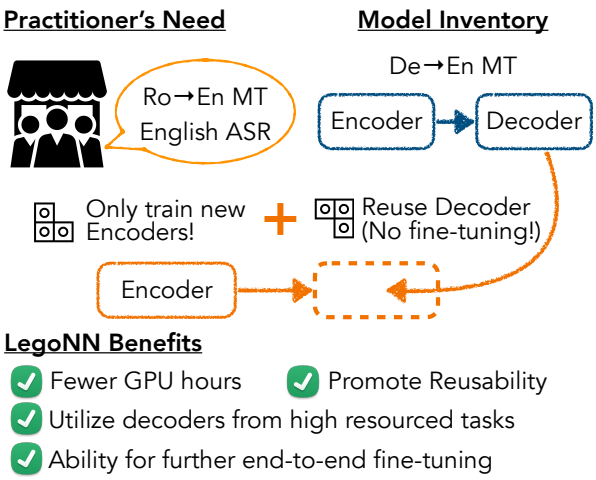



Figure 4.1: **LegoNN Framework**: Building encoder-decoder models in the LegoNN framework, allows practitioners to reuse components like decoder modules for various sequence prediction tasks. For example, in this figure an English predicting decoder from German-English machine translation system can be re-used for both Romanian-English machine translation and English speech recognition without any fine-tuning steps. This saves overall compute resources, promotes re-usability, allows practitioners to utilize decoders from high-resourced tasks for under-resourced ones. The composed model is end-to-end differentiable leaving room for further improvements through fine-tuning.

LegoNN decoder module from German-English (De-En) MT task can be reused with no fine-tuning for the Europarl English ASR and the Romanian-English (Ro-En) MT tasks to match or beat respective baseline models. When fine-tuned towards the target task for few thousand updates, our LegoNN models improved the Ro-En MT task by 1.5 BLEU points, and achieved 12.5% relative WER reduction for the Europarl ASR task. Furthermore, to show its extensibility, we compose a LegoNN ASR model from three modules – each has been learned within different end-to-end trained models on three different datasets – boosting the WER reduction to 19.5%.

## 4.2 Introduction

Training end-to-end models for machine translation (MT) or automatic speech recognition (ASR) requires the learning of multiple implicit functions (Bahdanau et al., 2015; Sutskever et al., 2014; Vaswani et al., 2017; Chan et al., 2016; Bahdanau et al., 2016). An MT model implicitly does both word translation and language generation, while an ASR model combines phoneme recognition, pronunciation modeling, and language generation. These fully differentiable models are conceptually simple and work well in practice. However, they forgo opportunities to share common logical functions between different tasks, such as their decoders. This leads to wasted compute during training and less interpretable architectures overall.

Motivated by modularity principles in software design (Baldwin and Clark, 1999) where modules have interpretable interfaces and are reusable within other programs, we introduce LegoNN, a procedure for constructing encoder-decoder models with decoder modules that can be reused across various sequence generation tasks such as MT, ASR, or Optical Character Recognition (OCR). As summarized in Figure 4.1, for AI models, enforcing modularity helps save computing resources by reusing components and helps build systems for under-resourced tasks by utilizing shareable components from higher-resourced tasks. Additionally, having interpretable interfaces enables monitoring the performance of individual encoder or decoder modules and their contributions to the overall end-to-end performance.

More concretely, in our LegoNN encoder-decoder framework, encoders have an interpretable interface by outputing a sequence of distributions over a discrete vocabulary, derived from the final output labels (e.g. phonemes or sub-words). During training, we add an additional Connectionist Temporal Classification (CTC) loss (Graves et al., 2006) to the encoder output to enforce this modularity (§4.4.1). Our decoder modules are extended with Ingestor layers (ING) that accept these distributions as input (§4.4.4). We experiment with two types of ingestor layer: a differentiable Weighted Embedding (WEmb) ingestor allowing gradient flow across the entire network and a gradient-isolating Beam Convolution (BeamConv) one. Given that LegoNN decoder modules can be trained for one MT task and then reused for another MT task with a different source language or for a sequence generation task with the same target language such as ASR or OCR, we propose a modality agnostic encoder for a sequence prediction task that uses an output length controller (OLC) unit to adapt any input modality to a sequence of encoder representations that matches the expected input length of another (§4.4.3). LegoNNs, as also demonstrated in our experiments, enable the sharing of trained decoders [1] and intermediate modules between different tasks and domains without jointly training for both tasks and no fine-tuning steps. The composed LegoNN model preserves end-to-end differentiability of each individual system, allowing room for further improvements through fine-tuning.

Our experiments show that we can achieve modularity without sacrificing performance. On the standard large-scale En-De WMT and Switchboard ASR benchmarks, LegoNN models reach levels of performance competitive with non-modular architectures (§4.6.1), while still passing our stress tests for testing modularity (§4.6.2). We show the value of modularity by seamlessly composing a decoder module trained within a German-English (De-En) WMT system with other pre-trained encoder modules from different MT and ASR systems, without any joint training or fine-tuning, to match or beat generation performance for the Europarl English ASR task and the Romanian-English (Ro-En) WMT task (§4.6.3). When such composed LegoNN models are fine-tuned for few thousand steps towards the target domain, they improve the Ro-En MT task by 1.5 BLEU points and improve the Europarl ASR task by 12.5% WER relative to the baseline system

[1] Our work on LegoNNs are orthogonal to the recent work on sharing pre-trained encoders, e.g BERT (Devlin et al., 2019). Combining the benefits of these two complementary approaches is left for future work.

(§4.6.4). To demonstrate the flexibility of reusing LegoNN modules, we construct an ASR system which is composed of modules that have been trained independently on three different tasks, without performing any fine-tuning and with almost no performance degradation. Modules are: (1) a phoneme recognizer from the Europarl ASR model, (2) a pronunciation model from the TED-LIUM ASR model, and (3) a language generation decoder from the WMT model (§4.6.3). With a few end-to-end fine-tuning steps, the composed model beats the Europarl ASR baseline model by 19.5% relative WER, also improving over the previously composed LegoNN model for this task (§4.6.4).

## 4.3 Background

### 4.3.1 Cross-Entropy Loss in Encoder-Decoder Models

Given an input sequence $X_{1:T}$, encoder-decoder models (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) compute an output sequence $L_{1:N}$ by factorizing the probability of the joint target sequence into a product of auto-regressively generated outputs conditioned on previously generated tokens and the input sequence. They are trained by minimizing the token-level cross-entropy ($\mathcal{F}_{\text{CE}}$) loss between the true tokens ($L$) and the predicted distributions of the decoder ($\mathbb{P}^{\mathcal{Y}_{\text{Attn}}}$) over the set of output tokens $\mathcal{Y}_{\text{Attn}}$, where $L_n \in \mathcal{Y}_{\text{Attn}}$:

$$\mathbf{h}^{E}_{1:T} = \text{encoder}(X_{1:T}) \tag{4.1}$$

$$\mathbb{P}^{\mathcal{Y}_{\text{Attn}}}_{n} = \text{softmax}(\text{decoder}(\mathbf{h}^{E}_{1:T}, L_{1:n\text{-}1})) \tag{4.2}$$

$$\mathcal{F}_{\text{CE}}(L_{1:N}, \mathbb{P}^{\mathcal{Y}_{\text{Attn}}}_{1:N}) = -\log\left(\prod_{n=1}^{N} \mathbb{P}^{\mathcal{Y}_{\text{Attn}}}_{n,L_n}\right) \tag{4.3}$$

where the encoder processes the input sequence ($X_{1:T}$) and communicates it to the decoder through an attention mechanism (Bahdanau et al., 2015; Chan et al., 2016; Luong et al., 2015), which is conditioned on the decoder state. The encoder-decoder model thereby models the likelihood for next token prediction given the previous output tokens and the input sequence:

$$\mathbb{P}^{\mathcal{Y}_{\text{Attn}}}_{n} = P(y^{\text{Attn}}_{n} \mid X_{1:T}, L_{1:n\text{-}1}) \tag{4.4}$$

where $\mathbb{P}^{\mathcal{Y}_{\text{Attn}}}_{n}$ is the distribution of the random variable $y^{\text{Attn}}_{n}$ for position $n$, $\mathbb{P}^{\mathcal{Y}_{\text{Attn}}}_{n}$, over the output vocabulary space.

### 4.3.2 Connectionist Temporal Classification Loss

Rather than producing a soft alignment between the input and target sequences, the CTC loss (Graves et al., 2006) maximizes the log conditional likelihood of the output, by integrating over all possible monotonic alignments between both sequences:

$$\mathbb{P}_{1:T}^{\mathcal{Y}_{\text{CTC}}} = \text{softmax}\left(\text{encoder}(X_{1:T}) * W_o\right) \tag{4.5}$$

$$\mathcal{F}_{\text{CTC}}(L_{1:N}, \mathbb{P}_{1:T}^{\mathcal{Y}_{\text{CTC}}}) = -\log \sum_{z \in \mathcal{Z}(L,T)} \left( \prod_{t=1}^{T} \mathbb{P}_{t,z_t}^{\mathcal{Y}_{\text{CTC}}} \right) \tag{4.6}$$

$W_o \in \mathbb{R}^{d \times |\mathcal{Y}_{\text{CTC}}|}$ projects the encoder representations into the output vocabulary space of $\mathcal{Y}_{\text{CTC}}$. $\mathcal{Z}(L,T) \in \{(t,l) | t \in \{1:T\}, l \in \mathcal{Y}_{\text{CTC}}\}$ is the space of all possible monotonic alignments of $L$ into $T$ time steps, and the probability of an alignment $z$ is the product of locally normalized output probabilities per time step $\mathbb{P}_{1:T}^{\mathcal{Y}_{\text{CTC}}} \in \mathbb{R}^{T \times |\mathcal{Y}_{\text{CTC}}|}$. We omit the extra CTC blank symbol in the equation above for clarity of presentation; see (Graves et al., 2006) for more details. The marginalization sum is efficiently computed during training using dynamic programming.

## 4.4 LegoNN for Modular Encoder-Decoder Models

We propose decomposing encoder-decoder models into one (or more) encoder modules followed by an auto-regressive decoder module, which can be reused for other tasks and domain, without sacrificing its end-to-end differentiability. Each module produces a sequence of marginal distributions over a pre-defined discrete vocabulary, which is consumed by an ingestor component in the subsequent module. LegoNN modules trained either jointly or independently can be reused for new tasks without the need of any fine-tuning. The LegoNN procedure introduces three operations:

### 4.4.1 Designer-defined module interface

We look at encoder-decoder models as full software programs that execute the specific function of mapping one sequence of input symbols or vectors to another. Although some components of software programs, i.e. libraries, may be reused for numerous future programs, *trained* decoder components of the traditional encoder-decoder models (Sutskever et al., 2014; Bahdanau et al., 2015) are not designed to be reused independently with other models or tasks. A well-defined and abstract input/output interfaces are prerequisites for developing reusable software libraries; however, encoder-decoder models fall short in this respect. To convert decoders into reusable modules, the interface connecting them to encoders must: (1) be interpretable, and (2) use a vocabulary abstraction that generalizes to future tasks.

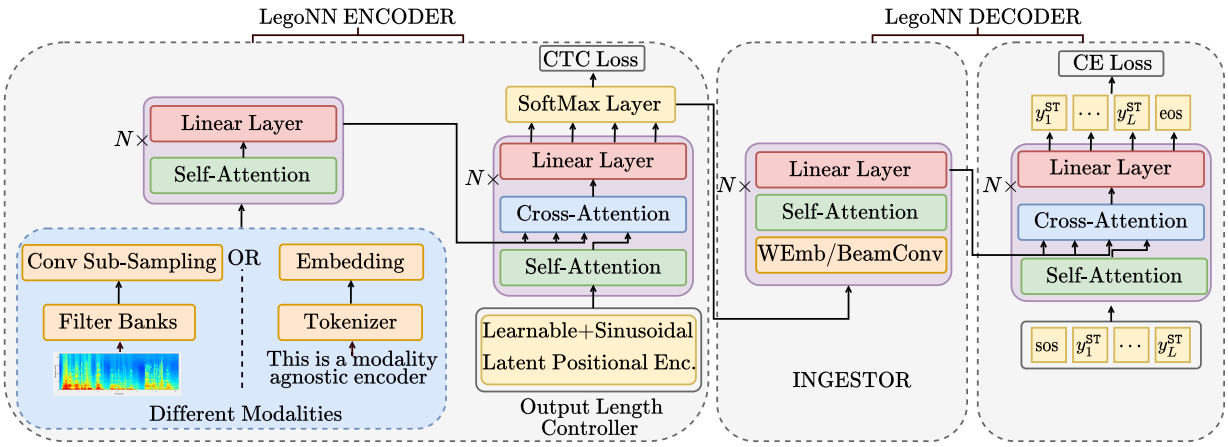



Figure 4.2: **LegoNN Encoder-Decoder Model**: This figure presents the schematics and the information flow in our LegoNN encoder-decoder model. LegoNN decomposes encoder-decoder models into reusable modules by: (1) grounding module output into an interpretable vocabulary; (2) adding an ingestor (ING) to process input marginal distributions; and (3) an output length controller (OLC) to match the input length of subsequent modules. Blocks with lighter colors in the figure show reused modules between the MT and ASR tasks without gradient updates.

#### 4.4.1.1 Interpretable interfaces between encoders and decoders

Encoders communicate with decoders through a sequence of continuous hidden representations that are by-products of the end-to-end model optimization process. These hidden vectors are uninterpretable and have a completely different meaning for different random seeds and training hyper-parameters even when the same task, model, and training data are used. As we see from Equation 4.2, although the decoder is conditioned on encoder states ($\mathbf{h}^{E}_{1:T}$) the encoder-decoder still directly models the conditional next word prediction over the input (Equation 4.4) as these encoder states have no physical meaning.

To enable reusability between different tasks and models, LegoNN grounds encoder outputs into a discrete vocabulary that is pre-defined by the model designer. For applications such as ASR and MT, such intermediate discrete vocabulary can be defined over phonemes, or some byte-pair encoding (BPE) dictionary (Sennrich et al., 2016; Kudo and Richardson, 2018) driven from the task labels. The chosen intermediate vocabulary is not necessarily the same as the model's output dictionary.

To enforce the desired encoder vocabulary during learning, the decoder token-level cross-entropy loss is combined with the supervised CTC loss which is applied to the encoder position-wise, locally-normalized output distributions. Having the CTC loss at the encoder output doesn't prevent the flow of gradients between the decoder and encoder modules. We chose the CTC loss because of its suitability for sequence prediction tasks, which are the focus of the proposed LegoNN procedure. Following the notation introduced in §4.3, a LegoNN encoder-decoder

model effectively models the likelihood for next token prediction given the previous output tokens ($L_{1:n\text{-}1}$) and the encoder states ($\mathbb{P}^{\mathcal{Y}_{\text{CTC}}}_{1:T}$):

$$\mathbb{P}^{\mathcal{Y}_{\text{Attn}}}_{n} = P(y^{\text{Attn}}_{n} \mid \mathbb{P}^{\mathcal{Y}_{\text{CTC}}}_{1:T}, L_{1:n\text{-}1}) \tag{4.7}$$

Note how the conditional is different from Equation 4.4. Here, the encoder states ($\mathbb{P}^{\mathcal{Y}_{\text{CTC}}}_{1:T}$) are not simply transformations of the input ($X_{1:T}$), but are distributions being modeled by the CTC loss given the input:

$$\mathbb{P}^{\mathcal{Y}_{\text{CTC}}}_{1:T} = P(y^{\text{CTC}}_{1:T} \mid X_{1:T}) \tag{4.8}$$

This modeling framework allows the decoder module to accept any encoder that produces the same distribution, $\mathbb{P}^{\mathcal{Y}_{\text{CTC}}}_{1:T}$, as the decoder was trained toward and hence building an interpretable interface to enforce modularity between the encoder-decoder modules. The above modeling also gives the encoder and decoder modules in the LegoNN modular framework specific functionalities towards the target task. For example, an MT encoder module is not expected to solve the overall translation task but rather acts as a word/phrase translation component or an ASR encoder module acts as a phoneme/sub-word recognizer whose outputs are later refined using an auto-regressive decoder.

A LegoNN model is not restricted to contain only two modules; an encoder and a decoder. There may be a sequence of encoder modules, each designed to perform a certain function through their respective pre-defined output vocabulary. The ASR system is one example whose encoder can be divided into two modules, a phoneme recognizer and a pronunciation model, followed by the auto-regressive language generating module. In this case, the CTC loss is applied more than once. During learning, LegoNN models optimize a joint loss over each module:

$$\mathcal{F}_{\text{obj}} = \mathcal{F}_{\text{CE}}(L^{\mathcal{Y}^M}, \mathbb{P}^{\mathcal{Y}^M_{\text{Attn}}}) + \sum_{i=1}^{M-1} \mathcal{F}_{\text{CTC}}(L^{\mathcal{Y}^i}, \mathbb{P}^{\mathcal{Y}^i_{\text{CTC}}}) \tag{4.9}$$

Where $M$ is the total number of modules in the LegoNN system, $\mathbb{P}^{\mathcal{Y}^i}$ is the prediction in module $i$, $L$ is the target sequence and $\mathcal{Y}^i$ is the pre-defined vocabulary for $L$ at the interface of module $i$. All modules in the LegoNN systems have a CTC loss at their output except the final auto-regressive decoder module with a token-level CE loss.

Furthermore, applying a supervised loss at the output of each module brings two extra benefits: (a) It extends our ability to evaluate and diagnose the performance on multiple points across the model. This can guide modeling decisions, e.g. adding more modeling capacity to one module over the other. (b) An encoder module may be trained in conjunction with its decoder or independently by itself. For an ASR task, this may be useful when getting access to more audio training data or adapting the system to new acoustic condition where re-training the decoder would be wasteful.

#### 4.4.1.2 Defining vocabularies that generalize across tasks

As with software libraries, designing a module interface with the right level of generality is challenging, but enables greater reuse of modules across tasks and domains. To reuse a decoder module from an MT LegoNN model to an ASR task, the output vocabulary of the speech encoder must be compatible with the input vocabulary of the translation decoder. This is true even within a single task – for example, in ASR, a vocabulary of phonemes developed for a phoneme recognizer module in a read speech dataset, e.g. audio books, may not be the best one for spontaneous conversational situations, which is full of hesitations and false starts. We propose designing a shared vocabulary by combining target units of multiple potential future tasks and finding a vocabulary at their intersection.

### 4.4.2 LegoNN Encoder-Decoder model

Figure 4.2 provides the schematics of our proposed LegoNN Encoder-Decoder model, which follows the modeling framework described in the section above. The model is designed to work for various sequence prediction tasks with various input modalities such as speech or text. The designed LegoNN modules take into account reusing LegoNN decoder modules across different tasks such as MT and ASR. For this purpose, we designed a modality agnostic encoder with an output length controller unit (§4.4.3), which we call the LegoNN encoder as shown in the left side of Figure 4.2. The LegoNN decoder, as shown on the right side of Figure 4.2, is modified with an added ingestor component (§4.4.4) that would consume the distributions produced by the LegoNN encoder. We have detailed the individual modules in the following sections.

### 4.4.3 LegoNN Encoder

For an input $X_{1:T}$, which can either be filter banks for speech frames or embeddings for text tokens, the LegoNN encoder consists of two sets of repeating blocks. The first set of repeating blocks, like the transformer encoder blocks (Vaswani et al., 2017) simply encodes the input context through repeating blocks of multi-head self-attention, $\text{MHA}(x, x, x)$, and position-wise feedforward layers, $\text{FFN}(x)$.

$$\tilde{X} = \text{LayerNorm}(X); \quad X = X + \text{MHA}(\tilde{X}, \tilde{X}, \tilde{X}) \tag{4.10}$$

$$X_{1:T} = X + \text{FFN}(\text{LayerNorm}(X)) \tag{4.11}$$

These blocks are followed by a final $\text{LayerNorm}(X)$ that returns an input context-aware representations of length $T$. Since these lengths can be quite different for different modalities like speech and text, we add a second set of repeating blocks called the output length controller unit.

#### 4.4.3.1 Ouput Length Controller (OLC) unit

One of the challenges of using modules across sequential tasks, where inputs and outputs have different lengths, is adapting the output length of an encoder module trained on one task to match the expected input length of another. For example, encoder modules from an ASR task encode inputs in more time steps compared to an MT encoder. Naive up- or down-sampling approaches, e.g., pooling or replicating time-steps (Libovický and Helcl, 2018), cover only integer length ratios, which is either aggressively down-sampling or unnecessarily up-sampling output sequence lengths. To solve this problem, we introduce an Output Length Controller (OLC) component in LegoNN encoders to enable working with fractional length ratios between inputs and outputs of the same module.

OLC is a novel application of cross-attention (Bahdanau et al., 2015) between two groups of transformer layers in a multi-layer module. Let a layer $l$ be the last one to process an input sequence of length $T$, $X_{1:T}$, which we want to convert into $K$ length. The OLC first initializes a sequence of positional embeddings $K$, $\mathbf{h}^{\text{PE}}_{1:K}$,

$$\mathbf{h}^{\text{PE}}_{1:K} = \text{SinusoidalPE}(1:K) + \text{LearnablePE}(1:K) \tag{4.12}$$

where LearnablePE and SinusoidalPE are learnable and sinusoidal positional embeddings respectively (Gehring et al., 2017; Vaswani et al., 2017).

The $\mathbf{h}^{\text{PE}}_{1:K}$ representations then pass through the second set of repeating transformer blocks, which applies cross attention to the learned representations from the first set of transformer blocks, $X_{1:T}$. These repeating blocks consist of multihead self-attention, $\text{MHA}(y, y, y)$, multiheaded cross-attention, $\text{MHA}(y, x, x)$, and position-wise feedforward layers, $\text{FFN}(y)$.

$$\tilde{\mathbf{h}}^{\text{PE}} = \text{LayerNorm}(\mathbf{h}^{\text{PE}}) \tag{4.13}$$
$$\mathbf{h}^{\text{PE}} = \mathbf{h}^{\text{PE}} + \text{MHA}(\tilde{\mathbf{h}}^{\text{PE}}, \tilde{\mathbf{h}}^{\text{PE}}, \tilde{\mathbf{h}}^{\text{PE}}) \tag{4.14}$$
$$\mathbf{h}^{\text{PE}} = \mathbf{h}^{\text{PE}} + \text{MHA}(\text{LayerNorm}(\mathbf{h}^{\text{PE}}), X, X) \tag{4.15}$$
$$\mathbf{h}^{\text{PE}}_{1:K} = \mathbf{h}^{\text{PE}} + \text{FFN}(\text{LayerNorm}(\mathbf{h}^{\text{PE}})) \tag{4.16}$$

These blocks are followed by a final $\text{LayerNorm}(\mathbf{h}^{\text{PE}})$, which now returns a sequence of $K$ representations encoding the input representations of length $T$. These $K$ can be either up-sampling or down-sampling depending on the input modality. Here we set $K$ as a factor for the input length that tries to match the length of the output length of the MT and ASR encoder.

#### 4.4.3.2 CTC Interface

Finally the encoder representations, $\mathbf{h}^{\text{PE}}_{1:K}$ is transformed into the encoder vocabulary size followed by a softmax operation, as shown in Equation 4.5, to produce a distribution ($\mathbb{P}^{\mathcal{Y}_{\text{CTC}}}_{1:K}$) over the

encoder vocabularies. These distributions are passed to the LegoNN decoder and to the CTC loss computation.

### 4.4.4 LegoNN Decoder with Ingestor of probability distributions

The LegoNN Decoder is the standard transformer decoder (Vaswani et al., 2017) with an added Ingestor (ING) component to consume input marginal distributions from preceding encoder modules. We propose two ingestor architectures; one that is differentiable, allowing for communicating gradients between modules, and another discrete one that communicates a ranked list of hypotheses while keeping the modules gradient-isolated. These ingestor components can be added on top of any module that accepts a distribution, so in a multi-module system (Equation 4.9) these can be part of a legoNN Encoder.

#### 4.4.4.1 A Weighted Embedding Ingestor (WEmb)

The weighted embedding ingestor (WEmb) computes an expected embedding vector ($\mathbf{h}$) of the encoder distributions ($\mathbb{P}^{\mathcal{Y}_{\text{CTC}}}_{1:K}$) per output time-step. Since these embedding vectors are formed out of a local normalized conditionally independent CTC distributions, we need to re-encode the positional information in these embeddings. We combine the expected embedding vector ($\mathbf{h}$) with sinusoidal positional embedding (PE) before applying a few layers of self-attention transformer encoder blocks (Vaswani et al., 2017) (Equation 4.10-4.11) to aggregate information across time-steps.

$$\mathbf{h} = \mathbb{P}^{\mathcal{Y}^{i-1}_{\text{CTC}}}_{1:K} * W_{\text{Emb}}; \quad \mathbf{h} = \mathbf{h} + \text{PE}(\mathbf{h}) \tag{4.17}$$

$$\mathbf{h} = \text{TransformerEncoder}(\mathbf{h}) \tag{4.18}$$

where $\mathbb{P}^{\mathcal{Y}^{i-1}_{\text{CTC}}}_{1:K}$ is the output distribution of the previous module $i-1$, $W_{\text{Emb}} \in \mathbb{R}^{|\mathcal{Y}^{i-1}_{\text{CTC}}| \times d}$ and $d$ is the input dimension of the current module $i$, where a module denotes the modularity point as shown in Equation 4.9.

The first operation, to compute the expected embedding, is equivalent to a 1-D convolution operation with a receptive field RF=1. When extended to larger receptive fields, WEmb offers the opportunity to learn local confusion patterns of the previous module: $\mathbf{h} = \text{Conv1D}(\mathbb{P}^{\mathcal{Y}^{i-1}_{\text{CTC}}}_{1:K})$; with RF $\geq 1$.

#### 4.4.4.2 A Beam Convolution Ingestor (BeamConv)

Rather than using the full output probability values $\mathbb{P}^{\mathcal{Y}^{i-1}_{\text{CTC}}}_{1:K}$, the beam convolution ingestor (BeamConv) uses only the token indices of the top-p, per-position hypotheses, from the output of the preceding module. This creates an information bottleneck (Tishby et al., 1999) in the model

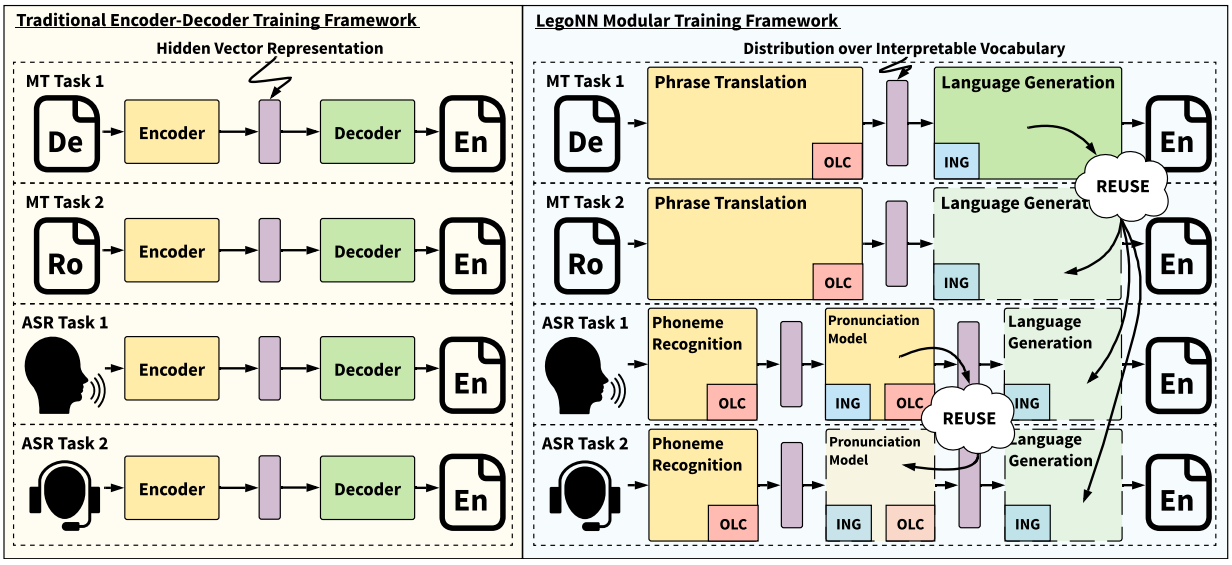



Figure 4.3: **Benefits of LegoNN**: Given a scenario where practioners have an De-En MT system and want to build additional ASR and MT systems, with the LegoNN framework, they only need to build new encoder systems and can directly re-use decoder modules from their inventory. For example, re-using De-En MT decoder for Ro-En MT task and English ASR task. Additionally, when building an ASR system on a different domain they can re-use components from both ASR and MT systems like the pronunciation module from the previous ASR system and decoder module from De-En MT system.

where gradients cannot be communicated.

$$\text{top-p}(\mathbb{P}_k^{\mathcal{Y}_{\text{CTC}}^{i-1}}) = \text{argmax}_{A \subset \mathbb{P}_k^{\mathcal{Y}_{\text{CTC}}^{i-1}}, |A|=p} \sum_{a \in A} a \tag{4.19}$$

where $\mathbb{P}_k^{\mathcal{Y}_{\text{CTC}}^{i-1}}$ is the output distribution at position $k$ for module $i-1$. The top-p indices are embedded into $d$ dimensional table, and, similar to the WEmb ingestor, we apply positional embedding and self-attention transformer encoder blocks. Furthermore, we can also use the 2-D convolution to aggregate local information.

$$\mathbf{r} = \text{Embedding}\left(\text{top-p}(\mathbb{P}_{1:K}^{\mathcal{Y}_{\text{CTC}}^{i-1}})\right); \quad \mathbf{h} = \text{Conv1D}(\mathbf{r}) \tag{4.20}$$

$$\mathbf{h} = \mathbf{h} + \text{PE}(\mathbf{h}); \quad \mathbf{h} = \text{TransformerEncoder}(\mathbf{h}) \tag{4.21}$$

where $\mathbf{r} \in \mathbb{R}^{T \times k \times d}$ when beam size $=k$ and RF $\geq 1$ for input of $T$ time steps.

### 4.4.5 LegoNN: Modularity Tests and Benefits of Modularity

To present the efficacy of LegoNN, we subject the LegoNN models to a variety of experiments. These tests are designed to show that LegoNNs do not compromise on performance, are modular, and flexible across tasks.

#### 4.4.5.1 Performance Tests

We show that we can build strong LegoNN models that are comparable to the baseline encoder decoder models across benchmark datasets for both speech recognition (ASR) and machine translation (MT).

#### 4.4.5.2 Modularity Tests

To show that LegoNN models are indeed modular, we subject LegoNN and the baseline encoder-decoder to various stress tests that check the modularity of these models. Similar to (Naik et al., 2018), these tests are designed to check if an encoder-decoder system is modular or not. We consider three basic tests:

1. *Random-Seed Swap* - Encoder or Decoder modules trained from a different random seed should have the same functionality and hence swappable.
2. *Architecture Swap* - Encoders or Decoders modules trained with different architectures should have the same functionality and hence swappable.
3. *Decoder Plug* - Encoders trained in isolation should be able to plug-in to Decoders from previously trained LegoNN models that have the same interface.

#### 4.4.5.3 Benefits of Modularity

In order to present the benefits of modularity, we consider a practical scenario where practitioners need to build multiple ASR and MT tasks, as shown in Figure 4.3. In a situation where practitioners have a high resourced De-En machine translation model in their inventory, we show that with LegoNN models they can save compute resources and leverage the well trained De-En Decoder by applying *Decoder Plug* to build systems for other MT and ASR tasks. In particular, we consider three scenarios:

1. *Romanian-English (Ro-En) MT* - In order to build a machine translation system for an under-resourced task such as Romanian-English MT. Practitioners using the LegoNN framework only need to build the Ro-En LegoNN encoder and re-use the De-En Decoder. They can also benefit the Ro-En MT task by using the De-En Decoder module trained on larger corpora.
2. *English ASR* - Practitioners can also use LegoNN decoders from an MT task for a different

task on a different modality. The OLC ensures that the expected encoder length for both MT and ASR task matches, thereby making the De-En Decoder re-usable.

3. *Different Domain English ASR* - We show that LegoNN modularity points are not limited to be between Encoder and Decoders. While building an English ASR system on a different domain, practitioners can re-use components from both the English ASR and De-En MT system.

All the composed LegoNN models are end-to-end differentiable allowing further fine-tuning to improve performance if the practitioner has additional compute resources available.

## 4.5 Experimental Setup

All our encoder-decoder models use the transformer architecture (Vaswani et al., 2017) implemented in the fairseq library (Ott et al., 2019) and run on DGX-1 nodes with 8 NVIDIA V100 GPUs. For both tasks, we apply LayerNorm (Ba et al., 2016) before every residual connection and a final one at the end of all the transformer blocks. We use Adam optimizer (Kingma and Ba, 2014) with eps $= 1e^{-9}$, betas $= (0.9, 0.999)$, label smoothing=0.1, and a gradient clip norm $= 5.0$. More details are provided regarding the data setup and a detailed model description is provided in Appendix A.1.1, A.1.2 and A.7.1.

### 4.5.1 Speech recognition task

*Data:* For our speech recognition experiments, we follow the standard 300 hours Switchboard (LDC97S62 (Godfrey and Holliman, 1993)) setup, and the Switchboard (SWB) and CallHome (CH) subsets of the HUB5 Eval2000 set (LDC2002S09 (Consortium, 2002b), LDC2002T43 (Consortium, 2002a)) for testing. We follow the data preparation setup of ESPNET (Watanabe et al., 2018), where we have 100 and 2000 target SentencePiece (Kudo and Richardson, 2018) units trained on the 300h text. We follow the same recipe for processing the TED-LIUM (Rousseau et al., 2014) and Europarl (Iranzo-Sánchez et al., 2020) data, with phonemes generated using (Park and Kim, 2019), detailed in Appendix A.1.2. We use the last model for inference with a beam size of 20 and length penalty of 1.0. We do not use an external LM or joint decoding over the encoder and decoder output (Watanabe et al., 2018). [0.05in] *Model architecture:* Input features are processed using two 2-D convolution blocks with $3 \times 3$ kernels, 64 and 128 feature maps respectively, $2 \times 2$ maxpooling, and ReLU non-linearity. The baseline model uses transformers (Vaswani et al., 2017) with 16 encoder blocks and 6 decoder blocks, each with 1024 dimensions, 16 heads, 4096 feed-forward units, and Sinusoidal positional embeddings are added to the output of the convolutional context layers (Mohamed et al., 2019). For the LegoNN encoder-only model trained using the CTC loss, we use the same architecture as the encoder of the baseline model along with a length control unit which reduces the length of

the input by a factor of 1.5 with a maximum allowable length of 230 time-steps. These positional embeddings are then passed through 6 layers of transformer layers with cross-attention as described in §4.4.3. The LegoNN decoder uses the same architecture as the baseline decoder. All Ingestor components (§4.4.4) use RF=1, 3 layers of transformers with 1024 dimensions, 16 heads and 4096 feed-forward layer. The BeamConv ingestor uses K=10, and embedding size= 100.

*Training:* We use an average batch-size=300 utterances, weight decay=$1e^{-6}$, and lr=$1e^{-3}$ with $35k$ warm-up steps then exponentially decay to $5e^{-6}$ over $44k$ steps. We follow the SS augmentation policy of SpecAugment (Park et al., 2019) without time-warping.

### 4.5.2 Machine translation task

*Data:* We train our models on the standard 4.5M dataset from WMT En-De task, as used by (Vaswani et al., 2017; Ott et al., 2018). We filter the training data to have a length ratio of 1.5 with 250 as max tokens. We test them on the newstest2011-2016 sets excluding the newstest2013 for the validation set. We use the shared 32K BPE vocabulary (Sennrich et al., 2016) provided by (Vaswani et al., 2017). We average a moving window of 10 checkpoints, and pick the one with the best validation BLEU score. A beam of 5 and length penalty of 0.6 are used for decoding. Models are evaluated on case-sensitive tokenized BLEU with compound-splitting using `multi-bleu.pl` (Moses-SMT, 2018). For De-En decoder models, we use the WMT19 data and prepare it as needed for the transfer task, detailed in Appendix A.1.1 and A.1.2. For the WMT16 Ro-En task, we use the data prepared by (Lee et al., 2018).

*Model architecture:* The baseline model uses transformers (Vaswani et al., 2017) with 12 encoder blocks and 6 decoder blocks, each with 1024 dimensions, 16 heads, 4096 feed-forward units, and sinusoidal positional embeddings. All embedding tables are shared across the model. To control for the extra parameters in the proposed LegoNN models, we added 6 additional encoder blocks which improved the baseline model. Other strategies for using these parameters in the baseline model yielded inferior performance. The LegoNN encoder model uses 12 transformer blocks with 1024 dimensions, 8 heads, and 2048 feed-forward units, with OLC, upsampling the input length by a factor of 2, applied to the second half of the encoder. The LegoNN decoder use the same architecture as the baseline decoder. All Ingestor components (§4.4.4) use RF 1, 3 layers of transformers with 1024 dimensions, 16 heads and 4096 feed-forward units. The BeamConv ingestor uses K=200 and embedding size=300. Input embedding tables are shared with encoder, ingestor, and decoder tables where applicable.

*Training:* We use an average batch-size=4000 sentences, weight decay=0.1, and lr=$1e^{-3}$ with $35k$ warm-up steps then inverse square root decay for $45k$ steps.

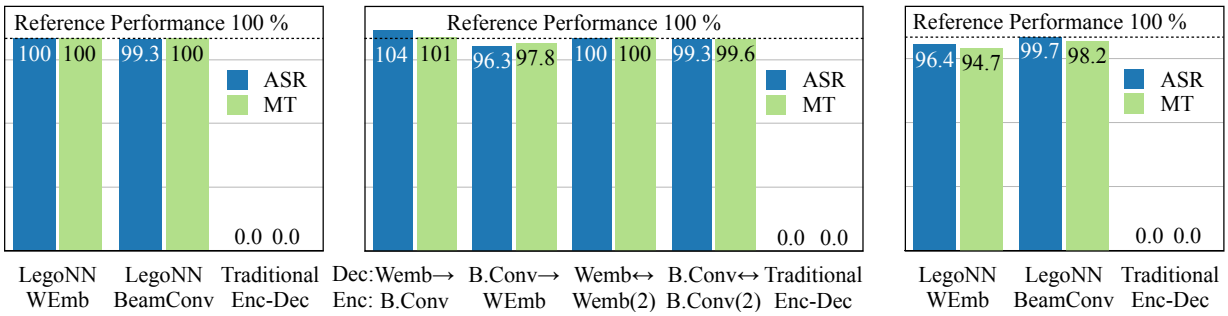



Figure 4.4: With reference performance of LegoNN models of Tables 4.1 and 4.2 (normalization to 100% enables us to plot ASR and MT performances in the same graph), we show the interchangeability of LegoNN models compared to traditional enc-dec models under three conditions: swapping encoder and decoder modules of two models trained with different random seeds (left), swapping modules between LegoNN models with different architectures and trained with different ingestor types (middle), and matching arbitrary decoder modules with LegoNN encoder modules trained in-isolation (right).

## 4.6 Results

The objective of our experiments is to demonstrate the feasibility of reusing LegoNN modules between ASR and MT tasks, as presented in the four scenarios on the right side of Figure 4.3.

### 4.6.1 Performance of LegoNN models

First, we show the performance of LegoNN models on their original tasks, without sharing any modules. Table 4.1 and 4.2 show the performance of models trained [2] with the LegoNN procedure: A CTC loss over an intermediate vocabulary at the encoder output with the proposed output length controller (OLC) and ingestor (ING) components for both ASR and MT benchmarks. For the WMT task, our best LegoNN model with the WEmb ingestor is only 0.8 BLEU behind our strong baseline encoder-decoder model (better than the publicly available reference model by (Ott et al., 2018)[3]) while being composed of modular reusable pieces (as we show in §4.6.2 and §4.6.3). For the ASR task, our LegoNN models reach the same level of performance as the baseline encoder-decoder model. The good ASR performance of the gradient-isolated case of the BeamConv ingestor shows that the ASR task is amenable to decomposition of linguistic unit recognition and language generation components. [4]

[2] Table 4.1 and 4.2 report averaged scores over three random seeds.

[3] We downloaded the public model by (Ott et al., 2018) to score the newstest2011-2016 test sets which weren't reported in their original paper.

[4] Using CTC models with language models is common for ASR (Miao et al., 2015; Amodei et al., 2016) and there can be some confusion regarding the relation of LegoNNs with them. We discuss this in the Appendix A.7.

Table 4.1: BLEU Scores (↑) on WMT for LegoNN and baseline enc-dec MT models.

| MT Task | Loss Criterion | | WMT En→De | |
|---|---|---|---|---|
| | CTC | CE | dev(↑) | test(↑) |
| Scaling NMT (Ott et al., 2018) | ✗ | ✓ | 27.3 | 28.0[3] |
| Baseline Models (Our Implementation) | | | | |
| Baseline Enc-Dec | ✗ | ✓ | 27.6 | 28.3 |
| **LegoNN Models** | | | | |
| Encoder Only | ✓ | ✗ | 19.1 | 18.5 |
| Encoder + BeamConv Decoder | ✓ | ✓ | 26.7 | 26.9 |
| Encoder + WEmb Decoder | ✓ | ✓ | 27.2 | 27.5 |

Table 4.2: % WER (↓) on SWBD 300h (no LM) for LegoNN and baseline enc-dec ASR models.

| ASR Task | Loss Criterion | | Eval 2000 | |
|---|---|---|---|---|
| | CTC | CE | SWB(↓) | CH(↓) |
| LAS + SpecAugment (Park et al., 2019) | ✗ | ✓ | 7.3% | 14.4% |
| IBM SWBD 300h (Tüske et al., 2020) | ✗ | ✓ | 7.6% | 14.6% |
| ESPNET (Karita et al., 2019) | ✓ | ✓ | 9.0% | 18.1% |
| Kaldi Hybrid system (Povey et al., 2016) | LF-MMI | | 8.8% | 18.1% |
| Baseline Models (Our Implementation) | | | | |
| Baseline Enc-Dec | ✗ | ✓ | 8.5% | 18.0% |
| **LegoNN Models** | | | | |
| Encoder Only | ✓ | ✗ | 11.5% | 24.1% |
| Encoder + BeamConv Decoder | ✓ | ✓ | 8.5% | 18.2% |
| Encoder + WEmb Decoder | ✓ | ✓ | 8.4% | 18.2% |

### 4.6.2 Modularity of LegoNN models

Figure 4.4 shows that LegoNN are modular; the decoder modules can be reused with encoders from other models for the same task [5](section §4.6.3 presents cross-task performance). The 100% reference level refers to the respective performance of the LegoNN and encoder-decoder models from Table 4.1 and 4.2. Normalizing the scores with respect to their reference performance allows us to plot ASR and MT systems in the same figure. For both ASR and MT tasks, with no fine-tuning, the performance barely changes when modules are swapped between two different models trained with different random seeds or completely different architectures, even though these two models have different learning dynamics due to the use of different ingestor types (Fig-

[5]All statistics computed in Figure 4.4 report averaged scores across all possible combinations of mixing three random seeds from each module.

Table 4.3: BLEU scores (↑) on Ro-En MT task using a LegoNN model composed of a decoder from a De-En MT model and an encoder-only module trained on Ro-En data.

| MT Task | Ro→En BLEU (↑) | GPU Hours (↓) |
|---|---|---|
| Baseline Ro-En Enc-Dec | 34.0 | 144 |
| LegoNN Encoder only | 30.7 | 120[6] |
| + De-En WMT LegoNN Modules | | |
| BeamConv Decoder | 33.0 | 0[7] |
| WEmb Decoder | **35.0** | 0[7] |

ure 4.4, left and middle). The right side of Figure 4.4 presents the case where a LegoNN encoder module is trained in isolation of any decoder module using a CTC loss, for either ASR or MT, then matched with an arbitrary decoder module at inference time. This is the most challenging condition, especially for the MT task. Relying entirely on the indices of top-p hypotheses, rather than their floating point marginal probabilities, the BeamConv ingestor shows more robustness when reused within the same task as compared to the WEmb ingestor. The traditional encoder-decoder models, which is built without reusability in mind, fail completely under all these conditions (shown on the right side of the three sub-figures).

### 4.6.3 Transfer of LegoNN modules across tasks

LegoNN modules can be reused across tasks with no fine-tuning and with almost no performance degradation. Table 4.3 shows the improved MT performance by 1.0 BLEU point when a Ro-En LegoNN encoder module is composed with a LegoNN decoder module that is trained on the De-En data (the second LegoNN scenario in Figure 4.3). Table 4.4 takes this a step further by mixing an MT trained decoder (De-En data) with an ASR trained encoder on the Europarl speech dataset (the third LegoNN scenario in Figure 4.3). The BeamConv ingestor depends on a fixed beam size $k$, which cannot be changed after initial training of the decoder module. This explains the inferior performance for the BeamConv as compared to the WEmb ingestor (which uses the full marginal distribution) when reused in a new task. To demonstrate the flexibility in building sequence-to-sequence models with LegoNN, Table 4.5 shows the fourth LegoNN scenario in Figure 4.3. A pronunciation modeling module trained on the TED-LIUM dataset is used in conjunction with a phoneme recognition module trained on the Europarl dataset. Then, a decoder trained on the De-En WMT task is added to the ASR model to bring the final WER, with no fine-tuning updates, just 0.6% from the baseline encoder-decoder model. The TED-LIUM dataset is used in this experiment because it is closer in speaking style to Europarl. Both the public TED talks and the Parliament speeches exhibit similarities in speaking style and are not as spontaneous as the Switchboard data. However, there is a clear domain mismatch which is apparent in the 26.7% WER of the initial TED-LIUM system when evaluated on the Europarl

Table 4.4: % WER (↓) on Europarl ASR task using a LegoNN model composed of a decoder from a De-En MT model and an encoder-only module trained on Europarl ASR data.

| ASR Task | Europarl % WER (↓) | GPU Hours (↓) |
|---|---|---|
| Baseline Europarl Enc-Dec | 18.4% | 22 |
| LegoNN Encoder only | 19.5% | 13[6] |
| + De-En WMT LegoNN Modules | | |
| BeamConv Decoder | 22.8% | 0[7] |
| WEmb Decoder | **18.4%** | 0[7] |

Table 4.5: % WER on the Europarl test with ASR encoder decomposed into two modules. Modules trained on the TED-LIUM dataset are combined with ones trained on Europarl and WMT to bring the overall WER of the LegoNN ASR system just 0.6% from the baseline with no fine-tuning.

| ASR Task | Europarl % WER (↓) |
|---|---|
| Baseline Europarl Enc-Dec | 18.4% |
| TED. Phoneme Recognizer + TED. Pronunciation Model | 26.7% |
| Europarl Phoneme Recognizer + TED. Pronunciation Model | 20.5% |
| Europarl Phoneme Recognizer + TED. Pronunciation Model + De-En WMT Decoder | **19.0%** |

test set. Utilizing LegoNN to develop models for new tasks benefits from the data-efficiency of encoder-only modules (Additional experiments regarding the data-efficiency of encoder-only models is shown in Appendix A.6) and overall shorter development time, e.g., 13 vs 22 gpu-hours on Europarl ASR and 120 vs 144 gpu-hours on Ro-En WMT. [6] [7]

### 4.6.4 Fine-tuning of LegoNN models

So far, we showed matching or better results for LegoNN models composed of pre-trained modules with no fine-tuning. Given that LegoNN decoders with the WEmb ingestor preserve full differentiability, such LegoNN models composed of pre-trained modules can be fine-tuned towards the target task. Table 4.6 shows that fine-tuning the De-En decoder with the Ro-En encoder achieves another 0.5 BLEU point, leading to an improvement of 1.5 points over the baseline model. Although the non-fine-tuned two-module LegoNN model is better than the three-module one for the Europarl English ASR task, fine-tuning LegoNN models composed of two pre-trained modules achieved 12.5% WER reduction, compared to a 19.5% reduction for the three-module one. Fine-tuning the three-modules helped reduce the domain mismatch while preserving the

[6] GPU hours for encoder training can be improved further by using CuDNN based CTC implementation (nvidia, 2022)

[7] Composing LegoNN decoders with the LegoNN encoders, is a simple plug and play and doesn't require any fine-tuning steps. So no additional gpu hours is used for the decoder.

Table 4.6: BLEU (↑) on Ro-En WMT and %WER (↓) on Europarl ASR task for the LegoNN models before and after end-to-end fine-tuning of the model composed of modules from different tasks in Table 4.3, Table 4.4 and Table 4.5.

| Composed LegoNN Model | No fine-tuning | With fine-tuning | Metric |
|---|---|---|---|
| Table 4.3: Ro-En WMT Encoder + De-En WMT Decoder | 35.0 | **35.5** | BLEU (↑) |
| Table 4.4: Europarl ASR Encoder + De-En WMT Decoder | 18.4 | **16.1** | % WER (↓) |
| Table 4.5: Europarl Phoneme Recognizer + TED. Pronun. Model + De-En WMT Decoder | 19.0 | **14.8** | % WER (↓) |

benefits of the TED-LIUM data.

## 4.7 Related work

This work is related to the large body of work on probabilistic modeling for ASR and MT (Jelinek, 1997; Brown et al., 1990) where predictors produce normalized probabilities to be easily combined. However, probabilistic models only combine output scores as opposed to chaining modules while preserving their full differentiability as in LegoNN models. Hierarchical mixture of experts (Jordan and Jacobs, 1994) and Graph transformer networks (Bottou et al., 1997) motivated this work, however, the first doesn't ground intermediate representations and the second communicates them in the form of directed graphs.

The proposed LegoNN procedure builds upon several research efforts for sequence to sequence learning. Encoder-decoder models for machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017) and speech recognition (Chan et al., 2016; Bahdanau et al., 2016) form the basis of this work. Although the CTC loss (Graves et al., 2006) has first been applied to speech recognition (Graves et al., 2013; Sak et al., 2014; Hannun et al., 2014a; Chan et al., 2020), more recently it was shown effective for machine translation as well (Libovický and Helcl, 2018; Saharia et al., 2020). This encourages us to utilize the CTC loss for enforcing the intermediate vocabulary in LegoNN. Other non-autoregressive sequence to sequence mapping methods (Gu et al., 2018; Ghazvininejad et al., 2019, 2020a; Chen et al., 2019) are potential alternatives. The CTC loss has been combined with the cross-entropy loss in encoder-decoder speech recognition systems to encourage monotonic alignment between input and output sequences (Kim et al., 2017; Karita et al., 2019). Different from LegoNN, their decoder attends over the encoder hidden output representations, maintaining their tight coupling.

There have been many proposals for inducing a modular structure on the space of learned concepts either through hierarchically gating information flow or via high-level concept blueprints (Andreas et al., 2016; Devin et al., 2017; Purushwalkam et al., 2019) to enable zero- and few-shot transfer learning (Andreas et al., 2017; Socher et al., 2013; Gupta et al., 2020; Pathak et al., 2019).

## 4.8 Conclusion

In this chapter, we presented the LegoNN procedure for constructing encoder-decoder models that are composed of reusable modules. LegoNN models preforms competitively to the best encoder-decoder ASR and MT models on large scale benchmarks. A key to reusable modules is a pre-defined vocabulary that is shared between many tasks across which modules can be reused. Without any fine-tuning steps, a LegoNN decoder trained for the De-En WMT task can replace an ASR decoder module without any impact on performance, and provide better generation quality for a Ro-En WMT task. When fine-tuned for few thousand steps, LegoNN models composed from multiple tasks and domains improve Ro-En WMT baseline model by 1.5 BLEU points and provide up to 19.5% WER reductions to the Europarl English ASR task.

# Chapter 5

# Joint CTC/Attn Models for Non-Monotonic Sequence Tasks

This chapter builds upon Chapter 4 and (Kim et al., 2017) to build joint CTC/Attn Models for non-monotonic tasks like speech and machine translation. Inspired by the principles of task compositionality we build Hierarchical encoders that perform reordering and length-adjustment thereby validating the length and monotonicity assumptions for CTC. We also present two joint decoding strategies for models that jointly model both CTC and Attention based models. They further strengthen our proposed systems as these strategies can be directly applied to the legoNN models shown in Chapter 4. Additionally, the techniques introduced in this chapter for building CTC encoders can be used to extend the CTC Hybrid Systems (Chapter 3) towards non-monotonic tasks like speech and machine translation.

## 5.1 Joint CTC/Attention for Speech and Machine Translation

Connectionist Temporal Classification (CTC) is a widely used approach for speech recognition that performs conditionally independent monotonic alignment. At first sight, this may seem very different than the standard attentional approaches to machine translation. However, we show that the distinct properties of these approaches are actually complementary in joint CTC/attention based translation when effectively coerced into a single network. We first propose to jointly train encoders with an initial length-adjustment stage followed by a target-oriented re-ordering stage. Encoding inputs in this *hierarchical* manner produces representations that are monotonic with respect to the output, easing the alignment burden of the attentional decoder. We then propose to jointly decode by *synchronizing* likelihood estimation along hypothesis expansion via stepwise input consumption or output production. Since CTC does not model dependencies between outputs and determines output lengths using hard alignment information, it complements the autoregressive generation of attentional decoders which are prone to exposure, label, and length

biases. Our joint models outperform pure-attention baselines across six benchmark text-to-text (MT) and speech-to-text (ST) tasks.

## 5.2 Introduction

Connectionist Temporal Classification (CTC) (Graves et al., 2006), a non-autoregressive sequence transduction framework, has seen a recent surge in machine (MT) and speech translation (ST) (Xiao et al., 2022). Originally proposed for automatic speech recognition (ASR), CTC predicts *monotonic alignments* of output units to input frames of speech with a *conditional independence assumption* that each frame-level prediction can be made independently. At first glance, the aforementioned properties seem to make CTC poorly fit for translation. Unlike ASR, translation contains non-monotonic input-to-output mappings due to word-order differences (Bahdanau et al., 2015; Ghazvininejad et al., 2020a) and expansion where outputs may be longer than inputs (Haviv et al., 2021). Additionally, context is often required to coherently and faithfully translate original meanings (Qian et al., 2021a; Yin et al., 2021). ST is further complicated by the compositional nature of the task as it requires both recognizing and translating the input (Dalmia et al., 2021; Bahar et al., 2021).

Still, CTC's promise of fast, parallelized, and streaming compatible inference has enticed the non-autoregressive translation community (Saharia et al., 2020; Kasai et al., 2021). Various up-sampling techniques (Libovický and Helcl, 2018; Dalmia et al., 2022) have been proposed to overcome CTC's assumption that the input length is at least as long as the output. Further, self-attention has been shown to be effective for handling non-monotonic mappings (Chuang et al., 2021) and hierarchical multi-tasking has been shown to be effective for task composition (Deng et al., 2022; Yan et al., 2022b). Despite these cumulative efforts, purely CTC-based methods in MT and ST currently lag behind their autoregressive counterparts and have not reached the same prevalence as they have in ASR (Gu and Kong, 2021; Inaguma et al., 2021a).

Inspired by the success of Hybrid CTC/Attention in ASR (Watanabe et al., 2017), we investigate jointly modeling CTC with an autoregressive attentional encoder-decoder for translation. Our conjecture is that the monotonic alignment and conditional independence properties of CTC, which weaken purely CTC-based translation, actually strengthen the joint modeling scenario through three mechanisms: (1) by encouraging input representations to be re-ordered such that they are monotonic with respect to the output, thereby simplifying the task of the attentional decoder, (2) by augmenting autoregressive likelihood estimation with conditionally independent likelihoods that do not model dependence between outputs, easing exposure/label biases that inhibit generalization towards unseen scenarios and (3) by incorporating the hard input-output alignments into the decoding process, thereby making it easier for the decoder to generate sentences that are the correct length compared to standard autoregressive models

We first propose a hierarchical encoding scheme for effective joint training of CTC and atten-

tional decoders with a shared encoder. By decomposing input-to-output alignment into an initial *length-adjustment* stage followed by a *re-ordering* stage, we resolve the critical incompatibility between CTC's monotonic alignment property and the functional demands of MT/ST. We further propose a generic one-pass beam search algorithm for efficient joint decoding of CTC and attentional likelihoods which can synchronize hypothesis expansion along either the output or input axes. *Output*-synchronous one-pass beam search proposes candidates at each step using autoregressive decoder output posteriors and augments likelihood estimations of the resulting hypotheses using CTC prefix scoring. The *Input*-synchronous variant proposes candidates using conditionally independent CTC input-to-output alignment posteriors and augments likelihood estimations using the decoder.

We show the efficacy of joint CTC-Attention for MT and ST on multiple language-pairs and in multilingual settings; both joint training and joint decoding yield improvements over attention-only baselines (§5.6). We demonstrate conjecture (1) by examining the monotonicity of decoder source attention in our proposed models (§5.6.2). This monotonicity alleviates the burden of variable word-ordering across languages, allowing multilingual decoders to more efficiently share source attention parameters across languages (§5.6.2). We demonstrate conjecture (2) by showing superior performance of our joint models when tested towards out-of-domain scenarios (§5.6.3). We demonstrate conjecture (3) by examining the stability of length prediction in joint decoding. We find that output lengths can be smoothly controlled in joint models while purely autoregressive models degenerate for long output lengths (§5.6.3). Finally, we present the choice of input vs. output-synchronous one-pass beam search as proposition of speed vs. accuracy (§5.6.4).

## 5.3 Background and Motivation

This section first provides a general formulation of seq2seq transduction before showing how CTC and attentional decoder models function under this formulation in §5.3.1 and §5.3.2 respectively. We also discuss the previously proposed Hybrid CTC/Attention architecture for ASR (Watanabe et al., 2017) in §5.3.3. We form conjectures regarding the key mechanisms behind the empirical findings of this prior work, motivating our approach in §5.4.

Seq2seq transduction is a mapping of a $T$-length input sequence, $X = \{\mathbf{x}_t \in \mathcal{S}^{\text{src}} | t = 1, ..., T\}$, to an $L$-length output sequence, $Y = \{y_l \in \mathcal{V}^{\text{tgt}} | l = 1, ..., L\}$.[1] Using Bayesian decision theory, we seek output, $\hat{Y}$, from all possible sequences, $\mathcal{V}^{\text{tgt}}*$:

$$\hat{Y} = \underset{Y \in \mathcal{V}^{\text{tgt}}*}{\text{argmax}}\ P(Y|X) \tag{5.1}$$

where $P(Y|X)$ is the posterior distribution. CTC and attentional decoders both follow this frame-

[1] In ASR, MT, and ST, the output sequences consist of discrete vocabulary units while the input sequence may be real or discrete depending on whether the source is speech or text.

work, but do so with different modeling properties.

### 5.3.1 CTC Models

CTC (Graves et al., 2006) is from a family of alignment-based networks (Hinton et al., 2012; Graves, 2012) which find warping paths between sequences of different lengths such that each unit in the input is mapped to a unit in the output. Input-to-output alignment is an important property of CTC, but requires several modeling assumptions.

First, CTC makes a *length assumption* that the $T$-length inputs are at least as long as the $L$-length outputs.[2] This allows for the introduction of alignment sequences which are of identical length to the input, $Z = \{z_t \in \mathcal{V}^{\text{tgt}} \cup \{\varnothing\} | t = 1 \dots T\}$, where $\varnothing$ denotes a null emission. Instead of producing output posteriors, $P(Y|X)$, CTC produces alignment posteriors, $P(Z|X)$. Every alignment sequence, $Z$, maps deterministically to an output sequence, $Y$, using CTC's repeat and blank removal operations.[3] Note that this $Z \rightarrow Y$ operation compresses $Z$ without any possibility of re-ordering where if $z_i \rightarrow y_j$, $z_{i'} \rightarrow y_{j'}$, and $i < i'$ then it must be true that $j < j'$. Therefore CTC requires a *monotonicity assumption* that the input-to-output warping path can be modeled by one-to-one input-to-alignment mapping (via $P(Z|X)$ posterior) followed by monotonic alignment-to-output compression (via blank/repeat rules).

Since there may be a number of $Z$ which map to the same $Y$, CTC estimates the output posterior during training, $P(Y|X)$, by marginalizing over alignment posteriors, $P(Z|X)$, for the set of all alignments that map to the output, $\mathcal{Z}(X, Y)$. Further, CTC makes a *conditional independence assumption*, yielding an estimation of Eq. (5.1):

$$P(Y|X) \approx \sum_{Z \in \mathcal{Z}(X,Y)} \underbrace{\prod_{t=1}^{T} P(z_t | X, \cancel{z_{1:t-1}})}_{\triangleq P_{\text{CTC}}(Z|X)} \tag{5.2}$$

Critically, this conditional independence assumption grants the Markov property for efficiently computing Eq. (5.2) via dynamic programming during inference. Conditional independence also allows CTC to eschew exposure and label biases (Hannun, 2019) that are apparent in their autoregressive counterparts (Bottou et al., 1997; Ranzato et al., 2016).

### 5.3.2 Attentional Decoder Models

While CTC approximates the Bayesian formulation in Eq. (5.1), the attentional decoder exactly models the likelihood of output units, $P(y_l | y_{1:l-1}, X)$, conditioned on the input, $X$, and the pre-

[2] $L \leq T$ holds true for most cases in ASR and ST, it does not in MT. We address this limitation in §5.4.1.1.

[3] If $Z = [a, \varnothing, a, b, b]$, repeat removal first yields $Z' = [a, \varnothing, a, b]$ before blank removal yields $Y = [a, a, b]$.

viously generated output sequence, $y_{1:l-1}$, as follows:

$$P_{\text{Attn}}(Y|X) \triangleq \prod_{l=1}^{L} P(y_l|y_{1:l-1}, X) \tag{5.3}$$

Compared to the conditionally independent CTC, the autoregressive attentional decoders enable contextualization. However, the local normalization for $P(y_l|y_{1:l-1}, X)$ engenders exposure and label biases (Bottou et al., 1997; Ranzato et al., 2016).

Since the attention mechanism performs soft-alignment of input-to-output (Bahdanau et al., 2015), there is also no monotonicity assumption as in CTC. This allows for flexible mappings, which are fit for translation tasks with large amounts of re-ordering, but prior works have shown that too much flexibility can destabilize optimization (Kim et al., 2017). Lastly, output lengths are not aligned with input lengths as in CTC. Instead, the attentional decoder is as an autoregressive generator where generation is stopped by introducing a special stop token, <eos>, to the output vocabulary, $y_l \in \mathcal{V}^{\text{tgt}} \cup \{\text{<eos>}\}$. Since these models need to predict $P(y_l = \text{<eos>}|y_{1:l-1}, X)$, label bias with respect to the stop token, <eos>, manifests as a brevity problem (Murray and Chiang, 2018).

### 5.3.3 Joint CTC/Attention Modeling for ASR

As previously shown by (Kim et al., 2017; Watanabe et al., 2017), jointly modeling CTC and an attentional decoder is highly effective in ASR. Not only was overall performance improved over CTC-only and attention-only approaches, joint modeling was showed faster and more robust convergence during training. Per conjecture (1), the mechanism behind these improvements is the enrichment of encoder representations with *target-oriented alignment* information, which reduces the input-to-output alignment burden on the attentional decoder.

The foundation of this architecture is a shared encoder, Enc, which feeds into both CTC, $P_{\text{CTC}}(\cdot)$, and attentional decoder, $P_{\text{Attn}}(\cdot)$, posteriors:

$$\mathbf{h} = \text{Enc}(X) \tag{5.4}$$

$$P_{\text{CTC}}(z_t|X) = \text{CTC}(\mathbf{h}_t) \tag{5.5}$$

$$P_{\text{Attn}}(y_l|X, y_{1:l-1}) = \text{Dec}(\mathbf{h}, y_{1:l-1}) \tag{5.6}$$

where $\text{CTC}(\cdot)$ denotes a projection to the CTC output vocabulary, $\mathcal{V}^{\text{tgt}} \cup \{\varnothing\}$ followed by softmax, and $\text{Dec}(\cdot)$ denotes autoregressive decoder layers followed by a projection to the decoder output vocabulary, $\mathcal{V}^{\text{tgt}} \cup \{\text{<eos>}\}$, and softmax. The joint network is optimized via a multi-tasked objective, $\mathcal{L}^{\text{ASR}} = \mathcal{L}^{\text{ASR}}_{\text{CTC}} + \lambda \mathcal{L}^{\text{ASR}}_{\text{Attn}}$, where $\lambda$ interpolates the CTC loss and the cross-entropy loss of the decoder.

Watanabe et al. (2017) propose to jointly decode with a one-pass beam search approximating:

$$\hat{Y} = \underset{Y \in \mathcal{V}^{\text{tgt}}*}{\text{argmax}} \Big\{ \big( \sum_{Z \in \mathcal{Z}} P_{\text{CTC}}(\cdot) \big) \times P_{\text{Attn}}(\cdot)^{\lambda} \Big\} \tag{5.7}$$

where $\hat{Y}$ represents the most likely output from all possible outputs, $\mathcal{V}^{\text{tgt}}*$. Watanabe et al. (2017) found that joint decoding improves over attention-only decoding and stabilizes the hypothesis lengths. Per conjecture (2), the mechanism behind the former effect is the *conditional independence* of CTC likelihood estimations, which eases the exposure and label biases of the attentional decoder. Per conjecture (3), the mechanism behind the latter effect is the output length information inherent in CTC's *input-to-output* alignment likelihoods, which eases the length problem of the attentional decoder.

## 5.4 Proposed Joint CTC/Attn for MT/ST

In training, our objective is to apply joint CTC/Attention modeling in translation to enrich a shared encoder with target-oriented alignment information. The joint CTC/Attention construction with a monolithic encoder (§5.3.3) has several key limitations when applied to translation. Firstly, we need to relax the monotonicity assumption (§5.3.1) of CTC to fit the MT/ST tasks. Further, for MT we seek to account for the length assumption of CTC, and for ST we seek to recognize before translating the input. We therefore propose to use a hierachical encoding scheme (§5.4.1) which first aligns inputs to length-adjusted *source*-oriented encodings before aligning to re-ordered *target*-oriented encodings.

In decoding, our objective is to improve generalization and end detection by augmenting autoregressive likelihoods with conditionally independent CTC alignment likelihoods. Our proposed joint decoding method is a synchronous one-pass beam search (§5.4.2) which can perform either *output* or *input*-synchronous hypothesis expansion (§5.4.2.1). We theorize that while both variants are valid approximations of the joint decoding objective, input-synchronous is expected to be faster while output synchronous to be more accurate (§5.4.2.2).

### 5.4.1 Hierachical Encoding for MT/ST

In order to build target-oriented encodings for translation, we decompose the process into two functions: length-adjustment and re-ordering. For MT, we up-sample the lengths of the source-oriented encodings in order to satisfy the length assumption of CTC (§5.3.1). For ST, we down-sample the lengths of the source-oriented encodings to coerce a discrete textual representation of the real-valued speech input. We enforce source-orientations using CTC criteria that seek to align intermediate encoder representations towards source text sequences.

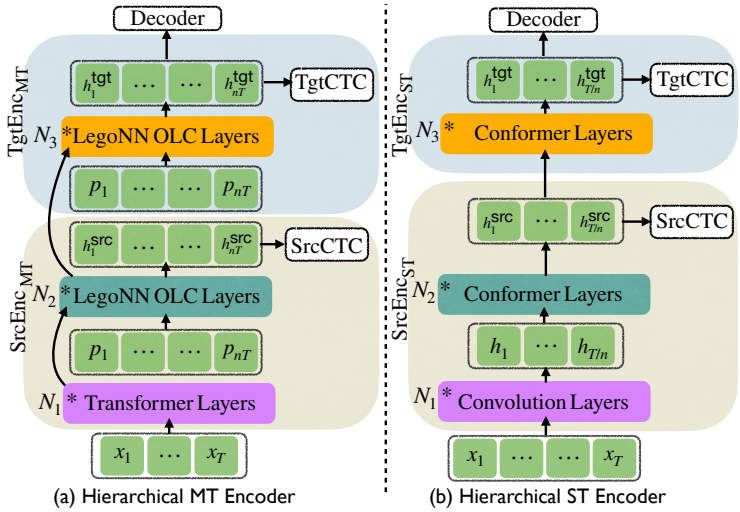



Figure 5.1: Hierarchical MT/ST encoders where representations are first up/down-sampled by $\textsc{SrcEnc}_{\text{MT/ST}}$ and then re-ordered by $\textsc{TgtEnc}_{\text{MT/ST}}$.

We then obtain target-oriented encodings with hierarchical encoder layers, where re-ordering is enforced using CTC criteria that seek to align final representations towards target text sequences. This style of hierarchical encoding via intermediate CTC supervision has been applied successfully for various multi-objective scenarios (Sanabria and Metze, 2018; Higuchi et al., 2022; Deng et al., 2022; Yan et al., 2022b). For our purposes, the intermediate supervision towards source text allows for the compartmentalization of length-adjustment to the initial encoder layers, leaving only the task of re-ordering of text-to-text for latter encoder layers.

#### 5.4.1.1 MT/ST Encoder Architectures

As shown in Figure 5.1, our MT and ST hierarchical encoders consist of the following components:

$$\mathbf{h}^{\text{Src}} = \textsc{SrcEnc}_{\text{MT/ST}}(X) \tag{5.8}$$

$$P_{\text{CTC}}(z_t^{\text{Src}}|X) = \textsc{SrcCTC}_{\text{MT/ST}}(\mathbf{h}_t^{\text{Src}}) \tag{5.9}$$

$$\mathbf{h}^{\text{Tgt}} = \textsc{TgtEnc}_{\text{MT/ST}}(\mathbf{h}^{\text{Src}}) \tag{5.10}$$

$$P_{\text{CTC}}(z_t^{\text{Tgt}}|X) = \textsc{TgtCTC}_{\text{MT/ST}}(\mathbf{h}_t^{\text{Tgt}}) \tag{5.11}$$

where $\textsc{SrcEnc}_{\text{MT}}(\cdot)$ is realized by $N_1$ Transformer (Vaswani et al., 2017) layers followed by $N_2$ up-sampling LegoNN Output Length Controller (OLC) layers (Dalmia et al., 2022), while $\textsc{TgtEnc}_{\text{MT}}(\cdot)$ is realized by $N_3$ non-up-sampling LegoNN OLC layers. We chose LegoNN based on its previously demonstrated effectiveness for up-sampling textual representations.

$\textsc{SrcEnc}_{\text{ST}}(\cdot)$ is realized by $N_1$ convolutional blocks for downsampling (Dong et al., 2018) fol-

**Algorithm 1** General One-Pass Beam Search: lines 10/11 invoke output/input-synchronous variants.

```
1: procedure SEARCH(X, N, STEP, b, p)
2:     topPrtHs = {<sos> : 1.0}; allEndHs = {}
3:     for i ∈ N do
4:         prtHs, endHs = STEP(topPrtHs, X, i, p, N)
5:         topPrtHs = top-k(prtHs, k = b)
6:         allEndHs = allEndHs ∪ endHs
7:     end for
8:     return top-1(allEndHs)
9: end procedure
```

```
10: SEARCH(X, max L, OUTPUTSTEP, b, p)        ▷ output-sync
11: SEARCH(X, T, INPUTSTEP, b, p)             ▷ input-sync
```

lowed by $N_2$ Conformer (Gulati et al., 2020), while $\text{TGTENC}_{\text{ST}}(\cdot)$ is realized by $N_3$ Conformer layers. We chose Conformer based on its previously demonstrated effectiveness for modeling local and global dependencies in speech signals. The hierarchical encoders are jointly optimized with an attentional decoder using a multi-tasked objective, $\mathcal{L} = \mathcal{L}_{\text{SRCCTC}} + \lambda_1 \mathcal{L}_{\text{TGTCTC}} + \lambda_2 \mathcal{L}_{\text{ATTN}}$, where $\lambda$'s interpolate between source-oriented CTC, target-oriented CTC, and decoder cross-entropy.

### 5.4.2 One-Pass Synchronous Joint Decoding

In order to make the joint decoding in Eq. (5.7) computationally tractable, we need to estimate CTC and attentional decoder likelihoods on a subset of all possible output sequences, $\mathcal{V}' \subseteq \mathcal{V}^{\text{tgt}}*$. There are a family of two-pass decoding algorithms (Watanabe et al., 2017; Sainath et al., 2019), which accomplish this by first estimating the likelihoods of $\mathcal{V}'$ with one module and then re-scoring the estimates with the other module. In these approaches, the subset $\mathcal{V}'$ is determined asynchronously, meaning the joint likelihood is not considered until the re-scoring step. If the attentional decoder is used to determine $\mathcal{V}'$, then $\mathcal{V}'$ would suffer from exposure/label bias and the length problem (§5.3.3). On the other hand, if CTC is used to determine $\mathcal{V}'$, the lack of contextual modeling in CTC leads to poor estimates of $\mathcal{V}'$ – particularly for translation.

Instead, we synchronize the two likelihood estimators along the hypothesis expansion step of beam search. This way the selection of $\mathcal{V}'$ considers the joint likelihood in a one-pass synchronous beam search, removing the problematic first-pass of the two-pass approaches. Watanabe et al. (2017) achieve one-pass joint decoding by synchronizing hypothesis expansion along with the step-wise production of output units by the attentional decoder. We consider this to be an *output*-synchronous one-pass beam search. It is also possible to perform an *input*-synchronous one-pass beam search, where hypotheses are expanded along with the step-wise consumption of input units

[4] Input-synchronization was previously applied for decoding CTC/LM (Hannun et al., 2014b) and RNN-T (Saon et al., 2020), but not for jointly decoding CTC/Attentional decoder.

**Algorithm 2** *Output*-Synchronous Step Function: attentional decoder proposes candidates to expand hypotheses which are all of $l$-length at step $l$.

**procedure** OUTPUTSTEP(prtHs, $X, l, p, \max L$)
  newPrtHs $= \{\}$; endHs $= \{\}$
  **for** $y_{1:l-1} \in$ prtHs **do**
    attnCnds $=$ top-k$(P_{\text{Attn}}(y_l|X, y_{1:l-1}), \text{k} = p)$
    **for** $c \in$ attnCnds **do**
      $y_{1:l} = y_{1:l-1} \oplus \text{c}$
      $\alpha_{\text{CTC}} = \text{CTCScore}(y_{1:l}, X_{1:T})$
      $\alpha_{\text{Attn}} = \text{AttnScore}(y_{1:l}, X_{1:T})$
      $\beta = \text{LengthPen}(y_{1:l})$
      $P_{\text{Beam}}(y_{1:l}|X) = \alpha_{\text{CTC}} + \alpha_{\text{Attn}} + \beta$
      **if** ($c$ is <eos>) or ($l$ is max$L$) **then**
        endHs$[y_{1:l}] = P_{\text{Beam}}(\cdot)$
      **else**
        newPrtHs$[y_{1:l}] = P_{\text{Beam}}(\cdot)$
      **end if**
    **end for**
  **end for**
  **return** newPrtHs, endHs
**end procedure**

Hypothesis Expansion

Joint Scoring

End Detection

**Algorithm 3** *Input*-Synchronous Step Function: CTC proposes candidates to expand hypotheses which are all produced from $t$ input units at step $t$.

**procedure** INPUTSTEP(prtHs, $X, t, p, T$)
  newPrtHs $= \{\}$; endHs $= \{\}$
  CTCCnds $=$ top-k$(P_{\text{CTC}}(z_t|X), \text{k} = p)$
  **for** $y \in$ prtHs **do**
    **for** $c \in$ CTCCnds **do**
      **if** ($c$ is $\varnothing$) or ($c$ is $y$[-1]) **then**
        $\tilde{y} = y$
      **else**
        $\tilde{y} = y \oplus \text{c}$
      **end if**
      $\alpha_{\text{CTC}} = \text{CTCScore}(\tilde{y}, X_{1:t})$
      $\alpha_{\text{Attn}} = \text{AttnScore}(\tilde{y}, X_{1:T})$
      $\beta = \text{LengthPen}(\tilde{y})$
      $P_{\text{Beam}}(\tilde{y}|X) = \alpha_{\text{CTC}} + \alpha_{\text{ATTN}} + \beta$
      **if** $t$ is $T$ **then**
        endHs$[\tilde{y}] = P_{\text{Beam}}(\cdot)$
      **else**
        newPrtHs$[\tilde{y}] = P_{\text{Beam}}(\cdot)$
      **end if**
    **end for**
  **end for**
  **return** newPrtHs, endHs
**end procedure**

(and production of input-to-output alignment units) by CTC.[4] In fact, both variants can be unified under a general one-pass beam search algorithm as shown in Algorithm 1. The two variants are distinguished only by the iteration range, $N$, and beam step function, STEP.

#### 5.4.2.1 Input/Output-Synchronous Beam Steps

As shown in Algorithms 2 and 3, both beam step functions consist of the same operational blocks: hypothesis expansion, joint scoring, and end detection. However, the inner-workings differ.

The OUTPUTSTEP performs hypothesis expansion by computing the attentional decoder's output posterior at label step $l$, $P_{\text{Attn}}(y_l|X, y_{1:l-1})$ for each partial hypothesis, $y_{1:l-1}$. A pre-beam size, $p$, is used to select the top candidate output units (Seki et al., 2019), attnCnds, which are used to expand the partial hypotheses via concatenation, denoted by $\oplus$. The INPUTSTEP performs hypothesis expansion by computing CTC's alignment posterior at time step $t$, $P_{\text{CTC}}(z_t|X)$. The same pre-beam size, $p$, is used to select top candidate alignment units, CTCCnds, but partial hypotheses are only expanded for non-blank and non-repeat candidates.

In the joint scoring block, the attentional decoder likelihood, AttnScore$(\cdot)$, and length penalty/reward, LengthPen$(\cdot)$, are applied similarly in both variants (He et al., 2016). However, CTC likelihood, CTCScore$(\cdot)$, is applied over the full input, $X_{1:T}$, in OUTPUTSTEP and over the partial input, $X_{1:t}$, in INPUTSTEP. This critical difference creates a speed vs. accuracy trade-off (§5.4.2.2).

Finally in end detection, OUTPUTSTEP must check for the stop token, <eos>, which may be proposed by attnCnds. On the other hand, INPUTSTEP simply knows the end when all input units

| Model Name | Model Type | | | MT | | | ST | | |
|---|---|---|---|---|---|---|---|---|---|
| | Joint Train? | Joint Decode? | Decoding Method | IWSLT14 De-En | IWSLT14 Es-En | MTedX All-En | MuST-C-v2 En-De | MuST-C-v2 En-Ja | MTedX All-En |
| Pure-Attn (Prior) | ✗ | ✗ | Attn O-sync | (32.15)† | (38.95)† | -♢ | 21.0‡ | 11.6‡ | -♢ |
| Pure-Attn (Ours) | ✗ | ✗ | Attn O-sync | 32.8 (33.73) | 39.0 (39.86) | 25.6 | 27.8 | 14.3 | 22.7 |
| Joint CTC/Attn | ✓ | ✗ | CTC I-sync | 27.3 | 33.8 | 22.4 | 24.4 | 10.2 | 21.4 |
| Joint CTC/Attn | ✓ | ✗ | Attn O-sync | 33.6 | 39.5 | 28.0 | 28.3 | 14.2 | 23.7 |
| Joint CTC/Attn | ✓ | ✓ | Joint I-sync | 33.7 | 39.7 | 27.8 | **29.2** | 15.1 | **25.1** |
| Joint CTC/Attn | ✓ | ✓ | Joint O-sync | **34.1** | **39.9** | **28.1** | **29.2** | **15.3** | **25.1** |

Table 5.1: Test set performances, as measured by BLEU (↑), of our proposed joint CTC/Attention models compared to pure-attention baselines. Joint CTC/Attention models are always jointly trained, but can be either jointly decoded using input/output synchronony or decoded using only their CTC or attention branches. For IWSLT14, we mention (*tokenized BLEU*) for comparison with prior works. †Raunak et al. (2020) and ‡Fukuda et al. (2022) are based on fairseq (Ott et al., 2019). ♢Prior MTedX works show only All-All or pair-wise settings.

have been consumed ($t = T$). Note that the stop token is only in the decoder's vocabulary (§5.3.2) and is thus never proposed by CTCCnds.

#### 5.4.2.2 Theoretical Speed vs. Accuracy

By comparing the CTC likelihood estimation in InputStep vs. in OuptputStep, it can be seen that there is a trade-off between speed and accuracy. First, note that the step-wise estimate in OutputStep, CTCScore$(y_{1:l}, X_{1:T})$, is a marginalization over the likelihoods of all possible alignments of the partial hypothesis, $y_{1:l}$, as follows: $\sum_{Z \in \mathcal{Z}} P(Z|X)$. Since the partial hypotheses in each step are single unit increments on the previous step, $y_{1:l-1}$, a dynamic programming implementation computes the likelihood with $\mathcal{O}(bpT)$ log-additions (Watanabe et al., 2017).

Next, consider that the complexity of the step-wise estimate in InputStep, CTCScore$(\tilde{y}, X_{1:t})$, is only $\mathcal{O}(bp)$ as a dynamic programming implementation only needs to increment the score of the previous step with the alignment posterior information of the current $t$-th step (Hannun et al., 2014b). Although this makes the InputStep faster, the estimation only yields a marginalization over the likelihoods of a pruned set of partial alignments of the partial hypothesis, $\tilde{y}$, as follows: $\sum_{\mathcal{Z}'} P(Z|X_{1:t})$. Even at step $T$, the estimation is not equivalent to the full marginalization over $\mathcal{Z}$ as beam pruning yields only a subset $\mathcal{Z}' \subseteq \mathcal{Z}$ which can be considered.[5] Therefore, the speed-up of the time-synchronous variant comes at the cost of accuracy — and this trade-off is a function of beam size.

[5]Consider that with $b$=1, $t$=2, and prtHs={[<sos>, a]}, CTCScore([<sos>, a], $X_{1:2}$) will not consider $P_{\text{CTC}}(Z_{1:2} = [\varnothing, \text{a}]|X_{1:2})$ since the hypothesis [<sos>] was pruned $t$=1.

## 5.5 Experimental Setup

**Data:** We examine the efficacy of our proposed approaches on two language pairs for each of the MT and ST tasks. For MT, we use German-to-English (De-En) and Spanish-to-English (Es-En) from IWSLT14 (Cettolo et al., 2012). For ST, we use English-to-German (En-De) and English-to-Japanese (En-Ja) from MuST-C-v2 (Di Gangi et al., 2019). We also examine the multilingual setting of 6 European languages to English (All-En) from MTedX (Salesky et al., 2021) for both tasks. Full dataset descriptions for reproducibility are in §B.1.

**Modeling:** We compare our joint CTC/Attention models to purely attentional encoder-decoder baselines. All proposed and baseline models were tuned separately, using validation sets only, within the same hyperparameter search spaces for training and for decoding to ensure fair comparison. All experiments were conducted using ESPnet (Watanabe et al., 2018). Full descriptions of model sizes, hyperparameters, and pre-processing are in §B.1.

**Evaluation:** Unless otherwise indicated, we measure performance with detokenized case-sensitive BLEU (Post, 2018) on punctuated 1-references.

## 5.6 Results and Analyses

We first present main results on 6 benchmark MT and ST tasks in §5.6.1. We then examine the effect of hierarchical encoding (§5.4.1) on the decoder's source attention and present evidence towards conjecture (1) in §5.6.2. We also examine the effect of synchronous decoding (§5.4.2) on the generalization and length stability of beam search hypotheses, presenting evidence towards conjecture (2) and (3) respectively in §5.6.3. Finally in §5.6.4, we experimentally validate the theoretical speed vs. accuracy trade-offs of input and output-synchrony (§5.4.2.2).

### 5.6.1 Main Results

As shown in Table 5.1, joint CTC/Attention outperforms pure-attention across 3 MT tasks (IWSLT14 De-En + Es-En and MtedX All-En) and 3 ST tasks (MuST-C-v2 En-De + En-Ja and MTedX All-En). In the second horizontal partition, we show that joint training of hierarchical encoders with output-synchronous decoding of only the attention branch outperforms the pure-attention models without any joint training. In the last horizontal partition, we show that both joint input and output-synchronous decoding yield further improvements.

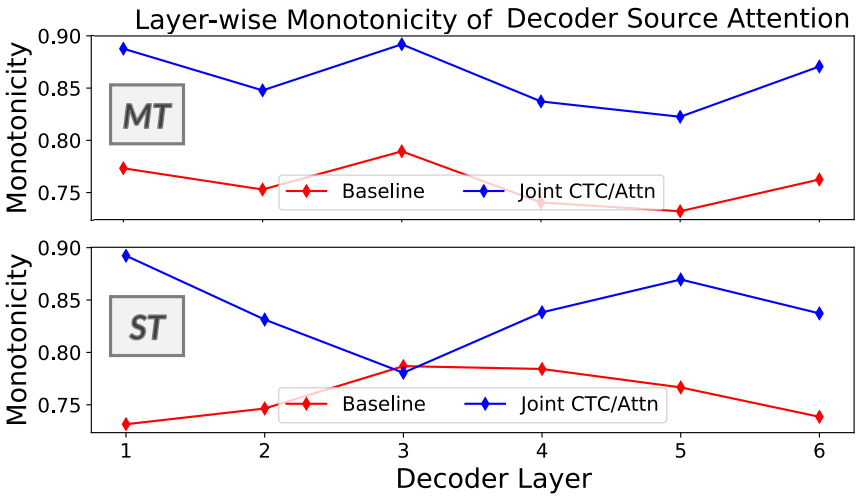



Figure 5.2: Layer-wise monotonicity of the source-attention patterns produced by MT/ST decoders.

### 5.6.2 Benefits of Hierarchical Encoding

We examine the regularization effect that CTC joint training has on the attentional decoder, per conjecture (1), by first quantifying the monotonicity, $m$ of a $(L, T)$ shaped source attention pattern, $A$:

$$m = \Big( \sum_{2<l\leq L} [\underset{t\in T}{\text{argmax}}\, A_l \geq \underset{t\in T}{\text{argmax}}\, A_{l-1}] \Big) / L$$

where $[\cdot]$ denotes the Iverson bracket. We compute $m$ over all examples in our validation sets for De-En MT and En-De ST and show the layer-wise averages over all examples and attention heads in Figure 5.2. It can be seen that the decoder source attention patterns are more monotonic when using jointly trained hierarchical encoders.

We further examine the source attention parameters in our All-En models to understand the impact that the aforementioned regularization has on multilingual parameter sharing. To do so, we extract sparse subnets for each language pair following the Lottery Ticket Sparse Fine-Tuning proposed by Ansell et al. (2022) and compute the pair-wise sharing across the 6 source languages, as measured by the count of overlapping parameters between subnets. In Figure 5.3, we show the relative change ($\Delta\%$) in multilingual sharing when using hierachical encoding compared to the baseline. The broad increases suggest that the target-orientation of our encoder reduced the decoder's burden of soft-aligning target English outputs to source languages with varying word-orders, allowing for more efficient allocation of modeling capacity.

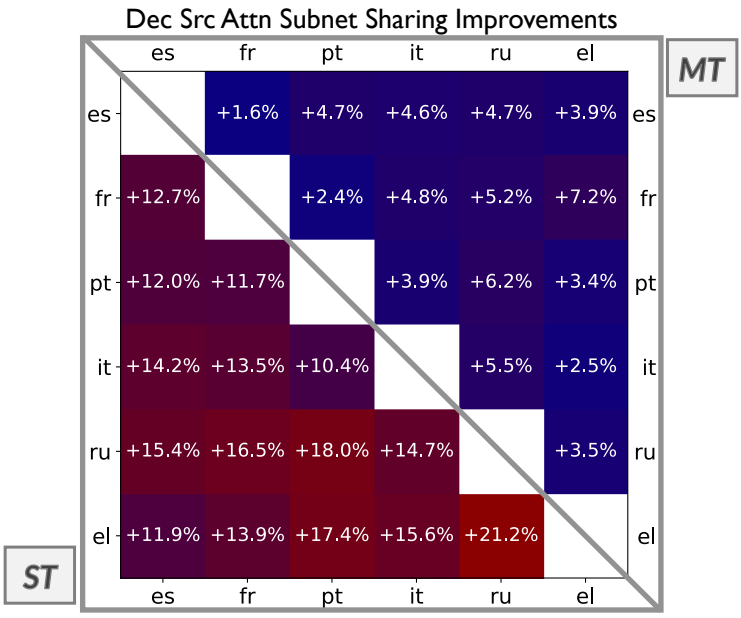



Figure 5.3: Improvement of multilingual sharing in MT/ST decoder source attention parameters when using joint CTC/Attention vs. attention-only training, as measured by pair-wise $\Delta$% in sparse subnet overlap.

### 5.6.3 Benefits of Synchronous Decoding

We examine the generalization effect that augmenting autoregressive likelihoods with conditionally independent likelihoods has during inference, per conjecture (2), by evaluating De-En MT and En-De ST models on out-of-domain EuroParl test sets (Iranzo-Sánchez et al., 2020). As shown in Table 5.2, joint CTC/Attention models outperform pure-attention baselines with consistency across in-domain (In-D) and out-of-domain (Out-D) settings. Further, synchronous joint decoding methods outperform their two-pass re-scoring counterparts (§5.4.2), suggesting that joint selection of the hypothesis set is necessary for easing the respective weaknesses of autoregressive and conditionally independent likelihood estimation.

We also examine the smoothing effect that CTC's alignment information has on length control of hypothesis generation, per conjecture (3), by sliding length penalty. As shown in Figure 5.4, pure-attention MT/ST baselines rapidly degenerate when forced to produce hypotheses that are longer than references which exhibits the label bias with respect to the stop token (§5.3.2). On the other hand, synchronous joint decoding produces gradually longer outputs, suggesting that the output length information inherent in CTC alignments (§5.3.1) eases the decoder's brevity problem.

| Model | Type<br>Decoding Method | MT (De-En)<br>IWSLT14 (In-D) | <br>EuroParl (Out-D) | ST (En-De)<br>MuST-C-v2 (In-D) | <br>EuroParl (Out-D) |
|---|---|---|---|---|---|
| Pure-Attn | Attn O-Sync | 32.8 | 15.8 | 27.8 | 20.5 |
| Joint C/A | Attn O-Sync | 33.6 | 17.1 | 28.3 | 21.0 |
| | +CTC Rescore | 33.6 | 17.1 | 28.3 | 21.0 |
| Joint C/A | Joint O-Sync | **34.1** | **17.6** | **29.2** | **21.7** |
| Joint C/A | CTC I-Sync | 27.3 | 13.1 | 24.4 | 16.5 |
| | +Attn Rescore | 29.5 | 13.9 | 26.2 | 17.8 |
| Joint C/A | Joint I-Sync | **33.7** | **17.4** | **29.2** | **21.1** |

Table 5.2: In/out domain test performances of joint CTC/attention models with various decoding methods.

Table 5.3: Speed vs. accuracy for joint input/output-sync decoding of En-De ST val. set as a fxn. of beam size.

### 5.6.4 Experimental Speed vs. Accuracy

Finally, we perform an experimental validation of our theoretical understanding of the speed vs. accuracy trade-off (§5.4.2.2) between the two synchronous joint decoding variants. To quantify speed, we compute the real-time factor (RTF) as the ratio of decoding time over the duration of input speech. To quantify accuracy beyond the BLEU metric, we compute the search error rate (Meister et al., 2020) by counting the sequences for which the hypothesis has higher exact likelihood (5.7) than the reference. For the same beam size, output is slower but more accurate than input-synchronous.

## 5.7 Discussion and Relation to Prior Work

There is a line of prior work seeking to enhance attentional encoder-decoder MT models with lexical (Alkhouli et al., 2016; Song et al., 2020), syntactical (Wang et al., 2019; Bugliarello and Okazaki, 2020; Deguchi et al., 2021), and phrasal (Watanabe, 2021; Zhang et al., 2021) alignment information derived from external discrete or latent knowledge bases. Our method of hierarchical encoding to enhance the attentional decoder with CTC alignment properties is related, but conceptually different in that we do not incorporate external information.

There is also a line of prior work based on hard alignment frameworks in statisical (Jelinek, 1997; Koehn et al., 2003) and sequence aligning (Graves, 2012; Ghazvininejad et al., 2020a; Sa-

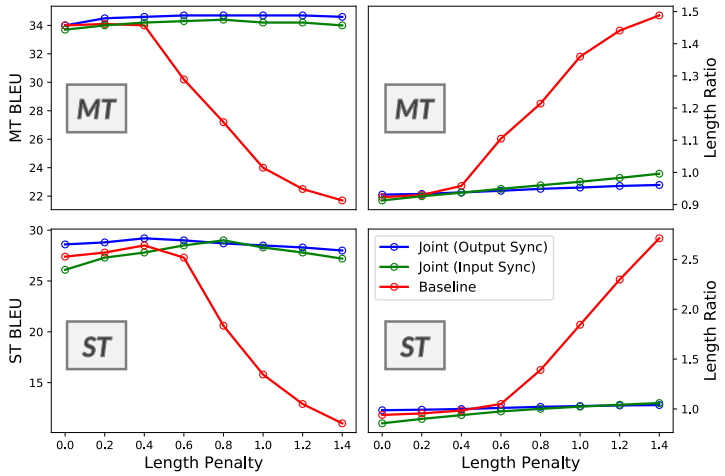



Figure 5.4: Elasticity of BLEU and length ratios (|hyp|/|ref|) w.r.t length penalty in validation sets.

haria et al., 2020) approaches ASR and MT which can be considered to be input-synchronous during their decodings. There are efforts towards integrating neural (Hannun et al., 2014b; Zhou et al., 2022) or n-gram language models (Devlin et al., 2014; Miao et al., 2015) into input-synchronous decoding of these hard alignment models. Our method of joint decoding is a new direction in which we seek to integrate a *conditional* language model (the attentional decoder) without breaking input-synchrony.

## 5.8 Conclusion

In this chapter, we propose to jointly train and decode CTC/Attention models for translation. Our hierarchical encoding method regularizes source attention patterns with monotonicity, allowing for more efficient allocation of attention parameters. Our synchronous decoding method reduces exposure, label, and length biases persistent in autoregressive likelihood estimation, allowing for more generalized and controllable inference. This chapter helps to strengthen the previous chapters 3 and 4. It strengthens Chapter 3 by proposing techniques to build strong CTC encoders for non-monotonic sequence tasks. It also discusses the joint decoding of sequence models that model both CTC and attention distributions, such as the LegoNN models in Chapter 4.

# Part II

## Compositional E2E Systems with Searchable Hidden Intermediates

# Chapter 6

# Searchable Intermediates Framework

The following two chapters aims to build compositional E2E models for complex sequence tasks that can be decomposed into a pipeline of simpler sequence tasks. We show that using our searchable hidden intermediates framework allows composing the simpler sequence tasks into an end-to-end framework to solve the overall complex sequence task. In this chapter, we first introduce the searchable hidden intermediates framework for building compositional E2E systems and apply them towards tasks like speech translation and speech recognition. In the next chapter, we apply our framework to spoken language understanding where we show how this framework enables end-to-end sequence labeling spoken utterances using the token level tagging formulation.

## 6.1 Searchable Hidden Intermediates for End-to-End Models of Decomposable Sequence Tasks

End-to-end approaches for sequence tasks are becoming increasingly popular. Yet for complex sequence tasks, like speech translation, systems that cascade several models trained on sub-tasks have shown to be superior, suggesting that the compositionality of cascaded systems simplifies learning and enables sophisticated search capabilities. In this chapter, we present an end-to-end framework that exploits compositionality to learn *searchable* hidden representations at intermediate stages of a sequence model using decomposed sub-tasks. These hidden intermediates can be improved using beam search to enhance the overall performance and can also incorporate external models at intermediate stages of the network to re-score or adapt towards out-of-domain data. One instance of the proposed framework is a Multi-Decoder model for speech translation that extracts the *searchable hidden intermediates* from a speech recognition sub-task. The model demonstrates the aforementioned benefits and outperforms the previous state-of-the-art by around +6 and +3 BLEU on the two test sets of Fisher-CallHome and by around +3 and +4 BLEU on the

[1] All code and models are released as part of the ESPnet toolkit: https://github.com/espnet/espnet.

English-German and English-French test sets of MuST-C.[1]

## 6.2 Introduction

The principle of compositionality loosely states that a complex whole is composed of its parts and the rules by which those parts are combined (Lake and Baroni, 2018). This principle is present in engineering, where task decomposition of a complex system is required to assess and optimize task allocations (Levis et al., 1994), and in natural language, where paragraph coherence and discourse analysis rely on decomposition into sentences (Johnson, 1992; Kuo, 1995) and sentence level semantics relies on decomposition into lexical units (Liu et al., 2020c).

Similarly, many sequence-to-sequence tasks that convert one sequence into another (Sutskever et al., 2014) can be decomposed to simpler sequence sub-tasks in order to reduce the overall complexity. For example, speech translation systems, which seek to process speech in one language and output text in another language, can be naturally decomposed into the transcription of source language audio through automatic speech recognition (ASR) and translation into the target language through machine translation (MT). Such cascaded approaches have been widely used to build practical systems for a variety of sequence tasks like hybrid ASR (Hinton et al., 2012), phrase-based MT (Koehn et al., 2007), and cascaded ASR-MT systems for speech translation (ST) (Pham et al., 2019).

End-to-end sequence models like encoder-decoder models (Bahdanau et al., 2015; Vaswani et al., 2017), are attractive in part due to their simplistic design and the reduced need for hand-crafted features. However, studies have shown mixed results compared to cascaded models particularly for complex sequence tasks like speech translation (Inaguma et al., 2020b) and spoken language understanding (Coucke et al., 2018). Although direct target sequence prediction avoids the issue of error propagation from one system to another in cascaded approaches (Tzoukermann and Miller, 2018), there are many attractive properties of cascaded systems, missing in end-to-end approaches, that are useful in complex sequence tasks.

In particular, we are interested in (1) the strong search capabilities of the cascaded systems that compose the final task output from individual system predictions (Mohri et al., 2002; Kumar et al., 2006; Beck et al., 2019), (2) the ability to incorporate external models to re-score each individual system (Och and Ney, 2002; Huang and Chiang, 2007), (3) the ability to easily adapt individual components towards out-of-domain data (Koehn and Schroeder, 2007; Peddinti et al., 2015), and finally (4) the ability to monitor performance of the individual systems towards the decomposed sub-task (Tillmann and Ney, 2003; Meyer et al., 2016).

In this chapter, we seek to incorporate these properties of cascaded systems into end-to-end sequence models. We first propose a generic framework to learn *searchable hidden intermediates* using an auto-regressive encoder-decoder model for any decomposable sequence task (§6.4). We then apply this approach to speech translation, where the intermediate stage is the output of ASR,

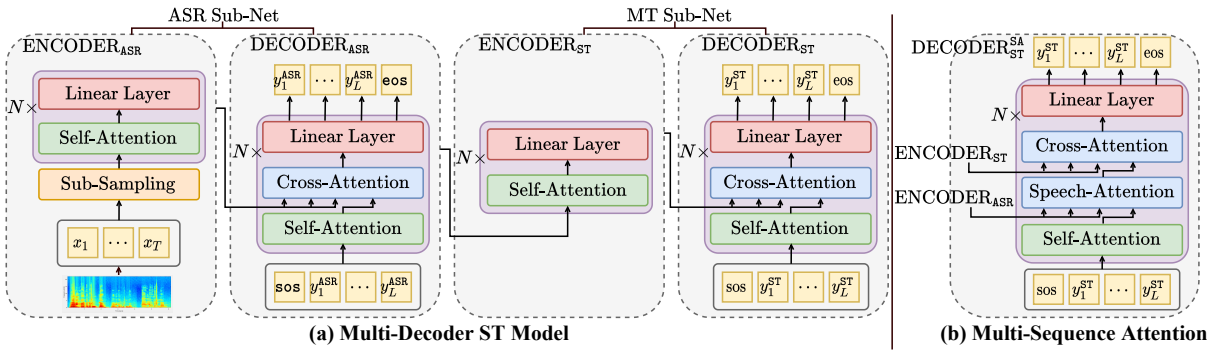



Figure 6.1: The left side present the schematics and the information flow of our proposed framework applied to ST, in a model we call the Multi-Decoder. Our model decomposes ST into ASR and MT sub-nets, each of which consist of an encoder and decoder. The right side displays a Multi-Sequence Attention variant of the DECODER$_{ST}$ that is conditioned on both speech information via the ENCODER$_{ASR}$ and transcription information via the ENCODER$_{ST}$.

by passing continuous hidden representations of discrete transcript sequences from the ASR sub-net decoder to the MT sub-net encoder. By doing so, we gain the ability to use beam search with optional external model re-scoring on the hidden intermediates, while maintaining end-to-end differentiability. Next, we suggest mitigation strategies for the error propagation issues inherited from decomposition.

We show the efficacy of *searchable intermediate representations* in our proposed model, called the Multi-Decoder, on speech translation with a 5.4 and 2.8 BLEU score improvement over the previous state-of-the-arts for Fisher and CallHome test sets respectively (§6.7). We extend these improvements by an average of 0.5 BLEU score through the aforementioned benefit of re-scoring the intermediate search with external models trained on the same dataset. We also show a method for monitoring sub-net performance using oracle intermediates that are void of search errors (§6.7.1). Finally, we show how these models can adapt to out-of-domain speech translation datasets, how our approach can be generalized to other sequence tasks like speech recognition, and how the benefits of decomposition persist even for larger corpora like MuST-C (§6.7.2).

## 6.3 Background and Motivation

### 6.3.1 Compositionality in Sequences Models

The probabilistic space of a sequence is combinatorial in nature, such that a sentence of $L$ words from a fixed vocabulary $\mathcal{V}$ would have an output space $\mathcal{S}$ of size $|\mathcal{V}|^L$. In order to deal with this combinatorial output space, an output sentence is decomposed into labeled target tokens,

$\mathbf{y} = (y_1, y_2, \ldots, y_L)$, where $y_l \in \mathcal{V}$.

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{L} P(y_i \mid \mathbf{x}, y_{1:i\text{-}1})$$

An auto-regressive encoder-decoder model uses the above probabilistic decomposition in sequence-to-sequence tasks to learn next word prediction, which outputs a distribution over the next target token $y_l$ given the previous tokens $y_{1:l\text{-}1}$ and the input sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_t, \ldots, \mathbf{x}_T)$, where $T$ is the input sequence length. In the next sub-section we detail the training and inference of these models.

### 6.3.2 Auto-regressive Encoder-Decoder Models

**Training:** In an auto-regressive encoder-decoder model, the ENCODER maps the input sequence $\mathbf{x}$ to a sequence of continuous hidden representations $\mathbf{h}^E = (\mathbf{h}_1^E, \mathbf{h}_t^E, \ldots, \mathbf{h}_T^E)$, where $\mathbf{h}_t^E \in \mathbb{R}^d$. The DECODER then auto-regressively maps $\mathbf{h}^E$ and the preceding ground-truth output tokens, $\hat{y}_{1:l\text{-}1}$, to $\mathbf{h}_l^D$, where $\mathbf{h}_l^D \in \mathbb{R}^d$. The sequence of decoder hidden representations form $\mathbf{h}^D = (\mathbf{h}_1^D, \mathbf{h}_l^D, \ldots, \mathbf{h}_L^D)$ and the likelihood of each output token $y_l$ is given by SOFTMAXOUT, which denotes an affine projection of $\mathbf{h}_l^D$ to $\mathcal{V}$ followed by a softmax function.

$$\mathbf{h}^E = \textsc{Encoder}(\mathbf{x})$$
$$\hat{\mathbf{h}}_l^D = \textsc{Decoder}(\mathbf{h}^E, \hat{y}_{1:l\text{-}1}) \tag{6.1}$$
$$P(y_l \mid \hat{y}_{1:l\text{-}1}, \mathbf{h}^E) = \textsc{SoftmaxOut}(\hat{\mathbf{h}}_l^D) \tag{6.2}$$

During training, the DECODER performs token classification for next word prediction by considering only the ground truth sequences for previous tokens $\hat{\mathbf{y}}$. We refer to this $\hat{\mathbf{h}}^D$ as *oracle* decoder representations, which will be discussed later.

**Inference:** During inference, we can maximize the likelihood of the entire sequence from the output space $\mathcal{S}$ by composing the conditional probabilities of each step for the $L$ tokens in the sequence.

$$\mathbf{h}_l^D = \textsc{Decoder}(\mathbf{h}^E, y_{1:l\text{-}1}) \tag{6.3}$$
$$P(y_l \mid \mathbf{x}, y_{1:l\text{-}1}) = \textsc{SoftmaxOut}(\mathbf{h}_l^D)$$
$$\bar{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{S}} \prod_{i=1}^{L} P(y_i \mid \mathbf{x}, y_{1:i\text{-}1}) \tag{6.4}$$

This is an intractable search problem and it can be approximated by either greedily choosing argmax at each step or using a search algorithm like beam search to approximate $\bar{\mathbf{y}}$. Beam search

(Reddy, 1988) generates candidates at each step and prunes the search space to a tractable beam size of $B$ most likely sequences. As $B \rightarrow \infty$, the beam search result would be equivalent to equation 6.4.

$$\begin{aligned} \textsc{GreedySearch} &:= \underset{y_l}{\text{argmax}}\, P(y_l \mid \mathbf{x}, y_{1:l\text{-}1}) \\ \textsc{BeamSearch} &:= \textsc{Beam}(P(y_l \mid \mathbf{x}, y_{1:l\text{-}1})) \end{aligned}$$

In approximate search for auto-regressive models, like beam search, the Decoder receives alternate candidates of previous tokens to find candidates with a higher likelihood as an overall sequence. This also allows for the use of external models like Language Models (LM) or Connectionist Temporal Classification Models (CTC) for re-scoring candidates (Hori et al., 2017).

## 6.4 Proposed Framework

In this section, we present a general framework to exploit natural decompositions in sequence tasks which seek to predict some output $\mathcal{C}$ from an input sequence $\mathcal{A}$. If there is an intermediate sequence $\mathcal{B}$ for which $\mathcal{A} \rightarrow \mathcal{B}$ sequence transduction followed by $\mathcal{B} \rightarrow \mathcal{C}$ prediction achieves the original task, then the original $\mathcal{A} \rightarrow \mathcal{C}$ task is decomposable.

In other words, if we can learn $P(\mathcal{B} \mid \mathcal{A})$ then we can learn the overall task of $P(\mathcal{C} \mid \mathcal{A})$ through $\max_{\mathcal{B}}(P(\mathcal{C} \mid \mathcal{A},\ \mathcal{B})P(\mathcal{B} \mid \mathcal{A}))$, approximated using Viterbi search. We define a first encoder-decoder $\textsc{Sub}_{\mathcal{A}\rightarrow\mathcal{B}}\textsc{Net}$ to map an input sequence $\mathcal{A}$ to a sequence of decoder hidden states, $\mathbf{h}^{D_\mathcal{B}}$. Then we define a subsequent $\textsc{Sub}_{\mathcal{B}\rightarrow\mathcal{C}}\textsc{Net}$ to map $\mathbf{h}^{D_\mathcal{B}}$ to the final probabilistic output space of $\mathcal{C}$. Therefore, we call $\mathbf{h}^{D_\mathcal{B}}$ *hidden intermediates*. The following equations shows the two sub-networks of our framework, $\textsc{Sub}_{\mathcal{A}\rightarrow\mathcal{B}}\textsc{Net}$ and $\textsc{Sub}_{\mathcal{B}\rightarrow\mathcal{C}}\textsc{Net}$, which can be trained end-to-end while also exploiting compositionality in sequence tasks. [2]

**$\textsc{Sub}_{\mathcal{A}\rightarrow\mathcal{B}}\textsc{Net}$:**

$$\begin{aligned} \mathbf{h}^E &= \textsc{Encoder}_\mathcal{A}(\mathcal{A}) \\ \hat{\mathbf{h}}_l^{D_\mathcal{B}} &= \textsc{Decoder}_\mathcal{B}(\mathbf{h}^E, \hat{\mathbf{y}}^\mathcal{B}_{1:l\text{-}1}) \\ P(y_l^\mathcal{B} \mid \hat{\mathbf{y}}^\mathcal{B}_{1:l\text{-}1}, \mathbf{h}^E) &= \textsc{SoftmaxOut}(\hat{\mathbf{h}}_l^{D_\mathcal{B}}) \end{aligned} \tag{6.5}$$

**$\textsc{Sub}_{\mathcal{B}\rightarrow\mathcal{C}}\textsc{Net}$:**

$$P(\mathcal{C} \mid \hat{\mathbf{h}}_l^{D_\mathcal{B}}) = \textsc{Sub}_{\mathcal{B}\rightarrow\mathcal{C}}\textsc{Net}(\hat{\mathbf{h}}_l^{D_\mathcal{B}}) \tag{6.6}$$

[2] Note that this framework does not use locally-normalized softmax distributions but rather the hidden representations, thereby avoiding label bias issues when combining multiple sub-systems (Bottou et al., 1997; Wiseman and Rush, 2016).

Note that the final prediction, given by equation 6.6, does not need to be a sequence and can be a categorical class like in spoken language understanding tasks. Next we will show how the *hidden intermediates* become *searchable* during inference.

### 6.4.1 Searchable Hidden Intermediates

As stated in section §6.3.2, approximate search algorithms maximize the likelihood, $P(\mathbf{y} \mid \mathbf{x})$, of the entire sequence by considering different candidates $y_l$ at each step. Candidate-based search, particularly in auto-regressive encoder-decoder models, also affects the decoder hidden representation, $\mathbf{h}^D$, as these are directly dependent on the previous candidate (refer to equations 6.1 and 6.3). This implies that by searching for better approximations of the previous predicted tokens, $\mathbf{y}_{l\text{-}1} = (\mathbf{y}_{\text{BEAM}})_{l\text{-}1}$, we also improve the decoder hidden representations for the next token, $\mathbf{h}^D_l = (\mathbf{h}^D_{\text{BEAM}})_l$. As $\mathbf{y}_{\text{BEAM}} \rightarrow \hat{\mathbf{y}}$, the decoder hidden representations tend to the *oracle* decoder representations that have only errors from next word prediction, $\mathbf{h}^D_{\text{BEAM}} \rightarrow \hat{\mathbf{h}}^D$. A perfect search is analogous to choosing the ground truth $\hat{y}$ at each step, which would yield $\hat{\mathbf{h}}^D$.

We apply this beam search of hidden intermediates, thereby approximating $\hat{\mathbf{h}}^{D_\mathcal{B}}$ with $\mathbf{h}^{D_\mathcal{B}}_{\text{BEAM}}$. This process is illustrated in algorithm 2, which shows beam search for $\mathbf{h}^{D_\mathcal{B}}_{\text{BEAM}}$ that are subsequently passed to the $\text{SUB}_{\mathcal{B}\rightarrow\mathcal{C}}\text{NET}$.[3] In line 7, we show how an external model like an LM or a CTC model can be used to generate an alternate sequence likelihood, $P_{\text{EXT}}(\mathbf{y}^\mathcal{B}_l)$, which can be combined with the $\text{SUB}_{\mathcal{A}\rightarrow\mathcal{B}}\text{NET}$ likelihood, $P_\mathcal{B}(\mathbf{y}^\mathcal{B}_l \mid \mathbf{x})$ , with a tunable parameter $\lambda$.

We can monitor the performance of the $\text{SUB}_{\mathcal{A}\rightarrow\mathcal{B}}\text{NET}$ by comparing the decoded intermediate sequence $\mathbf{y}^\mathcal{B}_{\text{BEAM}}$ to the ground truth $\hat{\mathbf{y}}^\mathcal{B}$. We can also monitor the $\text{SUB}_{\mathcal{B}\rightarrow\mathcal{C}}\text{NET}$ performance by using the aforementioned *oracle* representations of the intermediates, $\hat{\mathbf{h}}^{D_\mathcal{B}}$, which can be obtained by feeding the ground truth $\hat{\mathbf{y}}^\mathcal{B}$ to $\text{DECODER}_\mathcal{B}$. By passing $\hat{\mathbf{h}}^{D_\mathcal{B}}$ to $\text{SUB}_{\mathcal{B}\rightarrow\mathcal{C}}\text{NET}$, we can observe its performance in a vacuum, i.e. void of search errors in the hidden intermediates.

### 6.4.2 Multi-Decoder Model

In order to show the applicability of our end-to-end framework we propose our Multi-Decoder model for speech translation. This model predicts a sequence of text translations $\mathbf{y}^{\text{ST}}$ from an input sequence of speech $\mathbf{x}$ and uses a sequence of text transcriptions $\mathbf{y}^{\text{ASR}}$ as an intermediate. In this case, the $\text{SUB}_{\mathcal{A}\rightarrow\mathcal{B}}\text{NET}$ in equation 6.5 is specified as the ASR sub-net and the $\text{SUB}_{\mathcal{B}\rightarrow\mathcal{C}}\text{NET}$ in equation 6.6 is specified as the MT sub-net. Since the MT sub-net is also a sequence prediction task, both sub-nets are encoder-decoder models in our architecture (Bahdanau et al., 2015; Vaswani et al., 2017). In Figure 6.1 we illustrate the schematics of our transformer based Multi-Decoder

[3]The algorithm shown only considers a single top approximation of the search; however, with added time-complexity, the final task prediction improves with the n-best $\mathbf{h}^{D_\mathcal{B}}_{\text{BEAM}}$ for selecting the best resultant $\mathcal{C}$.

**Algorithm 2** Beam Search for Hidden Intermediates: We perform beam search to approximate the most likely sequence for the sub-task $\mathcal{A} \rightarrow \mathcal{B}$, $\mathbf{y}^{\mathcal{B}}_{\text{BEAM}}$, while collecting the corresponding $\text{DECODER}_{\mathcal{B}}$ hidden representations, $\mathbf{h}^{D_{\mathcal{B}}}_{\text{BEAM}}$. The output $\mathbf{h}^{D_{\mathcal{B}}}_{\text{BEAM}}$, is passed to the final sub-network to predict final output $\mathcal{C}$ and $\mathbf{y}^{\mathcal{B}}_{\text{BEAM}}$ is used for monitoring performance on predicting $\mathcal{B}$.

**Initialize:** BEAM $\leftarrow$ {sos}; k $\leftarrow$ beam size;
$\mathbf{h}^{E_{\text{A}}} \leftarrow \text{ENCODER}_{\mathcal{A}}(\mathbf{x})$
**for** $l$=1 **to** $\max_{\text{STEPS}}$ **do**
  **for** $\mathbf{y}^{\mathcal{B}}_{l\text{-}1} \in$ BEAM **do**
    $\mathbf{h}^{D_{\mathcal{B}}}_{l} \leftarrow \text{DECODER}_{\mathcal{B}}(\mathbf{h}^{E_{\mathcal{A}}}, \mathbf{y}^{\mathcal{B}}_{l\text{-}1})$
    **for** $\mathbf{y}^{\mathcal{B}}_{l} \in \mathbf{y}^{\mathcal{B}}_{l\text{-}1} + \{\mathcal{V}\}$ **do**
      $s_l \leftarrow P_{\mathcal{A}\rightarrow\mathcal{B}}(\mathbf{y}^{\mathcal{B}}_{l} \mid \mathbf{x})^{1\text{-}\lambda} P_{\text{EXT}}(\mathbf{y}^{\mathcal{B}}_{l})^{\lambda}$
      $\mathcal{H} \leftarrow (s_l, \mathbf{y}^{\mathcal{B}}_{l}, \mathbf{h}^{D_{\mathcal{B}}}_{l})$
    **end for**
  **end for**
  BEAM $\leftarrow \arg^{\text{k}}\max(\mathcal{H})$
**end for**
$(s^{\mathcal{B}}, \mathbf{y}^{\mathcal{B}}_{\text{BEAM}}, \mathbf{h}^{D_{\mathcal{B}}}_{\text{BEAM}}) \leftarrow \text{argmax}(\text{BEAM})$
**Return** $\mathbf{y}^{\mathcal{B}}_{\text{BEAM}} \rightarrow \text{SUB}_{\mathcal{A}\rightarrow\mathcal{B}}\text{NET}$ Monitoring
**Return** $\mathbf{h}^{D_{\mathcal{B}}}_{\text{BEAM}} \rightarrow$ Final $\text{SUB}_{\mathcal{B}\rightarrow\mathcal{C}}\text{NET}$

ST model which can also be summarized as follows:

$$\mathbf{h}^{E_{\text{ASR}}} = \text{ENCODER}_{\text{ASR}}(\mathbf{x}) \tag{6.7}$$

$$\hat{\mathbf{h}}^{D_{\text{ASR}}}_{l} = \text{DECODER}_{\text{ASR}}(\mathbf{h}^{E_{\text{ASR}}}, \hat{y}^{\text{ASR}}_{1:l\text{-}1}) \tag{6.8}$$

$$\mathbf{h}^{E_{\text{ST}}} = \text{ENCODER}_{\text{ST}}(\hat{\mathbf{h}}^{D_{\text{ASR}}}) \tag{6.9}$$

$$\hat{\mathbf{h}}^{D_{\text{ST}}}_{l} = \text{DECODER}_{\text{ST}}(\mathbf{h}^{E_{\text{ST}}}, \hat{y}^{\text{ST}}_{1:l\text{-}1}) \tag{6.10}$$

As we can see from Equations 6.9 and 6.10, the MT sub-network attends only to the decoder representations, $\hat{\mathbf{h}}^{D_{\text{ASR}}}$, of the ASR sub-network, which could lead to the error propagation issues from the ASR sub-network to the MT sub-network similar to the cascade systems, as mentioned in §6.2. To alleviate this problem, we modify equation 6.10 such that $\text{DECODER}_{\text{ST}}$ attends to both $\mathbf{h}^{E_{\text{ST}}}$ and $\mathbf{h}^{E_{\text{ASR}}}$:

$$\hat{\mathbf{h}}^{D^{\text{SA}}_{\text{ST}}}_{l} = \text{DECODER}^{\text{SA}}_{\text{ST}}(\mathbf{h}^{E_{\text{ST}}}, \mathbf{h}^{E_{\text{ASR}}}, \hat{y}^{\text{ST}}_{1:l\text{-}1}) \tag{6.11}$$

We use the multi-sequence cross-attention discussed by Helcl et al. (2018), shown on the right side of Figure 6.1, to condition the final outputs generated by $\hat{\mathbf{h}}^{D_{\text{ST}}}_{l}$ on both speech and transcript information in an attempt to allow our network to recover from intermediate mistakes during inference. We call this model the Multi-Decoder w/ Speech-Attention.

## 6.5 Baseline Encoder-Decoder Model

For our baseline model, we use an end-to-end encoder-decoder (Enc-Dec) ST model with ASR joint training (Inaguma et al., 2020b) as an auxiliarly loss to the speech encoder. In other words, the model consumes speech input using the $\text{ENCODER}_{\text{ASR}}$, to produce $\mathbf{h}^{E_{\text{ASR}}}$, which is used for cross-attention by $\text{DECODER}_{\text{ASR}}$ and the $\text{DECODER}_{\text{ST}}$. Using the decomposed ASR task as an auxiliary loss also helps the baseline Enc-Dec model and provide strong baseline performance, as we will see in Section 6.7.

## 6.6 Data and Experimental Setup

**Data:** We demonstrate the efficacy of our proposed approach on ST in the Fisher-CallHome corpus (Post et al., 2013) which contains 170 hours of Spanish conversational telephone speech, transcriptions, and English translations. All punctuations except apostrophes were removed and results are reported in terms of detokenized case-insensitive BLEU (Papineni et al., 2002; Post, 2018). We compute BLEU using the 4 references in Fisher (dev, dev2, and test) and the single reference in CallHome (dev and test) (Post et al., 2013; Kumar et al., 2014; Weiss et al., 2017). We use a joint source and target vocabulary of 1K byte pair encoding (BPE) units (Kudo and Richardson, 2018).

We prepare the corpus using the ESPnet library and we follow the standard data preparation, where inputs are globally mean-variance normalized log-mel filterbank and pitch features from up-sampled 16kHz audio (Watanabe et al., 2018). We also apply speed perturbations of 0.9 and 1.1 and the SS SpecAugment policy (Park et al., 2019).

**Baseline Configuration:** All of our models are implemented using the ESPnet library and trained on 3 NVIDIA Titan 2080Ti GPUs for ≈12 hours. For the Baseline Enc-Dec baseline, discussed in §6.5, we use an $\text{ENCODER}_{\text{ASR}}$ consisting of a convolutional sub-sampling by a factor of 4 (Watanabe et al., 2018) and 12 transformer encoder blocks with 2048 feed-forward dimension, 256 attention dimension, and 4 attention heads. The $\text{DECODER}_{\text{ASR}}$ and $\text{DECODER}_{\text{ST}}$ both consist of 6 transformer decoder blocks with the same configuration as $\text{ENCODER}_{\text{ASR}}$. There are 37.9M trainable parameters. We apply dropout of 0.1 for all components, detailed in the Appendix (C.1).

We train our models using an effective batch-size of 384 utterances and use the Adam optimizer (Kingma and Ba, 2014) with inverse square root decay learning rate schedule. We set learning rate to 12.5, warmup steps to 25K, and epochs to 50. We use joint training with hybrid CTC/attention ASR (Watanabe et al., 2017) by setting mtl-alpha to 0.3 and asr-weight to 0.5 as defined by Watanabe et al. (2018). During inference, we perform beam search (Seki et al., 2019) on the ST sequences, using a beam size of 10, length penalty of 0.2, max length ratio of 0.3 (Watanabe et al., 2018).

| Model Type | Model Name | Uses Speech Transcripts | Fisher dev(↑) | Fisher dev2(↑) | Fisher test(↑) | CallHome dev(↑) | CallHome test(↑) |
|---|---|---|---|---|---|---|---|
| Cascade | Inaguma et al. (2020b) | ✓ | 41.5 | 43.5 | 42.2 | **19.6** | **19.8** |
| Cascade | ESPnet ASR+MT 2018 | ✓ | **50.4** | **51.2** | **50.7** | **19.6** | 19.2 |
| Enc-Dec | Weiss et al. (2017) ♢ | ✗ | 46.5 | 47.3 | 47.3 | 16.4 | 16.6 |
| Enc-Dec | Weiss et al. (2017) ♢ | ✓ | 48.3 | 49.1 | 48.7 | 16.8 | 17.4 |
| Enc-Dec | Inaguma et al. (2020b) | ✓ | 46.6 | 47.6 | 46.5 | 16.8 | 16.8 |
| Enc-Dec | Guo et al. (2020) | ✓ | 48.7 | 49.6 | 47.0 | 18.5 | **18.6** |
| Enc-Dec | Our Implementation | ✓ | **49.6** | **50.9** | **49.5** | **19.1** | 18.2 |
| Multi-Decoder | Our Proposed Model | ✓ | 52.7 | 53.3 | 52.6 | 20.5 | 20.1 |
| Multi-Decoder | +ASR Re-scoring | ✓ | 53.3 | 54.2 | 53.7 | 21.1 | 20.8 |
| Multi-Decoder | +Speech-Attention | ✓ | **54.6** | **54.6** | **54.1** | **21.7** | **21.4** |
| Multi-Decoder | +ASR Re-scoring | ✓ | **55.2** | **55.2** | **55.0** | **21.7** | **21.5** |

Table 6.1: Results presenting the overall performance (BLEU) of our proposed multi-decoder model. Cascade and Enc-Dec results from previous papers and our own implementation of the Enc-Dec are shown for comparison. The best performing models are **highlighted**. ♢Implemented with LSTM, while all others are Transformer-based.

**Multi-Decoder Configuration:** For the Multi-Decoder ST model, discussed in §6.4, we use the same transformer configuration as the baseline for the $\text{Encoder}_{\text{ASR}}$, $\text{Decoder}_{\text{ASR}}$, and $\text{Decoder}_{\text{ST}}$. Additionally, the Multi-Decoder has an $\text{Encoder}_{\text{ST}}$ consisting of 2 transformer encoder blocks with the same configuration as $\text{Encoder}_{\text{ASR}}$, giving a total of 40.5M trainable parameters. The training configuration is also the same as for the baseline. For the Multi-Decoder w/ Speech-Attention model (42.1M trainable parameters), we increase the attention dropout of the ST decoder to 0.4 and dropout on all other components of the ST decoder to 0.2 while keeping dropout on the remaining components at 0.1. We verified that increasing the dropout does not help the vanilla multi-decoder ST model.

During inference, we perform beam search on both the ASR and ST output sequences, as discussed in §6.4. The ST beam search is identical to that of the baseline. For the intermediate ASR beam search, we use a beam size of 16, length penalty of 0.2, max length ratio of 0.3. In some of our experiments, we also include fusion of a source language LM with a 0.2 weight and CTC with a 0.3 weight to re-score the intermediate ASR beam search (Watanabe et al., 2017). For the Speech-Attention variant, we increase LM weight to 0.4.

Note that the ST beam search configuration remains constant across our baseline and Multi-Decoder experiments as our focus is on improving overall performance through searchable intermediate representations. Thus, the various re-scoring techniques applied to the ASR beam search are options newly enabled by our proposed architecture and are not used in the ST beam search.

| Model | Overall ST(↑) | Sub-Net ASR(↓) | Sub-Net MT(↑) |
|---|---|---|---|
| Multi-Decoder | 52.7 | 22.6 | 64.9 |
| +Speech-Attention | 54.6 | 22.4 | 66.6 |

Table 6.2: Results presenting the overall ST performance (BLEU) of our Multi-Decoder models, along with their sub-net ASR (% WER) and MT (BLEU) performances. All results are from the Fisher dev set.

## 6.7 Results

Table 6.1 presents the overall ST performance (BLEU) of our proposed Multi-Decoder model. Our model improves by +2.9/+0.3 (Fisher/CallHome) over the best cascaded baseline and by +5.6/+1.5 BLEU over the best published end-to-end baselines. With Speech-Attention, our model improves by +3.4/+1.6 BLEU over the cascaded baselines and +7.1/+2.8 BLEU over encoder-decoder baselines. Both the Multi-Decoder and Multi-Decoder w/ Speech-Attention on average are further improved by +0.9/+0.4 BLEU through ASR re-scoring.[4]

Table 6.1 also includes our implementation of the Baseline Enc-Dec model discussed in §6.5. In this way, we are able to make a fair comparison with our framework as we control the model and inference configurations to be analagous. For instance, we keep the same search parameters for the final output in the baseline and the Multi-Decoder to demonstrate impact of the intermediate beam search.

### 6.7.1 Benefits

#### 6.7.1.1 Sub-network performance monitoring

An added benefit of our proposed approach over the Baseline Enc-Dec is the ability to monitor the individual performances of the ASR (BLEU) and MT (% WER) sub-nets as shown in Table 6.2. The Multi-Decoder w/ Speech-Attention shows a greater MT sub-net performance than the Multi-Decoder as well as a slight improvement of the ASR sub-net, suggesting that ST can potentially help ASR.

#### 6.7.1.2 Beam search for better intermediates

The overall ST performance improves when a higher beam size is used in the intermediate ASR search, and this increase can be attributed to the improved ASR sub-net performance. Figure 2 shows this trend across ASR beam sizes of 1, 4, 8, 10, 16 while fixing the ST decoding beam size

[4]We also evaluate our models using other MT metrics to supplement these results, as shown in the Appendix (C.1.1).

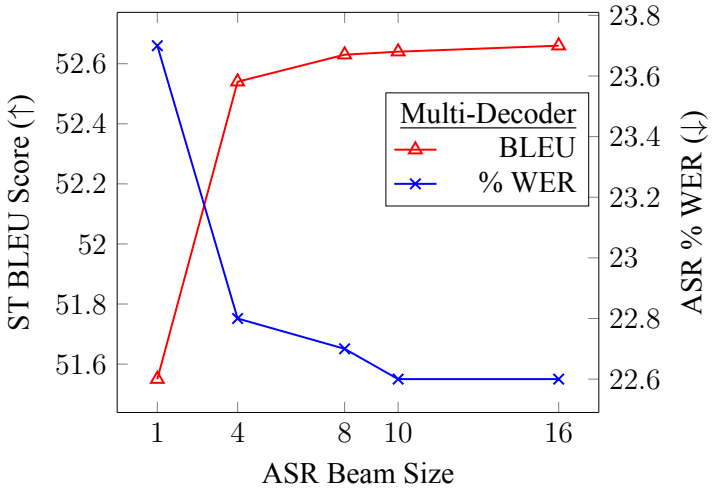



Figure 6.2: Results studying the effect of the different ASR beam sizes in the intermediate representation search on the overall ST performance (BLEU) and the ASR sub-net performance (% WER) for our multi-decoder model. Beam of 1 is same as greedy search.

to 10. A beam size of 1, which is a greedy search, results in lower ASR sub-net and overall ST performances. As beam sizes become larger, gains taper off as can be seen between beam sizes of 10 and 16.

#### 6.7.1.3 External models for better search

External models like CTC acoustic models and language models are commonly used for re-scoring encoder-decoder models (Hori et al., 2017), due to the difference in their modeling capabilities. CTC directly models transcripts while being conditionally independent on the other outputs given the input, and LMs predict the next token in a sequence.

Both variants of the Multi-Decoder improve due to improved ASR sub-net performance using external CTC and LM models for re-scoring, as shown in Table 6.3. We use a recurrent neural network LM trained on the Fisher-CallHome Spanish transcripts with a dev perplexity of 18.8 and the CTC model from joint loss applied during training. Neither external model incorporates additional data. Although the impact of the LM-only re-scoring is not shown in the ASR % WER, it reduces substitution and deletion rates in the ASR and this is observed to help the overall ST performance.

#### 6.7.1.4 Error propagation avoidance

As discussed in §6.4, our Multi-Decoder model inherits the error propagation issue as can be seen in Figure 6.3. For the easiest bucket of utterances with $< 40\%$ WER in Multi-Decoder's ASR sub-net, our model's ST performance, as measured by the corpus BLEU of the bucket, exceeds

| Model | Overall ST(↑) | Sub-Net ASR(↓) |
|---|---|---|
| Multi-Decoder | 52.7 | 22.6 |
| +ASR Rescoring w/ LM | 53.2 | 22.6 |
| +ASR Rescoring w/ CTC | 52.8 | 22.1 |
| +ASR Rescoring w/ LM | **53.3** | **21.7** |
| Multi-Decoder w/ Speech-Attn. | 54.6 | 22.4 |
| +ASR Rescoring w/ LM | 55.1 | 22.4 |
| +ASR Rescoring w/ CTC | 54.7 | 22.0 |
| +ASR Rescoring w/ LM | **55.2** | **21.9** |

Table 6.3: Results presenting the overall ST performance (BLEU) and the sub-net ASR (% WER) of our Multi-Decoder models with external CTC and LM re-scoring in the ASR intermediate representation search. All results are from the Fisher dev set.

that of the Baseline Enc-Dec. The inverse is true for the more difficult bucket of $[40, 80)$%, showing that error propagation is limiting the performance of our model; however, we show that multi-sequence attention can alleviate this issue. For extremely difficult utterances in the $\geq 80$% bucket, ST performance for all three approaches is suppressed.

#### 6.7.1.5 Qualitative Examples of Error Propagation Avoidance

To supplement our qualitative analysis of the error propagation avoidance of the Multi-Decoder with Speech-Attention model in §6.7.1.4, we also show four qualitative examples in Table 6.4. In the first three examples, the Multi-Decoder and Multi-Decoder with Speech-Attention models both make the same mistakes in the ASR portion of Spanish-English translation, but the model with Speech-Attention recovers by producing correct English translations despite mistakes in the Spanish transcription. On the other hand, the model without Speech-Attention propagates the Spanish transcription errors into English translation errors. In the fourth example only the Multi-Decoder w/ Speech-Attention makes a mistake in Spanish transcription, but the English translation still recovers.

### 6.7.2 Generalizability

In this section, we discuss the generalizability of our framework towards out-of-domain data. We also extend our Multi-Decoder model to other sequence tasks like speech recognition. Finally, we apply our ST models to a larger corpus with more language pairs and a different domain of speech.

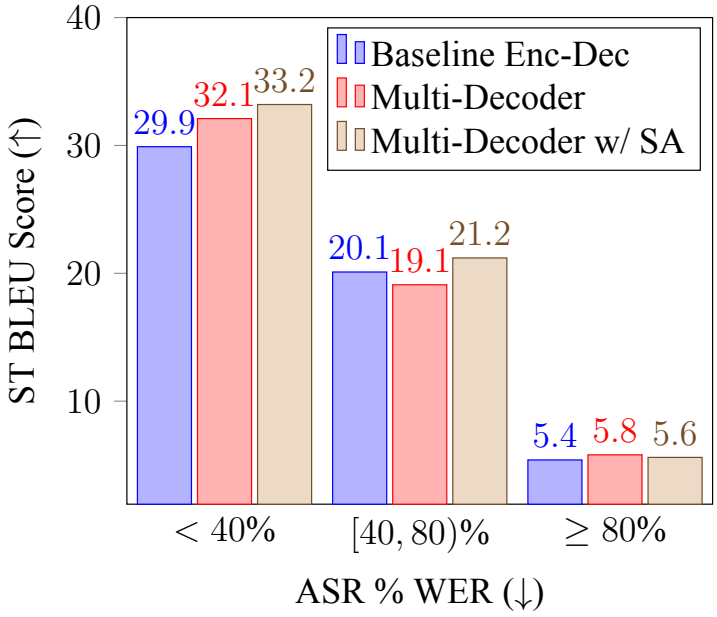



Figure 6.3: Results comparing the ST performances (BLEU) of our Baseline Enc-Dec, Multi-Decoder, and Multi-Decoder w/ Speech-Attention across different ASR difficulties measured using % WER on the Fisher dev set (1-ref). The buckets on the x-axis are determined using the utterance level % WER using the Multi-Decoder ASR sub-net performance.

#### 6.7.2.1 Robustness through Decomposition

Like cascaded systems, searchable intermediates provide our model adaptability in individual sub-systems towards out-of-domain data using external in-domain language model, thereby giving access to more in-domain data. Specifically for speech translation systems, this means we can use in-domain language models in both source and target languages. We test the robustness of our Multi-Decoder model trained on Fisher-CallHome conversational speech dataset on read speech CoVost-2 dataset (Wang et al., 2020b). In Table 6.5 we show that re-scoring the ASR sub-net with an in-domain LM improves ASR with around 10.0% lower WER, improving the overall ST performance by around +2.5 BLEU. Compared to an in-domain ST baseline (Wang et al., 2020a), our out-of-domain Multi-Decoder with in-domain ASR re-scoring demonstrates the robustness of our approach.

#### 6.7.2.2 Decomposing Speech Transcripts

We apply our generic framework to another decomposable sequence task, speech recognition, and show the results of various levels of decomposition in Table 6.6. We show that with phoneme, character, or byte-pair encoding (BPE) sequences as intermediates, the Multi-Decoder presents strong results on both Fisher and CallHome test sets. We also observe that the BPE intermediates perform better than phoneme/character variants, which could be attributed to the reduced search

| Model / Source | ASR Output | ST Output |
|---|---|---|
| Ground-Truth | … porque tengo a mis dos hijos acá | … because i have my two children here |
| Multi-Decoder | … porque tengo mis dos hijos acá | … because i have two kids here |
| +Speech-Attention | … porque tengo mis dos hijos acá | … because i have my two children here |
| Ground-Truth | puedes ayudar para que se haga justicia más rápido | you can help so that justice is served quickly |
| Multi-Decoder | puedes ayudar para que sea justicia más rápido | you can help so it's faster |
| +Speech-Attention | puedes ayudar para que sea justicia más rápido | you can help so that it's faster justice |
| Ground-Truth | pero tiene muchas cosas muy bonitas | but there are many beautiful things |
| Multi-Decoder | pero tienen muchas cosas muy bonitas | but they have a lot of nice things |
| +Speech-Attention | pero tienen muchas cosas muy bonitas | but there are many very beautiful things |
| Ground-Truth | acampar ir a pescar y ir a las montañas a esquiar | camping and fishing and going to the mountains to ski |
| Multi-Decoder | acampar y a pescar y y de las montañas esquiar | camping and fishing and and the mountains skiing |
| +Speech-Attention | a campar y ir a pescar y ir a las montañas a esquiar | camping and go fishing and go to the mountains to ski |

Table 6.4: Examples where the Multi-Decoder and Multi-Decoder w/ Speech-Attention models make errors in the ASR portion of Spanish-English ST. In these cases the Speech-Attention component alleviates ASR error propagation, producing correct translations despite mistakes in transcription. Words that are transcribed/translated correctly are highlighted in green and those that are incorrect are in pink.

capabilities of encoder-decoder models using beam search on longer sequences (Sountsov and Sarawagi, 2016) like in phoneme/character sequences.

#### 6.7.2.3 Extending to MuST-C Language Pairs

In addition to our results using the 170 hours of the Spanish-English Fisher-CallHome corpus, in Table 6.7 we show that our decompositional framework is also effective on larger ST corpora. In particular, we use 400 hours of English-German and 500 hours of English-French ST from the MuST-C corpus (Di Gangi et al., 2019). Our Multi-Decoder model improves by +2.7 and +1.5 BLEU, in German and French respectively, over end-to-end baselines from prior works that do not use additional training data. We show that ASR re-scoring gives an additional +0.1 and +0.4 BLEU improvement. [5]

By extending our Multi-Decoder models to this MuST-C study, we show the generalizability of our approach across several dimensions of ST tasks. First, our approach consistently improves over baselines across multiple language-pairs. Second, our approach is robust to the distinct domains of telephone conversations from Fisher-CallHome and the TED-Talks from MuST-C. Finally, by scaling from 170 hours of Fisher-CallHome data to 500 hours of MuST-C data, we show that the benefits of decomposing sequence tasks with searchable hidden intermediates persist even with more data.

Furthermore, the performance of our Multi-Decoder models trained with only English-German or English-French ST data from MuST-C is comparable to other methods which

[5] Details of the MuST-C data preparation and model parameters are detailed in Appendix (C.2).

| Model | Overall ST(↑) | Sub-Net ASR(↓) |
|---|---|---|
| In-domain ST Model | | |
| Baseline (Wang et al., 2020b) | 12.0 | - |
| +ASR Pretrain (Wang et al., 2020b) ♢ | 23.0 | 16.0 |
| Out-of-domain ST Model | | |
| Multi-Decoder | 11.8 | 46.8 |
| +ASR Rescoring w/ in-domain LM | 14.4 | **36.7** |
| Multi-Decoder w/ Speech-Attention | 12.6 | 46.5 |
| +ASR Rescoring w/ in-domain LM | **15.0** | **36.7** |

Table 6.5: Results presenting the overall ST performance (BLEU) and the sub-net ASR (% WER) of our Multi-Decoder models when tested on out-of-domain data. All models were trained on the Fisher-CallHome Es→En corpus and tested on CoVost2 Es→En corpus. ♢Pretrained with 364 hours of in-domain ASR data.

| Model | Intermediate | Fisher ASR(↓) | CallHome ASR(↓) |
|---|---|---|---|
| Enc-Dec ♢ | - | 23.2 | 45.3 |
| Multi-Decoder | Phoneme | 20.7 | 40.0 |
| Multi-Decoder | Character | 20.4 | 39.9 |
| Multi-Decoder | BPE100 | **19.7** | **38.9** |

Table 6.6: Results presenting the % WER ASR performance when using the Multi-Decoder model on decomposed ASR task with phoneme, character, and BPE100 as intermediates. All results are from the Fisher-CallHome Spanish corpus. ♢(Weiss et al., 2017)

incorporate larger external ASR and MT data in various ways. For instance, Zheng et al. (2021) use 4700 hours of ASR data and 2M sentences of MT data for pretraining and multi-task learning. Similarly, Bahar et al. (2021) use 2300 hours of ASR data and 27M sentences of MT data for pretraining. Our competitive performance without the use of any additional data highlights the data-efficient nature of our proposed end-to-end framework as opposed to the baseline encoder-decoder model, as pointed out by Sperber and Paulik (2020).

## 6.8 Discussion and Relation to Prior Work

**Compositionality:** A number of recent works have constructed composable neural network modules for tasks such as visual question answering (Andreas et al., 2016), neural MT (Raunak et al., 2019), and synthetic sequence-to-sequence tasks (Lake, 2019). Modules that are first trained separately can subsequently be tightly integrated into a single end-to-end trainable model

| Model | En→De ST(↑) | En→Fr ST(↑) |
|---|---|---|
| NeurST (Zhao et al., 2020) | 22.9 | 33.3 |
| Fairseq S2T (Wang et al., 2020a) | 22.7 | 32.9 |
| ESPnet-ST (Inaguma et al., 2020b) | 22.9 | 32.7 |
| Dual-Decoder (Le et al., 2020) | 23.6 | 33.5 |
| Multi-Decoder w/ Speech-Attn. | 26.3 | 37.0 |
| +ASR Re-scoring | **26.4** | **37.4** |

Table 6.7: Results presenting the overall ST performance (BLEU) of our Multi-Decoder w/ Speech-Attention models with ASR re-scoring across two language-pairs, English-German (En→De) and English-French (En→Fr). All results are from the MuST-C tst-COMMON sets. All models use speech transcripts.

by passing differentiable soft decisions instead of discrete decisions in the intermediate stage (Bahar et al., 2021). Further, even a single encoder-decoder model can be decomposed into modular components where the encoder and decoder modules have explicit functions (Dalmia et al., 2019b).

**Joint Training with Sub-Tasks:** End-to-end sequence models been shown to benefit from introducing joint training with sub-tasks as auxiliary loss functions for a variety of tasks like ASR (Kim et al., 2017), ST (Salesky et al., 2019; Liu et al., 2020b; Dong et al., 2020; Le et al., 2020), SLU (Haghani et al., 2018). They have been shown to induce structure (Belinkov et al., 2020) and improve the model performance (Toshniwal et al., 2017), but this joint training may reduce data efficiency if some sub-nets are not included in the final end-to-end model (Sperber et al., 2019; Wang et al., 2020c). Our framework avoids this sub-net waste at the cost of computational load during inference.

**Speech Translation Decoders:** Prior works have used ASR/MT decoding to improve the overall ST decoding through synchronous decoding (Liu et al., 2020b), dual decoding (Le et al., 2020), and successive decoding (Dong et al., 2020). These works partially or fully decode ASR transcripts and use discrete intermediates to assist MT decoding. Tu et al. (2017) and Anastasopoulos and Chiang (2018) are closest to our multi-decoder ST model, however the benefits of our proposed framework are not entirely explored in these works.

**Two-Pass Decoding:** Two-pass decoding involves first predicting with one decoder and then re-evaluating with another decoder (Geng et al., 10; Sainath et al., 2019; Hu et al., 2020; Rijhwani et al., 2020). The two decoders iterate on the same sequence, so there is no decomposition into sub-tasks in this method. On the other hand, our approach provides the subsequent decoder with

a more structured representation than the input by decomposing the complexity of the overall task. Like two-pass decoding, our approach provides a sense of the future to the second decoder which allows it to correct mistakes from the previous first decoder.

**Auto-Regressive Decoding:** As auto-regressive decoders inherently learn a language model along with the task at hand, they tend to be domain specific (Samarakoon et al., 2018; Müller et al., 2020). This can cause generalizability issues during inference (Murray and Chiang, 2018; Yang et al., 10), impacting the performance of both the task at hand and any downstream tasks. Our approach alleviates these problems through intermediate search, external models for intermediate re-scoring, and multi-sequence attention.

## 6.9 Conclusion

In this chapter, we present searchable hidden intermediates for end-to-end models of decomposable sequence tasks. We show the efficacy of our Multi-Decoder model on the Fisher-CallHome Es→En and MuST-C En→De and En→Fr speech translation corpora, achieving state-of-the-art results. We present various benefits in our framework, including sub-net performance monitoring, beam search for better hidden intermediates, external models for better search, and error propagation avoidance. Further, we demonstrate the flexibility of our framework towards out-of-domain tasks with the ability to adapt our sequence model at intermediate stages of decomposition. Finally, we show generalizability by training Multi-Decoder models for the speech recognition task at various levels of decomposition.

# Chapter 7

# Compositional E2E SLU Models

In Chapter 6 we presented the searchable intermediates framework that exploits the task compositionality for complex sequence tasks to build E2E systems using their simpler sub-task formulations. In this chapter we apply the searchable intermediates framework to spoken language understanding. We show how this framework enables end-to-end sequence labeling spoken utterances using the token level tagging formulation.

## 7.1 Token-level Sequence Labeling for Spoken Language Understanding using Compositional End-to-End Models

End-to-end spoken language understanding (SLU) systems are gaining popularity over cascaded approaches due to their simplicity and ability to avoid error propagation. However, these systems model sequence labeling as a sequence prediction task causing a divergence from its well-established token-level tagging formulation. We build compositional end-to-end SLU systems that explicitly separate the added complexity of recognizing spoken mentions in SLU from the NLU task of sequence labeling. By relying on intermediate decoders trained for ASR, our end-to-end systems transform the input modality from speech to token-level representations that can be used in the traditional sequence labeling framework. This composition of ASR and NLU formulations in our end-to-end SLU system offers direct compatibility with pre-trained ASR and NLU systems, allows performance monitoring of individual components and enables the use of globally normalized losses like CRF, making them attractive in practical scenarios. Our models outperform both cascaded and direct end-to-end models on a labeling task of named entity recognition across SLU benchmarks.

## 7.2 Introduction

Sequence labeling (SL) is a class of natural language understanding (NLU) tasks. These systems *tag* each word in a sentence to provide insights into the sentence structure and meaning (Jurafsky and Martin, 2000). An SL system that processes unstructured text, first encodes the context and relationships of words in the sentence using an encoder and then labels each token (Lample et al., 2016; Dozat et al., 2017; Akbik et al., 2018). However, when dealing with spoken utterances, sequence labeling introduces an additional complexity of also *recognizing* the mentions of the labels (Kubala et al., 1998; Zhai et al., 2004).

SL in spoken language understanding (SLU) has been approached by two schools of thought, (1) that seek to recognize the spoken words using an ASR engine and then tag the mentions using an NLU engine in a cascaded manner (Palmer and Ostendorf, 2001; Horlock and King, 2003; Béchet et al., 2004), and (2) that seek to recognize and tag the mentions directly from speech in an end-to-end (E2E) framework (Arora et al., 2019; Ghannay et al., 2018). Prior work has shown that cascaded systems suffer due to error propagation (Tran et al., 2018) from the ASR into the NLU engine, which can be overcome in an E2E framework. However, unlike cascaded models, E2E systems cannot utilize the vast abundance of NLU research (Shon et al., 2022) as they re-define the SL problem as a complex sequence prediction problem where the sequence contains both the tags and its mentions.

Inspired by the principles of task compositionality in SL for SLU, we seek to bring both schools of thought together. Our conjecture is that we can build compositional E2E systems that first convert the spoken utterance to a sequence of token representations (Dalmia et al., 2021), which can then be used to train token-wise classification systems as per the NLU formulation. By also conditioning our token-wise classification on speech, our compositional E2E system allows recovery from errors made while creating token representations. We instantiate our formulation on a popular SL task of named entity recognition (NER) and (1) present the efficacy of our compositional E2E NER-SLU system on benchmark SLU datasets (Bastianelli et al., 2020; Shon et al., 2022) surpassing both the cascaded and direct E2E systems §7.6.1. (2) Our compositional model consists of ASR and NLU components compatible with pre-trained ASR and NER-NLU models §7.6.2. (3) Our E2E systems exhibit transparency towards categorizing errors by enabling the evaluation of individual components of our model in isolation §7.6.3.

The remaining sections first describe the traditional SL formulation (§7.3), and discuss shortcomings in current SLU formulations (§7.4). Section §7.5 presents our compositional E2E model that can overcome these shortcomings. We then evaluate these approaches towards the SL task of NER (§7.6).

## 7.3 Sequence Labeling (SL)

SL systems tag each word, $w_i$, of a text sequence, $S = \{w_i \in \mathcal{V} | i = 1, \ldots, N\}$ of length $N$ and vocabulary $\mathcal{V}$, with a label from a label set $\mathcal{L}$, $\{w_i \rightarrow y_i \, | y_i \in \mathcal{L}\}$. This produces a label sequence, $Y = \{y_i \in \mathcal{L} | i = 1, \ldots, N\}$ of the same length $N$. Using decision theory, sequence labeling models seek to output $\hat{Y}$ from a set of all possible tag sequence $\mathcal{L}^N$,

$$\hat{Y} = \underset{Y \in \mathcal{L}^N}{\text{argmax}} \, P(Y|S) \tag{7.1}$$

where $P(Y|S)$ is the posterior distribution. This posterior can be modeled using various techniques like the traditional HMM (Morwal et al., 2012) and MEMM (McCallum et al., 2000) based modeling and more recently CRF (Ma and Hovy, 2016) and token classification (Devlin et al., 2019) based approaches. We discuss the latter two in detail:

**Conditional Random Field:** Lafferty et al. (2001) aims to directly compute the posterior of the entire label sequence $Y$ given the sentence $S$:

$$P(Y|S) = \frac{e^{F(Y,S)}}{\sum_{Y' \in \mathcal{L}^N} e^{F(Y',S)}} \tag{7.2}$$

where $F(Y, S)$ is global score of the tag sequence $Y$ given $S$. This is modeled using a linear chain CRF which computes the global score as a sum of local scores $f(.)$ for each position in $Y$ as follows

$$F(Y, S) = \sum_{l=1}^{N} f(y_{l-1}, y_l, S) \tag{7.3}$$

Lample et al. (2016) and Yan et al. (2019) use contextualized neural encoders like LSTMs and transformers to model context of the entire sequence $S$ for every word $w_l$. This allows for effective modeling of $f(.)$ by using encoder representations for each word as the emissions, and maintaining a separate transition score $t_{y_{l-1} \rightarrow y_l}$ to give $F(Y, S)$:

$$\mathbf{h}_{1:N} = \text{encoder}(w_{1:N}) \tag{7.4}$$

$$t_{y_{l-1} \rightarrow y_l} = \text{transitionScores}(|\mathcal{L}|, |\mathcal{L}|) \tag{7.5}$$

$$F(Y, S) = \sum_{l=1}^{N} (\mathbf{h}_{l,y_l} + t_{y_{l-1} \rightarrow y_l}) \tag{7.6}$$

**Token Classification Model:** Since the advent of strong contextual modeling using transformer based models, sequence labeling can also be treated as token classification (Devlin et al., 2019), a simplification over MEMM estimations (McCallum et al., 2000), with the assumption that the

current tag is conditionally independent to previous tag.

$$P(Y|S) = \prod_{l=1}^{N} P(y_l|\mathbf{h}_l) \tag{7.7}$$

These models are still effective as $\mathbf{h}_l$ is able to model the full context $S$ for every word $w_l$.

In cases like NER, where an entity can span multiple words, these problems are modeled using BIO tags (Ramshaw and Marcus, 1999), where begin (B), inside (I) tags are added for entities and an outside (O) tag for non-entity words, extending the tag set vocabulary from $\mathcal{L}$ to $\mathcal{L}' = \{l_B \oplus l_I | l \in \mathcal{L}\} \cup \{\text{O}\}$.When modeled using sub-word tokens the tags can be aligned to the first sub-word token of the word and the remaining ones can be marked with a special token $\varnothing$ giving $\mathcal{L}'' = \mathcal{L}' \cup \{\varnothing\}$.

## 7.4 Sequence Labeling in SLU

Sequence Labeling in SLU introduces an added complexity of recognizing mentions on top of text-based SL tasks (§7.3). Given a sequence of $d$ dimensional speech feature of length $T$ frames, $X = \{\mathbf{x}_t \in \mathbb{R}^d | t = 1, \ldots, T\}$, these systems seek to estimate the label sequence $\hat{Y}$ from

$$\hat{Y} = \underset{Y \in \mathcal{L}^*}{\text{argmax}}\ P(Y|X) \tag{7.8}$$

where $P(Y|X)$ have been modeled as:

**Cascaded SLU** (Béchet et al., 2004; Parada et al., 2011; Zhou et al., 2015) models $P(Y|X)$ from $P(Y|S)$ using an NLU framework (§7.3) and $P(S|X)$ using an ASR model (Povey et al., 2011; Chan et al., 2016; Graves, 2012), assuming conditional independence of $Y|S$ from $X$,

$$P(Y|X) = \sum_{S} P(Y|S,\cancel{X})P(S|X) \tag{7.9}$$

$$\approx \max_{S} P(Y|S)P(S|X) \tag{7.10}$$

$$\approx P(Y|\hat{S}) \max_{S} P(S|X) \tag{7.11}$$

$$\hat{S} = \underset{S \in \mathcal{V}*}{\text{argmax}}\ P(S|X) \tag{7.12}$$

Once $\hat{S}$ is estimated, $\hat{Y}$ can be estimated using Eq 7.1. Although this enables realizing $\hat{Y}$ using two well studied frameworks, the independence assumption doesn't allow recovery from errors in estimating $\hat{S}$.

**Direct End-to-End SLU** (Arora et al., 2019; Shon et al., 2022) systems avoid cascading errors by directly modeling $P(Y|X)$ in a single monolithic model. To achieve this while being able to recognize the spoken mentions, these systems enrich $Y$ with transcripts $S$, $Y^e = \{y_i^e \in \mathcal{V} \cup \mathcal{L} | i = 1, \ldots, N'\}$, where $N'$ is the length of $Y^e$. This can be modeled using an autoregressive decoder as:

$$P(Y|X) = \prod_{i=1}^{N'} P(y_i^e | y_{1:i-1}^e, X) \tag{7.13}$$

However this new formulation cannot utilize the well studied sequence labeling framework §7.3. Additionally, this applies an extra burden of labeling along with alignment on the decoder and makes understanding the errors made by these systems particularly difficult. For example, Eq 7.13 gives non-zero likelihood to a corrupt sequence with only labels and no words as $y^e \in \{\mathcal{V} \cup \mathcal{L}\}$.

## 7.5 Compositional End-to-End SLU

We propose to bring the two paradigms together in a compositional end-to-end system, by extending over the cascaded SLU formulation using searchable intermediate framework (Dalmia et al., 2021):

$$P(Y|X) = \sum_S P(Y|S, X) P(S|X) \tag{7.14}$$

$$\approx \max_S \underbrace{P(Y|S, X)}_{\text{SUB}_{\text{NLU}}\text{NET}} \underbrace{P(S|X)}_{\text{SUB}_{\text{ASR}}\text{NET}} \tag{7.15}$$

This system can be realized with two sub-networks

**$\text{SUB}_{\text{ASR}}$NET:** Modeling $P(S|X)$,

$$\mathbf{h}^E = \text{encoder}_{\text{ASR}}(X_{1:T}) \tag{7.16}$$

$$\mathbf{h}_l^{\text{ASR}} = \text{decoder}_{\text{ASR}}(\mathbf{h}^E, w_{1:l-1}) \tag{7.17}$$

$$P(w_l|X, w_{1:l-1}) = \text{softmaxOut}(\mathbf{h}_l^{\text{ASR}}) \tag{7.18}$$

$$P(S|X) = \prod_{l=1}^{N} P(w_l|X, w_{1:l-1}) \tag{7.19}$$

**SUB$_{\text{NLU}}$NET:** Modeling $P(Y|S,X)$,

$$\mathbf{h}^{\text{NLU}}_{1:N} = \text{encoder}_{\text{NLU}}(\mathbf{h}^{\text{ASR}}_{1:N}, \mathbf{h}^{E}_{1:T}) \tag{7.20}$$

$$P(Y|S,X) = \text{CRF}(\mathbf{h}^{\text{NLU}}_{1:N}) \qquad \textbf{OR} \tag{7.21}$$

$$P(Y|S,X) = \text{TokenClassification}(\mathbf{h}^{\text{NLU}}_{1:N}) \tag{7.22}$$

Where the end-to-end differentiability is maintained by using $\mathbf{h}^{\text{ASR}}_{1:N}$ in Eq 7.20. During inference, we approximate the viterbi max of $S$ using beam search to give $\hat{\mathbf{h}}^{\text{ASR}}_{1:N}$. Then $\hat{Y}$ can be found using viterbi search with no approximation as the output length is known and the solution is tractable.

This composition allows incorporating the ASR modeling and text-based sequence labeling framework §7.3. It also brings transparency to end-to-end modeling as we can also monitor performance of individual sub-nets in isolation. Further, encoder$_{\text{NLU}}$ can attend to speech representations $\mathbf{h}^{\text{E}}_{1:T}$ using cross attention (Dalmia et al., 2021) enabling the direct use of speech cues for NLU.

## 7.6 Spoken Named Entity Recognition

To show the effectiveness of our compositional E2E SLU model we build spoken NER systems on two publically available SLU datasets, SLUE (Shon et al., 2022) and SLURP (Bastianelli et al., 2020) (dataset and preparation details in §D.1). We compare our compositional E2E system with cascaded and direct E2E systems. We also compare with another compositional E2E system that we develop using the labeling formulation of direct E2E systems (§7.4). We build all our systems using ESPnet-SLU (Arora et al., 2019). All models were tuned separately using validation sets with the same hyperparameter search space. Full descriptions of model and training parameters are in §D.2.

SLURP is evaluated using SLU-F1 (Bastianelli et al., 2020) which weighs the entity labels with the word and character error rate of the predicted mentions and SLUE using F1 (Shon et al., 2022) which evaluates getting both the mention and the entity label exactly right. We also compute Label-F1 for both datasets which considers only the entity label. We report micro-averaged F1 for all results.

### 7.6.1 Performance of Compositional E2E SLU

Table 7.1 shows that our proposed compositional E2E models with the token-level NLU formulation outperform both cascaded and direct E2E models on all benchmarks using both CRF and Token Classification. In order to understand gains of our proposed model, we examine the performance of our compositional system with direct E2E formulation (§7.4). While being comparable to direct E2E models, they still lag behind our proposed models showing the efficacy of modeling

| Model | SLURP | | SLUE | |
|---|---|---|---|---|
| | SLU F1 | Label F1 | F1 | Label F1 |
| Direct E2E SLU (Arora et al.) | 71.9 | - | 54.7 | 67.6 |
| Casacaded SLU (Ours) | 73.3 | 80.9 | 48.6 | 63.9 |
| Direct E2E SLU (Ours) | 77.1 | 84.0 | 54.7 | 67.6 |
| Compositional E2E SLU | | | | |
| w/ Direct E2E formulation (§7.4) | 77.2 | 84.6 | 50.0 | 68.0 |
| w/ Proposed NLU formulation (§7.5) | | | | |
| CRF w/ Speech Attention (SA) | 77.7 | 85.2 | 59.4 | 73.6 |
| Token Classification w/ SA | **78.0** | **85.3** | **60.3** | **73.7** |
| w/o Speech Attention | 77.7 | 84.9 | 59.0 | 73.6 |

Table 7.1: Results presenting the micro F1 performance of our proposed compositional E2E models using CRF and Token Classification modeling. Cascaded, direct E2E and our compositional E2E with direct E2E formulation are shown for comparison. We also provide an ablation of our model with and without Speech Attention (SA).

SL tasks as a token-level tagging (§7.3) in an E2E SLU framework.

We further analyze our compositional systems that don't attend to speech representations. We observe a performance drop as these models are not able to recover from errors made while "recognising" entity mentions. For example, in an utterance that says "change the bedroom lights to green", though the ASR component incorrectly predicts the transcript as "change the color of lights to green", the NLU component w/ Speech Attention is able to recover the entity type HOUSE_PLACE.

### 7.6.2 Utilizing External Sub-Net models

Components of our compositional E2E SLU model have functions similar to an ASR and NLU model (Eq 7.16-7.22). This allows fine-tuning our models using sub-systems, pre-trained on large amounts of available sub-task data. Table 7.2 shows that our compositional model has better compatibility with ASR and NLU fine-tuning over direct E2E systems, thereby increasing their performance gap, particularly for SLUE, an under-resourced SLU dataset.

Further our models have the ability to use transcripts from a strong external model ($S^{\text{ext}}$) directly during inference, by instantiating our models with these transcripts to produce $\mathbf{h}^{\text{ASR}}$ and then evaluate $P(Y|S^{\text{ext}}, X)$. Table 7.2 shows using transcripts from an external ASR *with no fine-tuning steps* can achieve similar performance to ASR fine-tuning.

| Model | SLURP | | SLUE | |
|---|---|---|---|---|
| | SLU F1 | Label F1 | F1 | Label F1 |
| Direct E2E SLU | 77.1 | 84.0 | 54.7 | 67.6 |
| w/ NLU fine-tuning | Incompatible | | | |
| w/ ASR fine-tuning | 73.5 | 81.2 | 64.0 | 80.6 |
| Compositional E2E SLU (w/ SA) | 78.0 | 85.3 | 60.3 | 73.7 |
| w/ NLU finetuning (w/o SA) | 77.7 | 84.9 | 62.4 | 76.4 |
| w/ ASR finetuning (w/ SA) | **81.4** | **88.8** | **71.6** | **85.2** |
| Compositional E2E SLU (w/ SA) | 78.0 | 85.3 | 60.3 | 73.7 |
| w/ External ASR Transcripts ($S^{\text{ext}}$) | **81.0** | **88.1** | **70.1** | **81.2** |

Table 7.2: Results presenting the compatibility of our models with pre-trained ASR and NLU systems by (1) finetuning pre-trained components and (2) directly utilizing transcripts from an external ASR model.

| | SLURP | | SLUE | |
|---|---|---|---|---|
| | ASR (%WER ↓) | NLU (SLU-F1 ↑) | ASR (%WER ↓) | NLU (F1 ↑) |
| Pure ASR & NLU models | 16.1 | 82.4 | 30.4 | 58.1 |
| Compostional E2E SLU | | | | |
| CRF w/ Speech Attention (SA) | 16.3 | 88.3 | 27.4 | 75.6 |
| Token Classification w/o SA | 16.0 | 87.9 | 27.6 | 74.1 |
| Token Classification w/ SA | 16.1 | 88.7 | 27.5 | 75.6 |

Table 7.3: Results showcasing the transparency of our compositional E2E models by evaluating the individual sub-networks ASR (%WER) and NLU (F1) in isolation.

### 7.6.3 Transparency in Compositional E2E SLU

Following Eq 7.15, we can estimate ASR performance by calculating $\hat{S}$ using beam search and NLU performance by estimating $\hat{Y}$ from $P(Y|S^{\text{GT}}, X)$, where $S^{\text{GT}}$ is the ground truth transcripts. Table 7.3 shows the performances of individual components of our model along with performances of ASR and NLU only models suggesting that we can effectively monitor the performance of these components, helping practitioners analyze and debug them. For instance, while our models with and without speech attention have comparable performance on ASR, using speech attention improves NLU power. Further the one-to-one alignment of transcripts and sequence labels can provide further categorization of errors, as shown in §7.6.4.

| | Entity Correct | | Entity Incorrect | |
|---|---|---|---|---|
| | Model | # Examples | Model | # Examples |
| ASR Correct | w/ SA | 8520 | w/ SA | 465 |
| | w/o SA | 8501 | w/o SA | 474 |
| ASR Incorrect | w/ SA | 1568 | w/ SA | 1343 |
| | w/o SA | 1585 | w/o SA | 1336 |

Table 7.4: Number of examples per error category of our compositional E2E SLU with/without Speech Attention on SLURP test set. There are four categories depending on whether mistakes are made by ASR or NLU component. Note that the first quadrant lists # of *correct* examples, while the rest list *incorrect* ones. Direct E2E systems cannot offer such categorizations particularly for incorrect entities as there is no alignment between ASR and NLU outputs.

### 7.6.4 Error Categorization

The predictions made by our compositional E2E SLU model can be categorized into different buckets on the basis of the errors by ASR or NER component. Table 7.4 demonstrates this behavior by categorizing the errors of our compositional E2E model trained with and without speech attention. Most of the performance differences between compositional E2E SLU model w/ and w/o speech attention are caused by the kinds of errors where the ASR predictions are inaccurate, but the NLU module is nevertheless able to recover the correct entity type from the utterance. This confirms our intuition that cross attention on speech representations can help the NLU module to recover from mistakes made during "recognizing" spoken mentions. We also present anecdotes for each of these error categories in Figure 7.1. This further emphasizes the transparency in our compositional E2E SLU models. Due to the lack of one-to-one alignment between ASR and Sequence Labeling, such analysis is not possible in direct E2E SLU systems, making it particularly difficult to categorize errors when the entity prediction is wrong.

### 7.6.5 CRF vs Token Classification

For practical SLU the likelihoods of our compositional model $P(Y|S, X)$, should be correlated with errors in label sequence $Y$. We found that in SLURP our compositional E2E SLU, while using locally normalized token classification shows no correlation (Corr=0.13,p=0), using CRF exhibits moderate correlation (Corr=0.43, p=0). This makes globally normalized models attractive for real-world scenarios like automated data auditing and human in-the-loop ML (Mitchell et al., 2018) despite their marginal addition in computation cost.

| | Hypothesis | Reference |
|---|---|---|
| ASR Correct<br>Entity Correct | EVENT DATE<br>event reminder mona tuesday | EVENT DATE<br>event reminder mona tuesday |
| ASR Correct<br>Entity Incorrect | MOVIE TYPE NEWS TOPIC<br>is there anything happening on jazz scene around edinburgh | MOVIE TYPE PLACE NAME<br>is there anything happening on jazz scene around edinburgh |
| ASR Incorrect<br>Entity Correct | EVENT NAME PERSON DATE TIME<br>create meeting with paul for tomorrow at ten am | EVENT NAME PERSON DATE TIME<br>put meeting with pawel for tomorrow ten am |
| ASR Incorrect<br>Entity Incorrect | EVENT NAME DATE<br>set a birthday event for ninety | EVENT NAME PERSON<br>set a birthday event for martin |

Figure 7.1: Qualitative examples of our compositional E2E SLU model for various error categories. We can observe that in the first case, the model is correctly able to predict both entity types and mentions even when the name "mona" is not a common name for an event. In the second case, even though it predicts the correct ASR transcript, it mislabels "Edinburgh" as a news topic since the phrase "is there anything happening" usually occurs with news topics. In the third case, even though it makes a mistake in the person name, the model correctly tags it as a person. Finally, the model incorrectly generates the word "ninety," and this error gets propagated to the NLU component through token representations which then predicts entity type "date". This analysis shows that the alignment between ASR and NLU outputs can help us gain better insights into model performance.

## 7.7 Conclusion

In this chapter, we propose to combine text based sequence labeling framework into the speech recognition framework to build a compositional end-to-end model for SLU. These models not only show superior performance over cascaded and direct end-to-end SLU systems, but also bring the power of both these systems in a single framework. These models exhibit transparency of cascaded systems, while avoiding error propagation like direct end-to-end systems.

With our compositional end-to-end SLU model, we strive to bring the research from the text based sequence labeling directly into speech based spoken language understanding. Our aim is to avoid re-invention of the wheel, but rather come up with innovative ways to build end-to-end models by converting a complex problem into simpler ones that have seen substantial research in the past. Additionally we believe the increased capacity for error analysis in our compositional end-to-end system can help towards building better practical systems during deployment. Our compositional end-to-end systems can effectively utilize pre-trained ASR and NLU systems, thereby avoiding the need for collecting large labeled datasets for SLU. This framework also saves compute by utilizing pre-trained ASR systems directly during inference to improve downstream performances with no fine-tuning.

# Part III

## Evaluation of E2E models

# Chapter 8

# Practical Considerations for Evaluating End-to-End Systems

End-to-end systems are gaining popularity, and are being used to solve many complex sequence tasks (Arora et al., 2019; Devlin et al., 2019). These systems aims to perform many sub-tasks functions in a single monolithic design to solve the overall task. Our conjecture is that in order to use end-to-end models to build practical systems, it is critical to evaluate their generalizability towards variations in each sub-task that comprises the complex sequence task rather than testing generalizability to the input. In this chapter, we discuss this problem in detail and suggest techniques for building more comprehensive test sets.

## 8.1 Rethinking End-to-End Evaluation of Decomposable Tasks: A Case Study on Spoken Language Understanding

Decomposable tasks are complex and comprise of an hierarchy of sub-tasks. Spoken intent prediction, for example, combines automatic speech recognition and natural language understanding. Existing benchmarks, however, typically hold out examples for only the surface-level sub-task. As a result, models with similar performance on these benchmarks may have unobserved performance differences on the other sub-tasks. To allow insightful comparisons between competitive end-to-end architectures, we propose a framework to construct robust test sets using coordinate ascent over sub-task specific utility functions. Given a dataset for a decomposable task, our method optimally creates a test set for each sub-task to individually assess sub-components of the end-to-end model. Using spoken language understanding as a case study, we generate new splits for the Fluent Speech Commands and Snips SmartLights datasets. Each split has two test sets: one with held-out utterances assessing natural language understanding abilities, and one with held-out speakers to test speech processing skills. Our splits identify performance gaps up

to 10% between end-to-end systems that were within 1% of each other on the original test sets. These performance gaps allow more realistic and actionable comparisons between different architectures, driving future model development. We release our splits and tools for the community.[1]

## 8.2 Introduction

Complex, real-world tasks, such as the spoken language understanding (SLU) tasks of spoken intent prediction and spoken language translation, comprise of hierarchies of simpler sub-tasks. Spoken intent prediction combines automatic speech recognition (ASR) to process audio, followed by natural language understanding (NLU) to classify an utterance to a particular intent (intent prediction) (Gorin et al., 1997). Similarly, speech translation involves an ASR task followed by machine translation (MT) to translate a transcription of the input audio (Weiss et al., 2017).

Deep, end-to-end models (Qian et al., 2021b; Lugosch et al., 2019; Huang et al., 2020; Kuo et al., 2020; Chen et al., 2021c; Dinarelli et al., 2020) are adopted for these complicated tasks due to advancements in model architectures and computing capabilities. End-to-end architectures typically outperform traditional, modular architectures without requiring domain expertise or feature engineering (Agrawal et al., 2020). Moreover, end-to-end models avoid the error propagation arising from traditional approaches (Sperber and Paulik, 2020). However, traditional modular or cascade architectures, naturally structured into sub-components that each address a specific subtask, are more straightforward to evaluate. End-to-end models cannot quantify performance of decomposed sub-tasks (Dalmia et al., 2021), blurring the lines between the individual sub-tasks. Using pre-trained systems for some sub-tasks further reduces the chance of errors propagating to the downstream sub-tasks (Bansal et al., 2019; Erhan et al., 2010; Wang and Zheng, 2015). Thus, it is important to explicitly evaluate each sub-component of an end-to-end network.

Prior datasets for decomposable problems, however, often test only the top-level subtask. For example, the Fluent Speech Commands (FSC) (Lugosch et al., 2019) and Air Travel Information System (ATIS) (Hemphill et al., 1990) SLU benchmarks are open-speaker but not open-utterance, and thus, only effectively test speaker generalizability. Moreover, train-test overlap is a problem in modern question answering datasets (Lewis et al., 2020). In paradigms like encoder-decoder modeling or speech translation (Dalmia et al., 2021; Kano et al., 2020; Jia et al., 2019; Weller et al., 2021), neural network components each solve different logical functions which combine to solve the final task. Therefore, standard benchmarks may effectively test a particular sub-network of a given system, masking any weaknesses of other model sub-components and providing an inflated estimates of model performance.

To address this, we present a dataset-agnostic framework for evaluating end-to-end model

[1] https://github.com/robustEval/robustEvalSLU

on decomposable tasks. Using spoken intent prediction as a case study, we focus on two popular benchmarks, FSC (Lugosch et al., 2019) and Snips SmartLights (Snips) (Saade et al., 2019; Coucke et al., 2018) datasets. We provide evidence that the original test splits do not fairly evaluate the ASR and NLU subtasks of spoken intent prediction. Using our framework, we propose robust *Unseen* and *Challenge* splits that each contain two test sets: one test set with held-out speakers, and one with held-out utterances. For the Challenge set, we use coordinate ascent with speaker- and transcript-specific utility metrics to explicitly test for generalization to diverse speakers and varied phrasings of intents. Our experiments show the new test splits can amplify accuracy differences by up to 10% between sub-components of several state-of-the art models for spoken intent prediction, offering more in-depth analyses of strengths and weaknesses of various end-to-end modeling approaches. These splits have the potential to drive future modeling innovation, not only for this task, but for any similarly decomposable task.

## 8.3 Motivation

In the following section, we introduce the spoken intent prediction task and discuss limitations of existing SLU benchmarks.

### 8.3.1 Task Definition

Spoken intent prediction maps a spoken command (e.g. "Turn on the lights in the kitchen") and to a discrete, actionable set of slots (Action: "Activate"), (Object: "Lights"), (Location: "Kitchen"). This task challenges a model's speech recognition and semantic processing abilities: a good SLU model must generalize to new speakers *and* to new phrasings of similar intents.

The FSC dataset (Lugosch et al., 2019) tests a model's ability to predict intents from commands used with a home voice assistant. Following traditional speech processing paradigms, the FSC test set consists of audio from speakers unseen during training. Although this test split measures generalization to new speakers, the test set fails to explicitly test generalization to new utterances. As Table 8.2 illustrates, the training set provides 100% coverage of the transcripts seen during test time. This issue with transcript coverage is also seen in the popular ATIS benchmark (Hemphill et al., 1990) for slot filling from air travel commands. Snips, which is similar in content to FSC, does not release official splits, but typical split creation approaches (Coucke et al., 2018; Qin et al., 2019; Agrawal et al., 2020) do not explicitly consider testing generalizability for each sub-task independently. This can mask performance gaps in the ASR or NLU sub-tasks.

In the real world, we expect systems to understand the same commands spoken in different ways by speakers of diverse backgrounds. Thus, it is important to hold out new utterances to more robustly assess a model's semantic processing ability (McKenna et al., 2020). Moreover, open-speaker test sets should assess model generalizability to diverse demographics. In FSC, for

Table 8.1: % WER values for Google ASR model (Google, 2021) on speakers with first language English and non English. (S, I, and D refers to substitution, insertion and deletion errors, respectively)

| First Language Spoken | % S | % I | % D | % WER |
|---|---|---|---|---|
| English | 2.0 | 0.7 | 2.4 | 10.8 |
| Non English | 6.3 | 2.2 | 7.7 | 27.8 |

example, all held-out speakers are native English speakers, while accented speakers are seen only during training time. To understand how this affects spoken language understanding evaluation, we used Google's ASR system (Google, 2021) to generate transcripts from audio files in the dataset and computed Word Error Rates using the gold transcripts. Table 8.1 illustrates that the ASR model's WER on audio from speakers whose first language is not English is twice as high as the WER on audio from native English speakers. To develop technologies that are inclusive to different speaker demographics, it is important to create benchmarks that are representative of these diverse backgrounds.

Table 8.2: Comparing data statistics and different models compared on original and proposed splits for the Fluent Speech Commands dataset. Speaker and Utterance Coverage refer to the percentages of test set speakers and utterances, respectively, observed in the training set. "Speaker KL" is the symmetrised Kullback-Leibler divergence of speaker demographic distributions between training and test sets. We ensure that our proposed splits have roughly the same # of examples in each test set as in the split proposed in (Lugosch et al., 2019). We also construct different variants of the Unseen split by changing the random seed of our algorithm and report the standard deviation.

| | Dataset Statistics | | | | E2E SLU Model (Lugosch et al., 2019) Test Accuracy | | | |
|---|---|---|---|---|---|---|---|---|
| Fluent Speech Command Test Set | Speaker Coverage | Utterance Coverage | Speaker KL | Test Size | No Pretraining | w/ Pretrained ASR (Frozen) | Finetune Word + Intent Layers | Finetune All Layers |
| Original Split | 0% | 100% | 0.88 | 3793 | 96.8 | 98.5 | 99.1 | 97.2 |
| Random Split | 100% | 100% | <0.01 | 3793 | 94.6 | 96.2 | 97.2 | 95.8 |
| Unseen Split (Spk.) | 0% | 100% | 0.01 | 3366 | 92.0 (±0.4) | 92.9 (±0.2) | 94.2 (±0.3) | 93.9(±0.4) |
| Unseen Split (Utt.) | 100% | 0% | <0.01 | 3971 | 78.1 (±1.3) | 86.0 (±0.7) | 88.2 (±0.9) | 88.3(±2.0) |
| Challenge Split (Spk.) | 0% | 100% | 0.01 | 3349 | 87.2 | 90.9 | 92.3 | 91.1 |
| Challenge Split (Utt.) | 100% | 0% | <0.01 | 4204 | 68.2 | 73.4 | 78.3 | 74.1 |

## 8.4 Methodology

We discuss our approach for creating the open-utterance and open-speaker test sets for the *Unseen* and Challenge splits. The *Challenge* split uses additional constraints to make both test sets more difficult and realistic.

### 8.4.1 Dataset Optimization

We construct test splits using coordinate ascent (Wright, 2015; Metzler and Bruce Croft, 2007) over sub-task driven utility functions. Each coordinate direction corresponds to the test set assignment (either the open-speaker or open-utterance test sets) of a block of datapoints in the dataset. We first select an open-speaker test set, then choose the open-utterance test set. Finally, we randomly distribute the remaining instances into training and validation sets, preserving the original size ratio and intent distributions of these sets (Lugosch et al., 2019).

### 8.4.2 Unseen Split

We use the two functions to generate unseen-speaker and unseen-utterance splits with desirable qualities.

**Unseen Speaker Set** The FSC dataset contains speakers of various ages, native languages, English fluency levels, and genders, but the original, open-speaker test set is not representative of these groups. To ensure we are testing on speakers of diverse backgrounds, we minimize the symmetrised Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) between the discrete distributions of speaker demographics in the training and test sets.

**Unseen Utterance Set** When selecting unique utterances to hold out from training, we minimize the symmetrised KL divergence of the discrete intent label distributions between training and test sets. Utterances with the same intent are semantically similar, ensuring the semantic distributions of training and test sets match. We also minimize the KL divergence of the discrete distributions of transcript lengths between training and test sets.

### 8.4.3 Challenge Split

In addition to the constraints defined in the previous section, we define speaker-specific and transcript-specific utility functions to quantify the "hardness" for each subtask. When optimized, these functions create more challenging, realistic held-out sets. Notably, these test sets may capture dataset outliers due to noisy recordings, labeling errors, or poorly aligned data. Thus, we recommend using them in addition to the Unseen splits. The proposed utility functions are specific to spoken language tasks, but could be replaced with arbitrary task-specific objectives.

**Challenge Speaker Set** We compute the Word Error Rate, measured by insertion, deletion, and substitution errors, of Google's ASR model (Google, 2021) to identify particularly challenging utterances. However, high WER is not always indicative of a reasonably hard example. According to previous work (Gaur et al., 2016), substitution errors reflect confusions in ASR systems. Alignment errors, indicated by an increase in insertion and deletion errors and a large deviation between these quantities, are a sign of poor data quality. To produce a challenging speaker set

without compromising data quality, we use the following utility function, $U_{\text{WER}}$:

$$U_{\text{WER}} = S - \alpha\,|I - D| - \beta\,I - \gamma\,D$$

where $S$, $I$ and $D$ refer to substitution, insertion, and deletion rates, respectively, such that we maximize $S$ while minimizing $I$, $D$ and their deviation, $|I - D|$. $\alpha$, $\beta$ and $\gamma$ are hyperparameters. We empirically observe $\alpha = 0.05$, $\beta = 0.05$ and $\gamma = 0.4$ work well for the FSC dataset, producing a challenging split without compromising on test-set data quality.

**Challenge Utterance Set** A dataset with many unique n-grams makes the SLU task more difficult (McKenna et al., 2020). Thus, we create splits that minimize the n-gram overlap between our train and test set. We choose the Sentence BLEU (Papineni et al., 2002) score as a proxy for n-gram overlap and use it in the following utility function:

$$U_{\text{BLEU}} = -BP * \exp(\sum_{i=1}^{4} \alpha_i \log(p_i))$$

where $p_i$ is the modified precision (Papineni et al., 2002) for each n-gram, $\alpha_i$ weighs the respective importance of the $i^{\text{th}}$-gram overlap, and *BP* is the brevity penalty (Papineni et al., 2002) penalizing shorter sentences. Transcripts in the FSC dataset are 3-5 words in length, thus, considering only 1-gram and 2-gram overlap (i.e. $\alpha_1 = 0.5, \alpha_2 = 0.5, \alpha_3 = 0.0, \alpha_4 = 0.0$) worked well for holding out unique n-grams not seen during training.

Finally, to ensure both *Unseen* and *Challenge* speaker sets test generalization only to new speakers (and not utterances), we maximize the n-gram overlap with each split's respective training utterances. As Table 8.2 shows, our constructed speaker test sets have 100% n-gram overlap with their respective training sets. Unlike the original splits, in which test speakers are less demographically diverse than training set speakers, our new splits effectively minimize this distributional gap, as shown by the "Speaker KL" column. Each new test set has similar size to the original test set, and its distribution of utterance lengths is kept close to that of the training set to limit distribution shift.

## 8.5 Experiments

### 8.5.1 Comparing end-to-end SLU systems

We compare four different models from (Lugosch et al., 2019) on the *Original*, *Unseen*, and *Challenge* splits, as well as a stratified *Random* split (stratified over all intent labels). The four models are based on a three-stage neural architecture consisting of a phoneme layer, word layer, and intent layer. Each model uses different pretraining and finetuning schemes: using no pretraining, using a frozen pretrained ASR model (i.e. finetuning only the intent layers), finetuning

only word and intent layers, or finetuning all layers. When pretraining, the phoneme and word modules are pretrained on the LibriSpeech dataset (Panayotov et al., 2015). Using the Original test split, we successfully reproduced the results (Lugosch et al., 2019) for each of these freezing and unfreezing schedules.

Using the speaker and utterance test sets we create, we can highlight sub-task-level performance differences across the four models. As Table 8.2 illustrates, our Unseen and Challenge splits reveal that all models are better at generalizing to new speakers than to new utterances. However, all models achieve at least 3% lower accuracy on the Unseen and Challenge speaker sets compared with the Original held-out speaker set, indicating that current SLU models still do not generalize well to diverse speaker demographics. The results on the Challenge utterance set indicate that all models are significantly worse at generalizing to unique phrases of the same intent, suggesting an opportunity for enhancing semantic processing abilities of SLU models.

Our splits are also useful for comparing configurations of the same model. Intuitively, pretraining phoneme layers to detect phonetic patterns should help generalize to unseen speakers and utterances. However, Table 8.2 shows that on the Original and Random splits, there are small performance gaps between pretrained and non-pretrained models (1-2%, or $\sim$50 test set examples), suggesting pretraining offers limited value, considering the resources it requires. In contrast, the performance gap becomes significant in the Unseen utterance set and both the speaker and utterance Challenge splits. The model without pretraining performs 10.1% worse than the best pre-trained model (pretraining with finetuned word and intent layers) in both the Unseen and Challenge utterance sets, corresponding to $\approx$460 more mistakes. The gaps are smaller in the speaker Challenge set, suggesting a non-pretrained ASR model generalizes better to new speakers than to new words or phonemes. These results corroborate previous findings (McKenna et al., 2020) that finetuning models to the dataset's distinct acoustic and linguistic patterns improves generalization to new phrasings. Finally, we change the random seeds used to create the Unseen split to test the robustness of our methods. The relatively low standard deviations in performance, as seen in Table 8.2, illustrate that our method is stable.

### 8.5.2 Gap between SLU and NLU

Using the utterance test sets of the Unseen and Challenge splits, we identified that SLU systems struggle to effectively capture lexical and semantic information. As an ablation study, we used gold transcripts to train and test the intent prediction component of the end-to-end model (Lugosch et al., 2019) in isolation (keeping all word and phoneme layers frozen). As a baseline, we train a text-based intent classification model initialized with random word embeddings that are finetuned during training. To incorporate semantic information into the word representations, we extract two types of word embeddings: (1) pretrained FastText (Mikolov et al., 2018) embeddings and (2) contextual BERT embeddings (Devlin et al., 2019). In Figure 8.1, we compare the baseline and

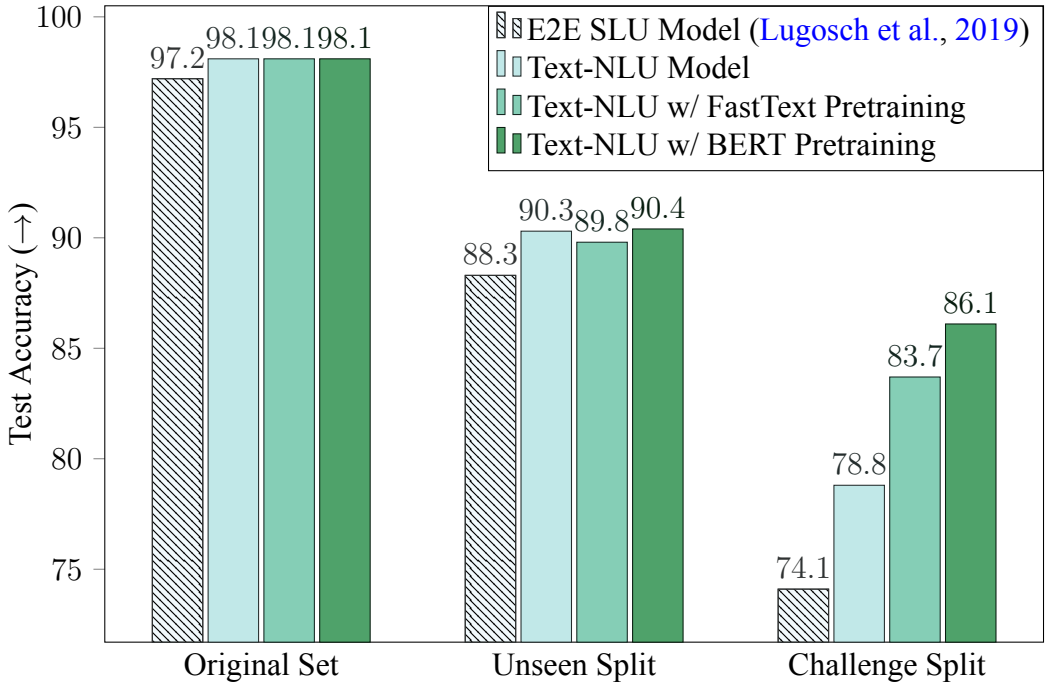



Figure 8.1: Comparing text-based NLU models with the "Finetune All Layers SLU" baseline on the original and proposed utterance test sets. "Text-NLU Model" refers to a text-based NLU system using randomly initialized word embeddings.

semantically-enhanced text NLU models with the "Finetune All Layers" SLU model of (Lugosch et al., 2019). Figure 8.1 illustrates that BERT pretraining can boost the accuracy of the intent subcomponent by 12% on the Challenge utterance set. These differences are not so apparent in the Unseen split, which is not as semantically challenging because it does not explicitly minimize n-gram overlap (McKenna et al., 2020). There is still a 2% gap between the SLU and NLU models' performance on the Unseen utterance split, suggesting that pretraining embeddings helps enhance semantic understanding.

Based on the results of our ablation study, we extend the frozen pretrained ASR model, the best reported SLU model from (Lugosch et al., 2019), with FastText embeddings. At each audio frame, the ASR module predicts a distribution over words; we use this compute a weighted average FastText word embedding (Bojanowski et al., 2017) and pass it to the intent layer. Table 8.3 illustrates that enriching the ASR outputs with semantic information gives very minor improvements on the original test set, but provides >2% improvement to the unseen-utterance sets of both Unseen and Challenge splits, consistent with the experiments in the previous section.

Table 8.3: Adding semantic word embeddings to the SLU system has only a minor effect (<1%) on the the Original split and proposed unseen-speaker splits. On the unseen-utterance splits, we see a magnified performance gap (>2%), in **bold**.

| E2E SLU Model (Lugosch et al., 2019) | Original | Unseen | | Challenge | |
|---|---|---|---|---|---|
| | | Spk. | Utt. | Spk. | Utt. |
| Pretrained ASR (Frozen) | 98.5 | 92.9 | 86.0 | 90.9 | 73.4 |
| + FastText Pretraining | 98.7 | 92.7 | **88.3** | 90.0 | **75.5** |

Table 8.4: % WER values of the Google ASR system (Google, 2021) on the original and proposed speaker test sets, and the corresponding accuracy of the Pretrained ASR (Frozen) model. (S, I, and D refers to substitution, insertion, and deletion, respectively.)

| Test Split | % S | % I | % D | % WER | SLU Acc. |
|---|---|---|---|---|---|
| Original | 1.0 | 0.5 | 0.6 | 6.5 | 98.5 |
| Unseen Speaker | 2.1 | 1.0 | 2.5 | 12.1 | 92.9 |
| Challenge Speaker | 3.2 | 1.2 | 2.6 | 13.9 | 90.9 |

### 8.5.3 Analyzing proposed utility functions

In Section 8.5.1, we illustrated how our optimized splits can distinguish model performance. We now verify our utility functions can effectively quantify the complexity of each subtask.

**Word Error Rate** As in Section 8.3.1, we use WER of Google's ASR system (Google, 2021) to quantify the difficulty of our splits. Table 8.4 illustrates that WER is twice as high on the proposed speaker-diverse splits as on the original splits. Substitution errors are most prominent in Challenge set, indicating that we create a hard test set without necessarily compromising on data quality.

**N-gram overlap** For each proposed split, we compute the average BLEU score for the test set relative to the training set. Table 8.5 highlights that our test splits have much lower n-gram overlap with their training sets. Minimizing n-gram overlap while preserving intent distributions of training and test sets further tests a model's generalization to new phrasings of the same intents.

### 8.5.4 Extending to the Snips SmartLights dataset

To illustrate our methodology is dataset agnostic, we extend our approach to Snips SmartLights, a popular SLU dataset (Lugosch et al., 2020; Bhosale et al., 2019).

Snips SmartLights dataset is unseen-utterance by design because all utterances are unique. Thus, we create a single Unseen test set that holds out speakers and utterances. We optimize both speaker and utterance utilities defined in Section 8.4.2 to create the split. Using the WER and n-gram based utilities defined in Section 8.4.3, we create separate speaker and utterance

Table 8.5: BLEU score values for unigram, bigram, trigram and 4-gram for Original and proposed (utterance) test sets. Utterances shorter than order of a given n-gram were removed.

| | N-gram overlap | | | | SLU |
|---|---|---|---|---|---|
| Test Split | 1 | 2 | 3 | 4 | Acc. |
| Original | 100.0 | 100.0 | 100.0 | 100.0 | 98.5 |
| Unseen Utterance | 98.0 | 87.4 | 73.4 | 71.7 | 86.0 |
| Challenge Utterance | 91.0 | 69.9 | 66.4 | 66.6 | 73.4 |

Table 8.6: Evaluating models on original and proposed splits for the Snips SmartLights dataset. Snips does not provide default splits, so we compare against a random split.

| | Rand. (Agrawal et al., 2020) | Unseen | Challenge Split | |
|---|---|---|---|---|
| E2E SLU Model (Lugosch et al., 2019) | Split | Split | (Spk.) | (Utt.) |
| No Pretraining | 60.4 | 27.3 | 37.8 | 45.2 |
| w/ Pretrained ASR (Frozen) | 83.2 | 78.5 | 73.2 | 67.4 |
| Finetune Word + Intent Layers | 88.0 | 80.9 | 82.6 | 75.3 |
| Finetune All Layers | 85.0 | 75.0 | 74.8 | 78.5 |

Challenge test sets. Following the Challenge test setup of FSC, we increase the speaker test set's n-gram overlap set with its train set to match that of the random split. We do not control for n-gram overlap in the Unseen split since it holds out both speakers and utterances. As a result, our Challenge speaker set may be easier than the Unseen split for SLU models. Moreover, Snips is a smaller dataset, so the Snips Challenge set's train, valid, speaker test, utterance test ratios are 75:10:7.5:7.5 as compared to 80:10:10 in the baseline random split (Agrawal et al., 2020).

Using the same models as Section 8.5.1, we compare the performance on our proposed splits against a random split (Agrawal et al., 2020) in Table 8.6 (Snips does not release official splits). We observe similar results as in the FSC setting. Pretraining ASR models improves speaker test set performance by nearly 40-50% for the Unseen and Challenge splits. Moreover, finetuning improves performance for all splits, especially on the Challenge utterance set, on which both finetuned models achieve nearly 12% gains in performance. Thus, we illustrate that we can easily extend our approach to another SLU benchmark, and see effects consistent with those on the FSC dataset. From our experiments, we believe this approach has potential to apply to other benchmarks for decomposable tasks across diverse research domains.

## 8.6 Conclusions

We present a novel, dataset-agnostic methodology for constructing splits for decomposable tasks, casting the construction of splits as an optimization problem over dataset-level utility functions.

We release *Unseen* and *Challenge* splits for the FSC and Snips datasets to the community, and show evidence that these splits can amplify performance differences between sub-components of models. We recommend the use of the *Unseen* splits for testing in-domain performance and the *Challenge* splits for more extreme out-of-domain generalization scenarios. As our methodology is task-agnostic, we encourage the extension of our re-splitting method to other decomposable tasks, such as speech translation or visual question answering.

# Chapter 9

# Conclusion

The software industry has made great strides in building independent reusable libraries that can be developed once and reused in a wide range of applications. This thesis takes one step towards bringing sequence to sequence neural models closer to computer programs with reusable components. This line of research can provide a way for the research community and industry to build complex neural systems while preventing an explosion in computational training costs.

In this thesis, we first define the principles of compositionality towards and identify its benefits towards building complex systems. We take inspiration from the works of traditional cascade systems and attempt to bring their benefits into the future of end-to-end modeling. We propose three models with the desired characteristics, which have various levels of reusability and modularity, which practitioners could use depending on their need and available resources. We also study the considerations for evaluating end-to-end sequence models which would increase their practicality towards their deployment on real-world scenarios.

## 9.1 Summary of Key Contributions

This thesis makes the following key contributions:

1. Chapter 3 shows the resource pooling capabilities of CTC Hybrid Systems by building ASR systems for low resourced languages. These systems exploit compositionality in the ASR task by pooling resources across languages to build strong CTC acoustic models, which are decoded with an WFST decoder.
2. Chapter 4 builds modular encoder-decoder systems by defining a clear interface between encoder and decoder modules using CTC frame level distributions. These systems, while being end-to-end, exhibit modular behavior like traditional HMM based systems. Modules from a trained LegoNN model can be reused for other sequence tasks without any fine-tuning, while still having the capability end-to-end fine-tuning for practioners with available resources.

3. Chapter 5 builds CTC encoders for non-monotonic sequence tasks like speech and machine translation, thereby opening the CTC Hybrid Systems (Chapter 3) for sequence tasks beyond speech recognition. This work also presents two strategies for a one-pass beam search that considers both the CTC and Attention distributions, thus strengthening the capabilities of LegoNN Systems (Chapter 4).
4. Chapter 6 builds compositional E2E systems for complex sequence tasks using searchable hidden intermediates. These systems rely on intermediate decoders to build cascade-like end-to-end systems. These systems do not suffer from error propagation while still exhibiting search capabilities.
5. Chapter 7 applies the searchable intermediate framework for the spoken language understanding task. Our compositional E2E SLU systems can utilize the traditional token-level sequence labeling formulation for natural language understanding in end-to-end spoken language understanding. This work also builds globally normalized CRF based model for end-to-end SLU.
6. Chapter 8 presents an approach for a more comprehensive evaluation of end-to-end sequence models. We show that it is important to build test sets that test generalizability on all sub-tasks that compromise the overall task. We present an approach to construct these test sets by optimizing through difficulties in sub-tasks using co-ordinate ascent.

## 9.2 Future Directions

This thesis is centered on building sequence models that use principles of task compositionality in their design. Going forward we believe these models will play an important role towards the practical use of these systems. In this section, we provide possible extensions to our work, which we believe can enhance the experience of practitioners while building complex sequence systems for real-world scenarios.

### 9.2.1 Composition Inspired Transformer Architectures

With the advent of attention based modeling (Vaswani et al., 2017) there has been growing interest in coming up with innovative ways to model relationships in sequences; for example, Conformer (Gulati et al., 2020) uses convolutions to model local relationships along with self-attention for global relations. We recently proposed Branchformer (Peng et al., 2022) that uses separate branches to model different levels of relationships in sequences; this allows for flexible inference where slower branches can be shut off to give faster inference at some cost of performance without any retraining. This level of flexibility can play a crucial role in the deployment of these models, especially when building real-time systems. One possible extension could be build-

ing architectures that can also function with a subset of the trained model parameters, providing free compression for edge devices.

### 9.2.2 Flexible Tokenization for Easy Composition of Systems

As we enter the world of multimodal machine learning, systems built to handle one modality need to interact with systems of another modality; for example, a speech recognition system interacts with a text-based natural language understanding system in voice assistants to understand commands and take appropriate action. However, different systems have different optimal tokenizations, for example, a speech recognition system cares about recognizing sounds, which makes their modeling stronger at a smaller unit than a word (Watanabe et al., 2018), while understanding systems require large word-like units to model the semantics (Devlin et al., 2019). Due to such discrepancies, their composition forgoes crucial information regarding the entropy of these predictions by converting them into surface level outputs before feeding them into the next system (Ravichander et al., 2021). With finite-state transducers (FSTs), we can go from distributions in one tokenization level to another. Furthermore, these transducers can be made end-to-end differentiable using differentiable weighted FSTs (Hannun et al., 2020), giving rise to another class of compositional end-to-end systems. This direction of work can also strengthen our LegoNN systems (Dalmia et al., 2022) by placing these between modules to allow flexible interfacing.

### 9.2.3 Zero-shot LegoNN systems

One limitation of the LegoNN systems (Dalmia et al., 2022) is that decoder modules cannot be trained in isolation, which means that they cannot utilize the vast text-only data to train their decoder module. In order to be able to do this, we suggest investigating a one-hot-to-distribution model. This model would mimic the CTC distributions of a LegoNN encoder by first corrupting the input sequence and then trying to predict the same input sequence using the CTC loss, such as BART (Lewis et al., 2019). These systems will allow for building encoder-decoder systems with little to no parallel data; for example, a LegoNN encoder system that models phoneme distributions from speech can be used with a LegoNN decoder built on text only data to build speech recognition systems for an unseen language. Additionally, these systems can also be used to build end-to-end cascaded systems in which independently trained ASR LegoNN and MT LegoNN models (trained with the one-hot-to-distribution component) can be combined later to build zero-shot speech translation systems.

### 9.2.4 Trained Sequence Module API using LegoNNs

Just like Pytorch (Paszke et al., 2019) provides APIs for trainable modules, and inspired by the success of HuggingFace (Wolf et al., 2019) and AdapterHub (Pfeiffer et al., 2020), we believe that with LegoNNs, we can provide APIs for trained modules that practitioners can utilize to enhance their system building. For example, providing trained LegoNN decoder modules can save computation as practitioners would only have to train encoder modules. They would also help researchers build systems on low-resourced datasets as the LegoNN APIs will be usable without any fine-tuning. In addition, the end-to-end capabilities of LegoNN systems will allow further fine-tuning for those with available compute.

### 9.2.5 Joint CTC Encoder and Masked Conditional Decoder Models

The current encoder-decoder models rely on left-to-right decoding to produce the predicted sentence, as the length of the output prediction is unknown and depends on when the decoder predicts <eos>. This limits the decoder's capability to model the sentence structure only using the past context (Peters et al., 2018). As shown in Chapter 5, we can build input synchronous decoding models that do not require <eos> prediction. This paves the way for a new set of encoder-decoder models in which the decoder is not an autoregressive conditional prediction model, but rather a masked conditional decoder similar to BERT (Devlin et al., 2019). These systems can be decoded by creating a CTC lattice composed with masked conditional decoder scores similar to CTC with n-gram LM decoding using WFSTs (Miao et al., 2015), thus combining the best of probabilistic modeling and neural networks. These systems can also be useful for efficient discriminative training (McDermott, 1997), as obtaining the denominator score would not require any auto-regressive decoding.

### 9.2.6 Streaming Compositional E2E Models

One current limitation of the compositional systems in Chapters 6 and 7 is that they require an intermediate beam search step before evaluating the second sub-network. Our follow-up work allowed us to use non-autoregressive CTC predictions to obtain intermediate representations in a single parallel step (Inaguma et al., 2021a). However, the CTC outputs can be sub-optimal when used to produce the best intermediate representations due to their conditional independence assumption. We believe that a wait-k policy (Zheng et al., 2019; Ren et al., 2020) between the intermediate and final predictions can allow simultaneous decoding of the final output while the intermediate decoder performs a beam search. Extending this line of work would play a crucial role in the use of these models for real-time prediction.

# Appendix

# Appendix A

# Supplemental Material for Chapter 4

## A.1 Experimental and Data Setup for transfer of LegoNN experiments

### A.1.1 Setup for the Ro-En transfer experiment

*Data:* For our Ro-En transfer experiment, we used the WMT16 Ro-En and WMT19 De-En datasets. For WMT16 Ro-En, we used the data prepared by (Lee et al., 2018) which is already tokenized and lowercased. For De-En, we excluded ParaCrawl from the standard WMT19 raw training set to obtain 7.4M parallel translation pairs. We applied the same processing as for the Ro-En dataset by lowercasing and tokenizing the De-En set. We prepared a single joint dictionary of 41000 BPE units by combining the training text from both datasets. We filtered the training data to have a length ratio of 1.5 and 80 as the max token length (Lee et al., 2018). Following (Lee et al., 2018; Ghazvininejad et al., 2020a), we report tokenized BLEU scores on this dataset.

*Training:* As the Ro-En dataset contains only 610K pairs, to achieve the best baseline performance, we reduce the size of the baseline model to avoid overfitting following (Ghazvininejad et al., 2020b). We use 6 encoder and decoder layers with 512 dimensions, 8 heads and 2048 feed-forward units. We modify the Ro-En encoder-only LegoNN model accordingly by using transformer blocks with 512 dimensions, 16 heads and 4096 feed-forward units. For De-En LegoNN models along with the architecture described in §4.5.2 we also experimented with larger models where the transformer encoder block has 16 heads, and 4096 feed-forward units. We found that the larger BeamConv model performs better for this cross-lingual modular experiments.

*Fine-tuning:* To finetune the composed LegoNN Ro-En WMT model we modify the learning rate to lr$= 5e^{-6}$ and warm-up steps to $15k$. We run the training for around $4k$ steps to get the best validation perplexity.

### A.1.2 Setup for the ASR-MT transfer experiment

*Data:* For Table 4.4 demonstrating the third LegoNN scenario in Figure 4.3, we use the Europarl speech data to train our ASR task. We followed the data preparation for the ASR models described in (Iranzo-Sánchez et al., 2020) by lowercasing, tokenizing and stripping the punctuations from the text. We followed the same recipe when training models for the WMT19 De-En dataset, using the raw text from Appendix A.1.1. To prepare the BPE target units, we trained a BPE model with vocab size 2000 on the Europarl train text and applied the prepared dictionary on the other sets of the Europarl speech data and the De-En WMT data. We had an OOV rate of 0.096% in the De-En WMT train data.

For Table 4.5, demonstrating the fourth LegoNN scenario in Figure 4.3, we used the TED-LIUM dataset (Rousseau et al., 2014) for training the pronunciation model module. We processed the transcripts in the same way by lowercasing, tokenizing, and stripping the punctuation from the text data. We trained the 2000 BPE target units on Europarl and TED-LIUM train text and applied the prepared dictionary on the other sets of Europarl, TED and De-En WMT data. We had an OOV rate of 0.090% in the De-En WMT train data.

We use the grapheme-to-phoneme library described in (Park and Kim, 2019) to get the target phonemes for the speech data used in both the experiments. We report final WER on the lower-cased set to follow the standard ASR data setups.

*Training:* As the Europarl dataset contains only 70 hours, we reduced the size of the baseline model to avoid overfitting. We use transformers (Vaswani et al., 2017) with 12 encoder blocks and 6 decoder blocks, each with 512 dimensions, 8 heads, 2048 feed-forward units.

For the LegoNN phoneme+pronunciation encoder-only model trained using the CTC loss, we use a length control unit (for phonemes) which reduces the length of the input by a factor of 1.2 with a maximum allowable length of 365 time-steps. It outputs a marginal distribution over the phonemes which is then passed to an Ingestor component (Wemb RF=5) followed by 6 layers of encoder and another output length control unit (for BPE output tokens) reducing the speech input by a factor 3.5 with a maximum allowable length of 130 time-steps. For the De-En MT model we use the same architecture as Appendix A.1.1 but with non-shared dictionary between source and target.

*Fine-tuning:* To finetune the composed LegoNN Europarl ASR model we modify the peak learning rate to lr$= 1e^{-4}$ and warm-up to a $1k$ steps. We run the training for $20k$ steps to get the best validation perplexity.

[1] We downloaded the public model of (Ott et al., 2018) to score the newstest2011-2016 test sets which weren't reported in their original paper.

[2] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/detokenizer.perl

Table A.1: SacreBLEU Scores (↑) on WMT for LegoNN and baseline enc-dec MT models.

| MT Task | Loss Criterion CTC | Loss Criterion CE | WMT En→De dev(↑) | WMT En→De test(↑) |
|---|---|---|---|---|
| Scaling NMT (Ott et al., 2018) | ✗ | ✓ | 26.8 | 28.2 [1] |
| Baseline Models (Our Implementation) | | | | |
| Baseline Enc-Dec | ✗ | ✓ | 27.2 | 28.3 |
| **LegoNN Models** | | | | |
| Encoder Only | ✓ | ✗ | 19.0 | 18.2 |
| Encoder + BeamConv Decoder | ✓ | ✓ | 26.3 | 26.7 |
| Encoder + WEmb Decoder | ✓ | ✓ | 26.8 | 27.3 |

Table A.2: Individual year tokenized (tok.) and detokenized (detok.) BLEU Scores (↑) on WMT for LegoNN and baseline enc-dec MT models.

| MT Task | WMT En→De (↑) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | newstest11 | | newstest12 | | newstest13 | | newstest14 | | newstest15 | | newstest16 | |
| | tok. | detok. | tok. | detok. | tok. | detok. | tok. | detok. | tok. | detok. | tok. | detok. |
| Scaling NMT (Ott et al., 2018) [1] | 22.7 | 22.3 | 23.1 | 23.0 | 27.3 | 26.8 | 29.8 | 29.2 | 32.2 | 31.8 | 35.0 | 34.8 |
| Baseline Models (Our Implementation) | | | | | | | | | | | | |
| Baseline Enc-Dec | 23.3 | 22.8 | 23.3 | 23.1 | 27.6 | 27.2 | 29.8 | 29.0 | 31.9 | 31.5 | 35.6 | 35.1 |
| **LegoNN Models** | | | | | | | | | | | | |
| Encoder Only | 16.0 | 15.8 | 15.7 | 15.7 | 19.1 | 19.0 | 18.0 | 17.8 | 20.3 | 20.2 | 21.6 | 21.7 |
| Encoder + BeamConv Decoder | 22.4 | 21.9 | 22.4 | 22.3 | 26.7 | 26.3 | 27.3 | 26.6 | 29.8 | 29.5 | 33.4 | 33.0 |
| Encoder + WEmb Decoder | 22.7 | 22.3 | 22.8 | 22.6 | 27.2 | 26.8 | 28.3 | 27.6 | 30.3 | 30.0 | 34.3 | 33.9 |

## A.2 Detokenized BLEU performance of our LegoNN MT models

Table A.1 shows the detokenized BLEU performance of the MT models used in Table 4.1[1]. We used the detokenizer from mosesdecoder [2] to detokenize the model output and SacreBLEU (Post, 2018) [3] to calculate the BLEU score with the following hash -

`BLEU+case.mixed+lang.en-de+numrefs.1+smooth.exp+test.{wmt11|wmt12|wmt13|wmt14/full|wmt15|wmt16}+tok.13a+version.1.2.9`

[3] https://github.com/mjpost/sacrebleu

## A.3 BLEU performance on individual years for our LegoNN MT models

Table A.2 shows the BLEU performance (tokenized and detokenized BLEU) from individual years (newstest11-16) for the MT models used in Table 4.1.

## A.4 Interpretable Interface in LegoNN models

Each LegoNN module has a task, clear performance metric, and defined interpretable interface allowing one to asses the quality of individual modules. This gives a sense of the contribution from each module towards the overall performance of the task and help in better debugging of the end-to-end systems. In Table A.3, we show the loss and performance of the encoder modules of the LegoNN models that were presented in the table 4.1 and 4.2. The performance of the encoder module was calculated using greedy decoding of the encoder marginal distributions without using an external language model.

Table A.3: BLEU Scores (↑) and % WER (↓) at the output of encoder modules of the trained LegoNN systems presented in Table 4.1 and 4.2.

| ASR Task | CTC Loss | Enc. WER (↓) SWB | CH |
|---|---|---|---|
| WEmb LegoNN | 0.60 | 11.6% | 24.2% |
| BeamConv LegoNN | 0.65 | 11.4% | 24.4% |
| **MT Task** | **CTC Loss** | **Enc. BLEU (↑) dev** | **test** |
| WEmb LegoNN | 2.51 | 18.9 | 18.4 |
| BeamConv LegoNN | 2.56 | 18.2 | 17.8 |

## A.5 Importance of encoder grounding using CTC for reusability

Table A.4 shows the results of training WEmb LegoNN MT models without an OLC unit nor a CTC loss at the encoder output. The initial system works fine but its components fails completely when used with another independently trained module.

Table A.4: Importance of CTC encoder grounding for modularity

| LegoNN Models | En → De (↑) newstest14 |
|---|---|
| WEmb Ingestor | 29.2 |
| w/ Enc. Swap | 28.7 |
| No CTC Loss at encoder output | 29.2 |
| w/ Enc. Swap | 0.0 |

Table A.5: Effect of % WER (↓) of transferring a fully trained ASR LegoNN decoder module on encoder-only module trained on different amounts of data.

| | 10% Data | | 30% Data | |
|---|---|---|---|---|
| ASR Task | SWB(↓) | CH(↓) | SWB(↓) | CH(↓) |
| Baseline Enc-Dec | 129.7% | 136.6% | 20.4% | 37.1% |
| LegoNN Encoder Module | 70.5% | 80.5% | 25.4% | 43.8% |
| + SWBD 300H LEGONN MODULES | | | | |
| BeamConv Decoder | **40.7%** | **59.6%** | **17.5%** | **34.5%** |
| WEmb Decoder | 42.2% | 63.4% | 18.5% | 34.7% |

## A.6 Transfer of LegoNN modules to low-resource conditions

Table A.5 shows the improvements observed when using LegoNN decoder modules with encoders trained with 10% and 30% of the switchboard ASR training data. Applying a decoder trained on the full data provided more than 50% and 10% relative improvement on average compared to the baseline encoder-decoder models, and 30% and 25% compared to a LegoNN encoder module, when trained on the 10% and 30% low resource conditions respectively. Given that this experiment shows module transfer between models trained on the same dataset, encoders are trained without the output length control unit because the their output length distribution matches that of the decoder input.

## A.7 Relation to Incorporating Text-only Resources

There are other ways to incorporate text-only resources in both ASR and MT tasks, e.g., back-translation (Sennrich et al., 2016) and decoding with a language model (Miao et al., 2015; Amodei et al., 2016). However, the LegoNN approach offers a novel way for sharing pre-trained components across tasks and languages (Table 4.3 and Table 4.4), with benefits going beyond competing with the text-only LM decoding. For example, we can further fine-tune the composed LegoNN model (Table 4.6) or through the modular design of LegoNNs, components from an ASR and

MT system would allow building zero-shot speech translation systems, which we have as future work. Additionally, integrating an LM during decoding would also benefit the LegoNN models and encoder-decoder in general when added via shallow-fusion during the beam-search (Watanabe et al., 2018).

For completeness we found that when we apply our LegoNN ASR encoder with a with an external language model using a WFST decoder, the performance is inferior to our LegoNN performance in Table 4.2, achieving 9.1% and 19.4% WER in the SWB and CH test sets averaged across 3 seeds. Since, the gap in the LegoNN encoder performance of MT from the full encoder-decoder is quite large we expect the language model decoding will see a larger degradation compared to the LegoNN encoder-decoder model.

### A.7.1 Training hyperparameters for ASR and MT task

Table A.6 and Table A.7 contain the training parameter details for the LegoNN model.

Table A.6: Hyperparameters for training LegoNN model ASR task.

| Hyperparameter | Value |
|---|---|
| Hidden Dropout | 0.15 |
| Attention dropout | 0.15 |
| Activation dropout | 0.15 |
| Ingestor attention dropout | 0.15 |
| Batch size | 300 utt. |
| LR schedule | tristage (Park et al., 2019) |
| Start learning rate | $1e^{-6}$ |
| Max learning rate | $1e^{-3}$ |
| End learning rate | $5e^{-6}$ |
| Number of steps | 80K |
| Warmup steps | 35K |
| Hold steps | 1K |
| Adam eps | $1e^{-9}$ |
| Adam betas | (0.9, 0.999) |
| Weight decay | $1e^{-6}$ |

Table A.7: Hyperparameters for training LegoNN model MT task.

| Hyperparameter | Value |
|---|---|
| Hidden Dropout | 0.3 |
| Attention dropout | 0.3 |
| Activation dropout | 0.3 |
| Ingestor attention dropout | 0.1 |
| Batch size | 4K sent. |
| LR schedule | inv. sqrt. (Ott et al., 2018) |
| Start learning rate | $1e^{-6}$ |
| Max learning rate | $1e^{-3}$ |
| Number of steps | 150K |
| Warmup steps | 35K |
| Adam eps | $1e^{-9}$ |
| Adam betas | (0.9, 0.999) |
| Weight decay | 0.1 |

# Appendix B

# Supplemental Material for Chapter 5

## B.1 Reproducibility

### B.1.1 Dataset Descriptions

See Table B.1 for dataset descriptions. Data preparation was done using ESPnet recipes: IWSLT14[1], MuST-C[2], MTedX[3].

### B.1.2 Model Architectures

See Table B.2 for model architectures.

### B.1.3 Training/Decoding Hyperparameters

See Table B.3 for MT training, Table B.4 for ST training, Table B.5 for MT decoding, and Table B.6 for ST decoding hyperpameters.

### B.1.4 Metrics

Sacrebleu signature for all non-Japanese:

`BLEU+case.mixed+numrefs.1 +smooth.exp+tok.13a+version.1.5.1`

Sacrebleu signature for Japanese:

`BLEU+case.mixed+lang.en-ja+numrefs.1 +smooth.exp+tok.ja-mecab-0.996-IPA +version.1.5.1`

[1] https://github.com/espnet/espnet/tree/master/egs2/iwslt14/mt1
[2] https://github.com/espnet/espnet/tree/master/egs/must_c/st1
[3] https://github.com/espnet/espnet/tree/master/egs/mtedx/st1

| Dataset | Task | Source Lang(s) | Target Lang(s) | Domain | # Train/Valid/Test Utts | # Speech Train Hours |
|---|---|---|---|---|---|---|
| IWSLT17 (Cettolo et al., 2012) | MT | De | En | TED Talk | 160k/7k/7k | - |
| IWSLT17 (Cettolo et al., 2012) | MT | De | Es | TED Talk | 160k/7k/7k | - |
| MuST-C-v2 (Di Gangi et al., 2019) | ASR/ST | En | De | TED Talk | 250k/1k/3k | 450h |
| MuST-C-v2 (Di Gangi et al., 2019) | ASR/ST | En | Ja | TED Talk | 330k/1k/3k | 540h |
| MTedX (Salesky et al., 2021) | MT | Es, Fr, Pt, It, Ru, El | En | TED Talk | 130k/6k/6k | - |
| MTedX (Salesky et al., 2021) | ASR | Es, Fr, Pt, It, Ru, El | En | TED Talk | 400k/6k/6k | 730h |
| MTedX (Salesky et al., 2021) | ST | Es, Fr, Pt, It, Ru, El | En | TED Talk | 130k/6k/6k | 250h |
| EuroParl (Iranzo-Sánchez et al., 2020) | MT | De | En | Parliament Speech | - /-/2k | - |
| EuroParl (Iranzo-Sánchez et al., 2020) | ST | En | De | Parliament Speech | -/-/1k | - |

Table B.1: MT/ST/ASR dataset descriptions. Utterance counts are rounded to the nearest thousand. Language codes: De=German, En=English, Es=Spanish, Ja=Japanese, Fr=French, Pt=Portuguese, It=Italian, Ru=Russian, El=Greek

| Model | Task | # Encoder Layers [$\mathcal{S}$] | # Decoder Layers | SrcCTC Layer | Up/Down-Sample | Pre-Train Init | Src BPE Size | Tgt BPE Size | # Params |
|---|---|---|---|---|---|---|---|---|---|
| Pure-Attn | MT | 12 [6,12,18] | 6 | - | - | - | 10k (joint) | | 54M |
| Joint CTC/Attn | MT | 18 [6,12,18] | 6 | 6 | 3x | - | 10k (joint) | | 95M |
| Pure-Attn | ST | 18 [12, 18] | 6 | - | 1/4x | Enc lyr 1-12 from ASR | 4k | 4k | 74M |
| Joint CTC/Attn | ST | 18 [12, 18] | 6 | 12 | 1/4x | Enc lyr 1-12 from ASR | 4k | 4k | 72M |
| Pure-Attn | ASR | 12 | 6 | - | - | - | 4k | 4k | 46M |

Table B.2: MT/ST/ASR model descriptions. The best MT/ST Encoder layers settings were selected over a search space indicated by $\mathcal{S}$. Parameter counts are rounded to the nearest million.

For tokenized BLEU in the IWSLT MT datasets we used `mutibleu.perl` (Moses-SMT, 2018)

### B.1.5 Computing

ST models were trained on 2 x V100 for 2 days. MT models were trained on 1 x A6000 for 1 day.

### B.1.6 Valid Set performances

Table B.7 presents the validation performances for our ST and MT models.

## B.2 Qualitative

### B.2.1 View of Regularized Attention

See Figure B.1 for a qualitative example of monotonic source attention patterns (supplementary to the quantitative monotonicity in Table 5.2).

Table B.3: Training Hyperparameters for MT Models.

| Hyperparameter | Value |
|---|---|
| Hidden Dropout | 0.3 |
| Attention dropout | 0.3 |
| Activation dropout | 0.3 |
| LR schedule | inv. sqrt. (Vaswani et al., 2017) |
| Max learning rate | best of [1e-3, 3e-3] |
| Warmup steps | 10000 |
| Number of steps | 200 epoch |
| Adam eps | 1e-9 |
| Adam betas | (0.9, 0.98) |
| Weight decay | 1e-4 |

Table B.4: Training Hyperparameters for ST Models.

| Hyperparameter | Value |
|---|---|
| Hidden Dropout | 0.1 |
| Attention dropout | 0.1 |
| Activation dropout | 0.1 |
| LR schedule | inv. sqrt. (Vaswani et al., 2017) |
| Max learning rate | 0.002 |
| Warmup steps | 25000 |
| Number of steps | 40 epoch |
| Adam eps | 1e-9 |
| Adam betas | (0.9, 0.98) |
| Weight decay | 0.0001 |

## B.3 Quantitative

### B.3.1 compare_mt.py **Analysis Report**

As shown in Figure B.2 and Figure B.3, both joint synchronous decodings are more robust than pure-attention for long output lengths. Input-synchrony appears most robust in generation of very long outputs for ST.

Table B.5: Decoding Search Space MT Models.

| Decoding Type | Hyperparameter | Value |
|---|---|---|
| Pure Attn | Max Length Ratio | [1, 1.2, 1.4, 1.6, 1.8, 2, 2.5, 3] |
| | Penalty | [0, 0.2, 0.4, 0.6, 0.8, 1.0] |
| | Beam Size | 5 |
| Joint O-Sync | Max Length Ratio | 1 |
| | Penalty | [0, 0.2, 0.4, 0.6, 0.8, 1.0] |
| | CTC Weight | 0.3 |
| | Beam Size | 5 |
| Joint I-Sync | Max Length Ratio | 1 |
| | Penalty | [0, 0.2, 0.4, 0.6, 0.8, 1.0] |
| | Blank Penalty | [0.5, 0.75, 1.0] |
| | CTC Weight | [0.3, 0.5] |
| | Beam Size | [10, 30] |

Table B.6: Decoding Search Space ST Models.

| Decoding Type | Hyperparameter | Value |
|---|---|---|
| Pure Attn | Max Length Ratio | 1 |
| | Penalty | [0,0.2,0.4,0.6,0.8,1.0] |
| | Beam Size | [10, 30, 50] |
| Joint O-Sync | Max Length Ratio | 1 |
| | Penalty | [0,0.2,0.4,0.6,0.8,1.0] |
| | CTC Weight | [0.3, 0.5] |
| | Beam Size | [10, 30, 50] |
| Joint I-Sync | Max Length Ratio | 1 |
| | Penalty | [0,0.2,0.4,0.6,0.8,1.0] |
| | Blank Penalty | 1 |
| | CTC Weight | [0.3, 0.5] |
| | Beam Size | [10, 30, 50] |

| Model Name | Model Type | | | MT | | ST | |
|---|---|---|---|---|---|---|---|
| | Joint Train? | Joint Decode? | Decoding Method | IWSLT14 De-En | IWSLT14 Es-En | MuST-C-v2 En-De | MuST-C-v2 En-Ja |
| Pure-Attn (Ours) | ✗ | ✗ | Attn O-sync | 34.1 | 41.2 | 28.5 | 11.3 |
| Joint CTC/Attn | ✓ | ✓ | Joint I-sync | 34.6 | 42.0 | 29.0 | 12.4 |
| Joint CTC/Attn | ✓ | ✓ | Joint O-sync | 35.0 | 42.3 | 29.2 | 12.4 |

Table B.7: Valid set performances, as measured by BLEU (↑), of our proposed joint CTC/Attention models compared to pure-attention baselines. Joint CTC/Attention models are always jointly trained, but can be either jointly decoded using input/output synchronony or decoded using only their CTC or attention branches.

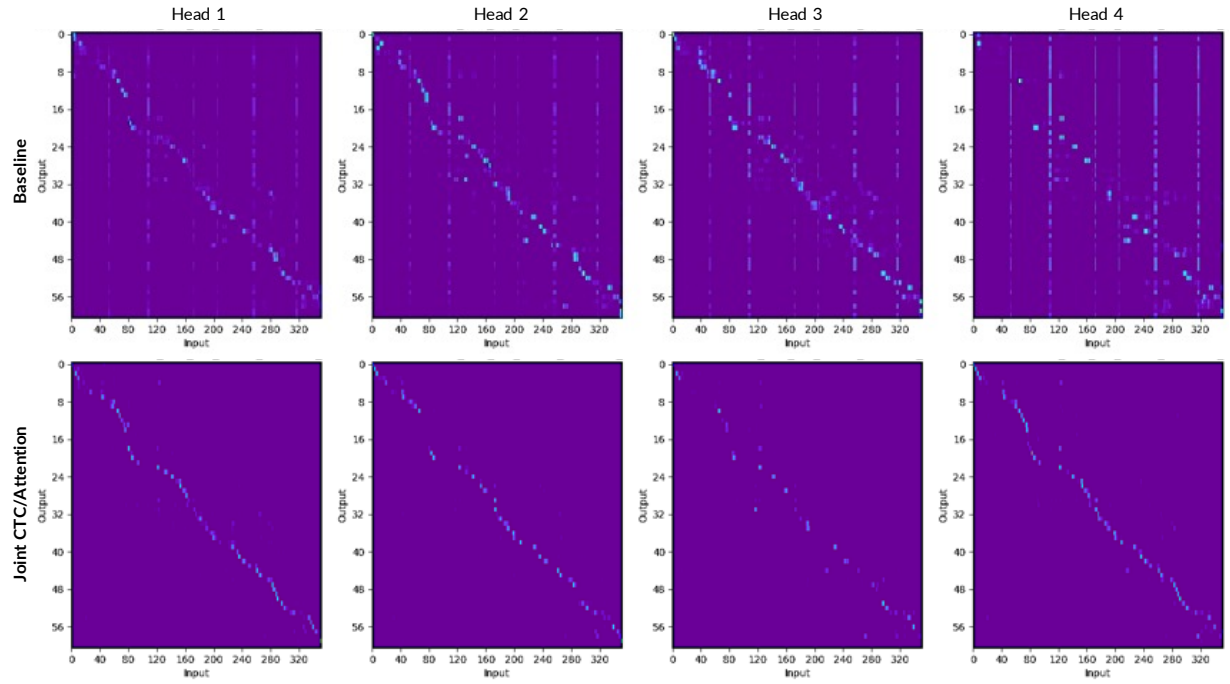



Figure B.1: Visualization of source attention patterns produced by pure-attention baseline (top) vs. joint CTC/attention (bottom) ST models. Qualitative example extracted from the final decoder layer.

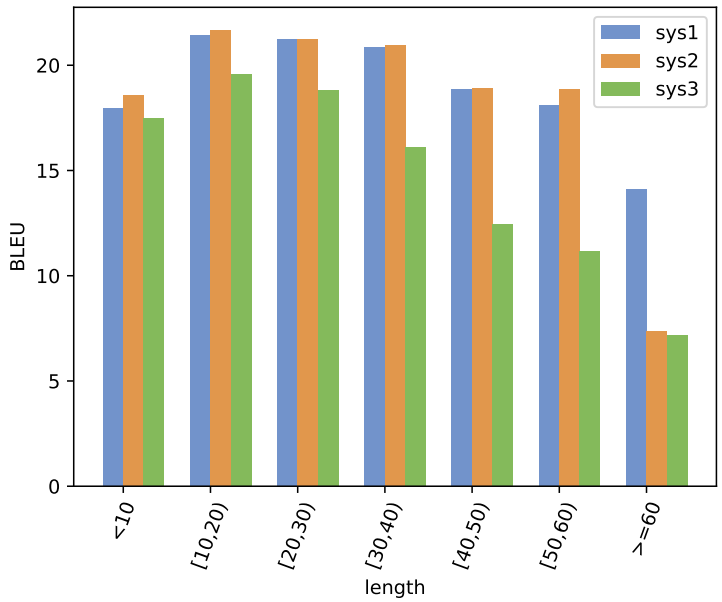



Figure B.2: Compare-mt (Neubig et al., 2019) output sentence length to BLEU for joint decoding vs pure-attention ST models. Model codes: sys1 = Joint Input-Sync, sys2 = Joint Output-Sync, sys3 = Pure-Attn

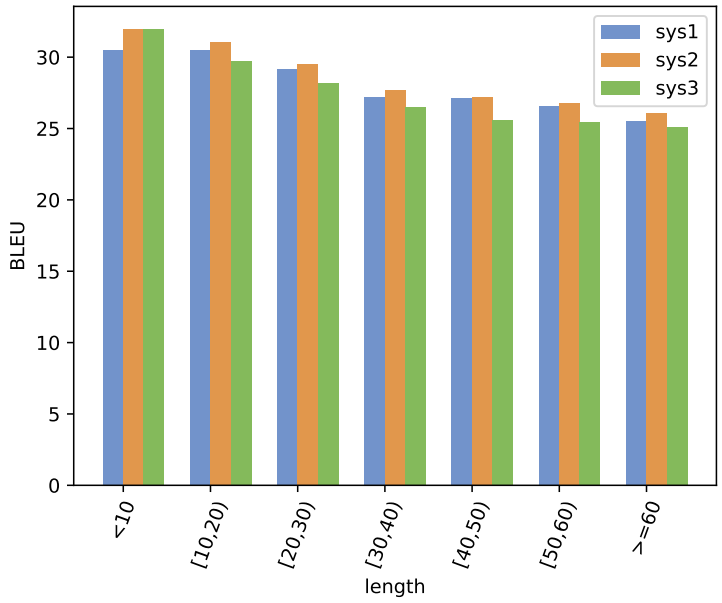



Figure B.3: Compare-mt (Neubig et al., 2019) output sentence length to BLEU for joint decoding vs pure-attention MT models. Model codes: sys1 = Joint Input-Sync, sys2 = Joint Output-Sync, sys3 = Pure-Attn

# Appendix C

# Supplemental Material for Chapter 6

## C.1 Training and Inference hyperparameters

We tune training and inference hyperparameters using only the dev sets. We first determined the best hyperparameters for our baseline Enc-Dec implementation and fixed all settings not pertaining to the unique searchable hidden intermediates of our Multi-Decoder. Then, we find the best hyperparameters for our proposed models under these constraints to demonstrate a true comparison against the baseline. For our Speech-Attention variant, we found that increasing attention dropout in the ST sub-net decoder to 0.4 improved performance, which we verified was not true for the vanilla Multi-Decoder model. For our external model re-scoring, we found that a CTC weight of 0.3 is best for all Multi-Decoder and Multi-Decoder w/ Speech-Attention. The best LM weight for the Multi-Decoder was 0.2, while the best LM weight for the Multi-Decoder w/ Speech-Attention was 0.4. For both of these re-scoring hyperparameters, we tried $[0.2, 0.3, 0.4]$. For deciding the beam size, we use the experiment demonstrated in Figure 6.2 which uses beam sizes of $[1, 4, 8, 10, 16]$.

### C.1.1 Multi-Decoder ST Performance across other automatic MT Metrics

To supplement our overall ST results on the Fisher/CallHome corpus in Table 6.1, which shows BLEU scores, we also evaluated the same Multi-Decoder and Baseline Enc-Dec (Our Implementation) models on two additional metrics: METEOR (Banerjee and Lavie, 2005) and Translation Edit Rate (TER) (Snover et al., 2006). Performance across all three metrics show consistent trends, with the Multi-Decoder outperforming the Baseline Enc-Dec model on all metrics. We see that both the Multi-Decoder and Multi-Decoder w/ Speech-Attention models are improved through ASR Re-scoring. Further, the models with Speech-Attention perform better than those without.

| Model | Fisher test | | | CallHome test | | |
|---|---|---|---|---|---|---|
| | BLEU (↑) | METEOR(↑) | TER(↓) | BLEU (↑) | METEOR(↑) | TER(↓) |
| Baseline Enc-Dec | 49.5 | 37.9 | 42.7 | 18.2 | 22.9 | 68.7 |
| Multi-Decoder | 52.6 | 39.7 | 40.5 | 20.1 | 24.6 | 66.5 |
| +ASR Re-scoring | 53.7 | 40.0 | 39.6 | 20.8 | 24.9 | 65.3 |
| +Speech-Attention | 54.1 | 40.2 | 39.2 | 21.4 | 25.2 | 65.3 |
| +ASR Re-scoring | **55.0** | **40.4** | **38.5** | **21.5** | **25.4** | **64.2** |

Table C.1: Results presenting the performance of our Baseline Enc-Dec implementation and our Multi-Decoder models as evaluated by three metrics: BLEU, METEOR, and Translation Edit Rate (TER). These are the same models as in Table 6.1, which uses BLEU. All results are from the Fisher-CallHome Spanish-English test corpus.

## C.2 MuST-C Data Setup and Model Details

**Data:** We extend our approach to other language pairs from the MuST-C speech translation corpus (Di Gangi et al., 2019). These are recordings of TED talks in English with translations in various target languages. In our experiments we show results on two language pairs, namely, English-German and English-French. We use the provided dev set for deciding the training and inference hyperparameters, as mentioned in Appendix (C.1). We report detokenized case-sensitive BLEU (Post, 2018) on the tst-COMMON set. We apply the same text processing as done in (Inaguma et al., 2020b) and use a joint source and target vocabulary of 8K byte pair encoding (BPE) units (Kudo and Richardson, 2018). Similar to §6.6, we use the ESPnet library to prepare the corpus, and apply the same data preparation and augmentations.

**Multi-Decoder Configuration:** For the MuST-C experiments, we scaled our Multi-Decoder w/ Speech-Attention config from the Fisher-CallHome experiments by increasing the $\text{Encoder}_{\text{ST}}$ to contain 4 transformer encoder blocks. We increased the attention dim and attention heads of the $\text{Encoder}_{\text{ASR}}$ and $\text{Decoder}_{\text{ASR}}$ to 512 dimension and 8 heads respectively, while only increasing the attention dimension to 512 for $\text{Encoder}_{\text{ST}}$ and $\text{Decoder}_{\text{ST}}$. This increased the total trainable parameters to 135M, which we trained on 4 NVIDIA V-100 GPUs for ≈3 days. We also found that increasing the attention dropout of ASR decoder to 0.2 helped with the increased parameters. We kept the remaining dropout parameters the same as our previous experiments. We also keep the remaining training configurations the same like the effective batch-size, learning rate and warmup steps, loss weighting and SpecAugment policy.

During inference, we use the same beam sizes from our Fisher-CallHome experiments and we perform a search across the length penalty and max length ratio settings using the MuST-C dev sets. In the intermediate ASR beam search we use a length penalty of 0.1 and 0.2 for English-German and English-French respectively. In the ST beam search we use a max length ratio of 0.3

and length penalties of 0.6 and 0.5 for English-German and English-French respectively. For our experiments with ASR re-scoring, we use a LM weight of 0.1 and a CTC weight of 0.1. In these re-scoring experiments we also set the ASR length penalty to 0.6 and the ST length penalty to 0.5, while increasing the ST max length ratio to 0.5. The LMs used were trained on the English transcripts of the MuST-C English-German and English-French corpora, with dev perplexities of 32.7 and 23.2 respectively.

# Appendix D

# Supplemental Material for Chapter 7

## D.1 SLURP and SLUE Dataset Description

We evaluated our proposed approach on publicly available SLU datasets, namely SLUE (Shon et al., 2022) and SLURP (Bastianelli et al., 2020) datasets on the task of Named Entity Recognition (NER) from naturally available speech. SLURP is a linguistically diverse and challenging spoken language understanding benchmark that consists of single-turn user conversation with a home assistant, annotated with both intent and entities. Similar to the approach followed in our prior work (Bastianelli et al., 2020; Arora et al., 2019), we bootstrap our train set with 43 hours of synthetic data for all our experiments. We evaluate our approach using SLU-F1 (Bastianelli et al., 2020), a metric for spoken entity prediction, and Label F1, which considers only entity-tag predictions.

SLUE is a recently released SLU benchmark that focuses on Spoken Language Understanding from limited labeled training data. Specifically, it consists of SLUE VoxPopuli dataset that can be used for building systems for ASR and NER. Similar to (Shon et al., 2022), we evaluate our systems using two micro-averaged F1 scores, the first score that evaluates both named entity and tag pairs is referred to as F1, and the second that evaluates only entity-tag phrases is referred to as Label-F1. Note that the released test sets are blind without ground truth labels, and hence we compare different methods using the development set.

The dataset download and evaluation links for SLURP can be found here - https://github.com/pswietojanski/slurp and for SLUE here - https://github.com/asappresearch/slue-toolkit. The datasets have been processed and prepared using ESPnet, SLURP - https://github.com/espnet/espnet/tree/master/egs2/slurp_entity and SLUE - https://github.com/espnet/espnet/tree/master/egs2/slue-voxpopuli

Table D.1: Overview of the two publicly available SLU datasets (Shon et al., 2022; Bastianelli et al., 2020) used for our experiments.

| Dataset | Size (utterances / hours) | | |
|---|---|---|---|
| | Train | Dev | Test |
| SLURP | 11,514 / 40.2 | 2,033 / 6.9 | 2,974 /10.3 |
| SLUE-VoxPopuli | 5,000 / 14.5 | 1,753 / 5.0 | 1,842 / 4.9 |

## D.2 Experimental Setup

Our models are implemented in PyTorch (Paszke et al., 2019), and the experiments are conducted using the ESPnet-SLU toolkit (Arora et al., 2019; Watanabe et al., 2018).

### D.2.1 Speech Preprocessing

Speech inputs are globally mean-variance normalized 80 dimensional logmel filterbanks using a 16kHz sampling and window of 512 frames and a 128 hop length. We apply speed perturbation for the under-resourced dataset of SLUE of 0.9 and 1.1 to increase the samples. We also apply specaugmentation (Park et al., 2019) on both datasets. We also remove all examples smaller than 0.1 seconds and larger than 20 seconds from the training data.

### D.2.2 Text Processing

For the cascaded system, we processed ASR transcripts $S$ using bpe tokenisation (Kudo and Richardson, 2018) and trained ASR models to generate bpe subtokens. We use bpe size of 500 for SLURP and 1000 for SLUE dataset. For the direct E2E models, we predicted the enriched label sequence $Y^e$ using the same bpe size as the ASR models in cascaded sequence. Similarly, compositional models also used the same bpe size to generate the ASR transcripts.

For creating the BIO tags we modified the data preparation such that we take the entities for each utterance and create a "label utterance". This consists of one-to-one mapping of the label tags with the words with Begin (B), Inside (I) and Outside (O) marked for each label. After performing BPE tokenization we add $\varnothing$ for every subtoken of the word. We have attached the data preparation code.

### D.2.3 Model and Training Hyperparameters

We run parameter search for both direct end-to-end and our compositional end-to-end systems using the same model search space (Table D.2). For the cascaded systems we build systems that have the same size as that of our ASR and NLU sub-networks. In this section we will describe

our best architecture for both direct and compositional E2E systems.

**Direct E2E SLU systems** We use encoder decoder based architecture for our baseline E2E system. We use Conformer encoder blocks (Gulati et al., 2020) and Transformer decoder blocks (Vaswani et al., 2017) with CTC multi-tasking (Arora et al., 2019). After our searching through the hyperparameter space we found 12-layer Conformer (Gulati et al., 2020) with 8 attention heads and decoder is a 6-layer Transformer (Vaswani et al., 2017) with 8 attention heads for the SLURP dataset. We use a dropout of 0.1, output dim of 512 and feedforward dim of 2048. This gives a total parameter size of 109.3 M.

For SLUE dataset, we found 12-layer Conformer (Gulati et al., 2020) with 4 attention heads and decoder is a 6-layer Transformer (Vaswani et al., 2017) with 4 attention heads. We use a dropout of 0.1, output dim of 256 and feedforward dim of 1024 in encoder and 2048 in the decoder, giving a total parameter size of 31.2 M.

**Compositional E2E SLU systems** Our Compositional model which uses Direct E2E SLU formulation consists of 12-layer conformer block for encoder, 6-layer transformer block for decoder in it's ASR component and 4-layer transformer encoder and 6-layer transformer decoder in it's NLU component. Each of these attention blocks consist of 8 attention heads, dropout of 0.1, output dim of 512, feedforward dim of 2048, giving a total of 153.9M parameters in SLURP dataset. For SLUE dataset, each of these attention blocks of 4 attention heads, dropout of 0.1, output dim of 256, feedforward dimension of 1024 in encoder and 2048 in decoder, giving a total parameter size of 46.8M.

Our Composition model with Proposed NLU formulation replaces NLU component in Direct E2E formulation with 8-layer transformer encoder followed by linear layer. All these attention blocks consist of 8 attention heads, dropout of 0.1, output dim of 512, feedforward dim of 2048, giving a total of 142.9M parameters in SLURP dataset. For SLUE dataset, each of these attention blocks of 4 attention heads, dropout of 0.1, output dim of 256, feedforward dimension of 1024 in encoder and 2048 in decoder, giving a total parameter size of 43.8M. Our NLU component can further attend to speech representations using cross attention (Dalmia et al., 2021). We further implement CRF loss using publicly available python library [1].

The loss from the ASR ($\mathcal{L}^{\text{asr}}$) and NLU ($\mathcal{L}^{\text{nlu}}$) subnet are combined combined as follows

$$\mathcal{L} = \mathcal{L}^{\text{asr}} + \alpha\mathcal{L}^{\text{nlu}}$$

We search alpha values over [0.3, 0.4, 0.5, 0.6] and found 0.6 to be best for SLURP and 0.3 for SLUE.

[1] https://pytorch-crf.readthedocs.io/en/stable/index.html

| Hyperparameter | Value |
|---|---|
| Output Size | [256, 512] |
| Attention Heads | [4, 8] |
| Number of blocks | [4, 6, 8, 12] |
| Hidden Dropout | [0.1, 0.2] |
| Attention dropout | [0.1, 0.2] |
| Position dropout | [0.1, 0.2] |
| Activation dropout | [0.1, 0.2] |
| Src Activation dropout | [0.1, 0.2] |
| Batch size | [ 50, 64] |
| LR schedule | [inv. sqrt., exp. lr.] |
| Max learning rate | [0.001, 0.002, 0.003] |
| Warmup steps | [5000, 15000, 25000] |
| Number of steps | [50, 70, 100] |
| Adam eps | 1e-9 |
| Adam betas | (0.9, 0.98) |
| Weight decay | 0.000001 |

Table D.2: Model and Training Search for SLU Models.

### D.2.4 Decoding Hyperparameters

We keep the same decoding parameter of beam size and penalty as that of Arora et al. (2019). For direct E2E systems and our models CTC weight of 0.1 worked best. We searched over CTC weight of [0, 0.1, 0.3, 0.5].

### D.2.5 Development Results

We use F1 scores on the validation data to select the best hyperparameters. Table D.3 presents the validation performances for our models.

### D.2.6 Compute Infrastructure

Our models were trained using mixed precision training on either a100, v100 or A6000 on our compute infrastructure depending on their availability. Depending on the GPU and the file i/o latency, the training time ranged from 4-7 hours for SLUE, while for SLURP the training time ranged from 12-18 hours.

[2] https://zenodo.org/record/4630406

| Model | SLURP | | SLUE | |
|---|---|---|---|---|
| | SLU F1 | Label F1 | F1 | Label F1 |
| Casacaded SLU (Ours) | 76.9 | 83.9 | 48.6 | 63.9 |
| Direct E2E SLU (Ours) | 79.2 | 85.4 | 54.7 | 67.6 |
| Compositional E2E SLU | | | | |
| w/ Direct E2E formulation (§7.4) | 79.3 | 86.6 | 50.0 | 68.0 |
| w/ Proposed NLU formulation (§7.5) | | | | |
| CRF w/ Speech Attention (SA) | 79.9 | 87.0 | 59.4 | 73.6 |
| Token Classification w/ SA | 79.8 | 86.9 | 60.3 | 73.7 |
| w/o Speech Attention | 79.7 | 87.0 | 59.0 | 73.6 |

Table D.3: Results presenting the micro F1 performance for all models using CRF and Token Classification modeling on development set for SLURP and SLUE

### D.2.7 External ASR and NLU components

For the experiments in Table 7.2, we used ASR and NLU models trained on external data. For the ASR fine-tuning we used an ESPnet model [2] trained on the GigaSpeech dataset (Chen et al., 2021a). This model has the same architecture as the baseline direct E2E model on SLURP. We initialize both the encoder and decoder for direct E2E SLU and the ASR sub-net for the compositional E2E SLU model. For NLU fine-tuning we used Canine (Clark et al., 2021), a character based BERT language model, which exhibits strong performance on named entity recognition while being able to model token sizes comparable to our SLU systems. [3] We initialize our NLU sub-network without speech attention with Canine and keep the model parameters fixed during training. For finding the best parameters we only tuned the learning rate and LR schedule from Table D.2 and report the best numbers among CRF and Token Classification loss.

For using External ASR Transcripts, we trained an ASR system on SLURP and SLUE we trained an ASR system initialized using GigaSpeech and WavLM (Chen et al., 2021b) respectively. They were then fine-tuned on the respective datasets. These systems achieve 10.0% WER and 9.2% WER on SLURP and SLUE respectively.

[3] https://huggingface.co/google/canine-s

# Bibliography


Bhuvan Agrawal, M. Müller, Martin Radfar, Samridhi Choudhary, A. Mouchtaris, and S. Kunzmann. Tie your embeddings down: Cross-modal latent spaces for end-to-end spoken language understanding. *ArXiv*, abs/2011.09044, 2020. 94, 95, 102

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1139. 82

Tamer Alkhouli, Gabriel Bretschner, Jan-Thorsten Peter, Mohammed Hethnawi, Andreas Guta, and Hermann Ney. Alignment-based neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 54–65, 2016. 58

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR, 2016. 39, 115

Antonios Anastasopoulos and David Chiang. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 82–91, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1008. 78

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016. 6, 7, 43, 77

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *ICML17*, 2017. 7, 43

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796, 2022. 56

Siddhant Arora, Siddharth Dalmia, Pavel Denisov, Xuankai Chang, Yushi Ueda, Yifan Peng,

Yuekai Zhang, Sujay Kumar, Karthik Ganesan, Brian Yan, et al. ESPnet-SLU: Advancing spoken language understanding through espnet. In *Proc. Interspeech*, 2019. 82, 85, 86, 87, 93, 129, 130, 131, 132

Siddhant Arora, Alissa Ostapenko, Vijay Viswanathan, Siddharth Dalmia, Florian Metze, Shinji Watanabe, and Alan W. Black. Rethinking end-to-end evaluation of decomposable tasks: A case study on spoken language understanding. In *Interspeech 2021*, pages 1264–1268. ISCA, 2021. URL https://doi.org/10.21437/Interspeech.2021-1537. 4

Siddhant Arora, Siddharth Dalmia, Florian Metze, Alan W Black, and Shinji Watanabe. Token-level sequence labeling for spoken language understanding using compositional end-to-end models. In *Submission to the Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022. 4

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 37

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33, 2020. 8, 14

Parnia Bahar, Tobias Bieschke, Ralf Schlüter, and Hermann Ney. Tight integrated end-to-end training for cascaded speech translation. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 950–957. IEEE, 2021. URL https://doi.org/10.1109/SLT48900.2021.9383462. 46, 77, 78

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 3, 5, 6, 7, 25, 26, 28, 29, 33, 43, 46, 49, 64, 68

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *ICASSP15*, 2016. 26, 43

Carliss Y. Baldwin and Kim B. Clark. *Design Rules: The Power of Modularity Volume 1*. MIT Press, 1999. 1, 27

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W05-0909. 125

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 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 (NAACL)*, 2019. doi: 10.18653/v1/N19-1006. 94

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. SLURP: A spoken language understanding resource package. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.588. URL https://doi.org/10.18653/v1/2020.emnlp-main.588. 82, 86, 129, 130

Daniel Beck, Trevor Cohn, and Gholamreza Haffari. Neural speech translation using lattice transformations and graph networks. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 26–31, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5304. 5, 7, 64

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. On the linguistic representational power of neural machine translation models. *Computational Linguistics*, 46 (1):1–52, 2020. 78

Swapnil Bhosale, Imran A. Sheikh, Sri Harsha Dumpala, and Sunil Kumar Kopparapu. End-to-end spoken language understanding: Bootstrapping in low resource scenarios. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association*, 2019. 101

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 2017. 100

Léon Bottou, Yoshua Bengio, and Yann Le Cun. Global training of document processing systems using graph transformer networks. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 489–494. IEEE, 1997. 43, 48, 49, 67

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Frederick Jelinek, John Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2), 1990. 43

Emanuele Bugliarello and Naoaki Okazaki. Enhancing machine translation with dependency-aware self-attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1618–1627, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.147. URL https://aclanthology.org/2020.acl-main.147. 58

Lukáš Burget, Petr Schwarz, Mohit Agarwal, Pinar Akyazi, Kai Feng, Arnab Ghoshal, Ondřej Glembek, Nagendra Goel, Martin Karafiát, Daniel Povey, et al. Multilingual acoustic modeling for speech recognition based on subspace Gaussian mixture models. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference On*, pages 4334–4337. IEEE, 2010. 16

Frédéric Béchet, Allen L Gorin, Jeremy H Wright, and Dilek Hakkani Tür. Detecting and extracting named entities from spontaneous speech in a mixed-initiative spoken dialogue context: How may i help you?sm,tm. *Speech Communication*, 42(2):207–225, 2004. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2003.07.003. URL https://www.sciencedirect.com/science/article/pii/S016763930300116X. 82, 84

Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit3: Web inventory of transcribed and translated talks. In *Conference of european association for machine translation*, pages 261–268, 2012. 55

William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015. 5

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, Attend and Spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP16*, 2016. 3, 6, 25, 26, 28, 43, 84

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. Imputer: Sequence modelling via imputation and dynamic programming. In *ICML20*, 2020. 5, 43

Aditi Chaudhary, Siddharth Dalmia, Junjie Hu, Xinjian Li, Austin Matthews, Aldrian Obaja Muis, Naoki Otani, Shruti Rijhwani, Zaid Sheikh, Nidhi Vyas, et al. The ariel-cmu systems for lorehlt18. *arXiv preprint arXiv:1902.08899*, 2019. 4

Guoguo Chen et al. GigaSpeech: An evolving, multi-domain ASR corpus with 10,000 hours of transcribed audio. *arXiv preprint arXiv:2106.06909*, 2021a. 133

Nanxin Chen, Shinji Watanabe, Jesús Villalba, and Najim Dehak. Non-autoregressive transformer automatic speech recognition. *arXiv preprint arXiv:1911.04908*, 2019. 43

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, and Furu Wei. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *CoRR*, abs/2110.13900, 2021b. URL https://arxiv.org/abs/2110.13900. 133

Yixin Chen, Weiyi Lu, Alejandro Mottini, Erran Li, Jasha Droppo, Zheng Du, and Belinda Zeng. Top-down attention in end-to-end spoken language understanding. In *ICASSP 2021*, 2021c. 94

Shun-Po Chuang, Yung-Sung Chuang, Chih-Chiang Chang, and Hung-yi Lee. Investigating the reordering capability in CTC-based non-autoregressive end-to-end speech translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1068–1077, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.92. URL https://aclanthology.org/2021.findings-acl.92. 46

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. CANINE: pre-training an efficient tokenization-free encoder for language representation. *CoRR*, abs/2103.06874, 2021. URL https://arxiv.org/abs/2103.06874. 133

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*, 2020. 8

Linguistic Data Consortium. 2000 HUB5 english evaluation transcripts LDC2002T43. *Web Download. Philadelphia: Linguistic Data Consortium*, 2002a. 22, 37

Linguistic Data Consortium. 2000 HUB5 english evaluation speech LDC2002S09. *Web Download. Philadelphia: Linguistic Data Consortium*, 2002b. 22, 37

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. In *Privacy in Machine Learning and Artificial Intelligence workshop, ICML*, 2018. 5, 64, 95

Siddharth Dalmia, Ramon Sanabria, Florian Metze, and Alan W. Black. Sequence-Based Multi-Lingual Low Resource Speech Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4909–4913, April 2018. doi: 10.1109/ICASSP.2018.8461802. 3, 8, 22

Siddharth Dalmia, Xinjian Li, Alan W Black, and Florian Metze. Phoneme Level Language Models for Sequence Based Low Resource ASR. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6091–6095, May 2019a. doi: 10.1109/ICASSP.2019.8683225. 4, 8, 13

Siddharth Dalmia, Abdelrahman Mohamed, Mike Lewis, Florian Metze, and Luke Zettlemoyer. Enforcing Encoder-Decoder Modularity in Sequence-to-Sequence Models. *arXiv:1911.03782 [cs]*, November 2019b. 2, 4, 78

Siddharth Dalmia, Brian Yan, Vikas Raunak, Florian Metze, and Shinji Watanabe. Searchable hidden intermediates for end-to-end models of decomposable sequence tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021. 4, 46, 82, 85, 86, 94, 131

Siddharth Dalmia, Dmytro Okhonko, Mike Lewis, Sergey Edunov, Shinji Watanabe, Florian Metze, Luke Zettlemoyer, and Abdelrahman Mohamed. Legonn: Building modular encoder decoder models. In *Submission to the 2022 Transactions of Audio, Speech and Language Processing (TASLP)*, 2022. 4, 46, 51, 107

Hiroyuki Deguchi, Akihiro Tamura, and Takashi Ninomiya. Synchronous syntactic attention for transformer neural machine translation. In *Proceedings of the 59th Annual Meeting of*

*the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 348–355, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-srw.36. URL https://aclanthology.org/2021.acl-srw.36. 58

Keqi Deng, Shinji Watanabe, Jiatong Shi, and Siddhant Arora. Blockwise streaming transformer for spoken language understanding and simultaneous speech translation. *arXiv preprint arXiv:2204.08920*, 2022. 46, 51

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 7, 43

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *proceedings of the 52nd annual meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, 2014. 59

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. 8, 27, 83, 93, 99, 107, 108

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2012–2017, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1202. URL https://www.aclweb.org/anthology/N19-1202. 55, 76, 126

Marco Dinarelli, Nikita Kapoor, Bassam Jabaian, and Laurent Besacier. A data efficient end-to-end spoken language understanding architecture. In *ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 8519–8523. IEEE, 2020. 94

Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE, 2018. 51

Qianqian Dong, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. SDST: Successive decoding for speech-to-text translation. *arXiv preprint arXiv:2009.09737*, 2020. 78

Timothy Dozat, Peng Qi, and Christopher D. Manning. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/

v1/K17-3002. URL https://aclanthology.org/K17-3002. 82

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010. 94

Ryo Fukuda, Yuka Ko, Yasumasa Kano, Kosuke Doi, Hirotaka Tokuyama, Sakriani Sakti, Katsuhito Sudoh, and Satoshi Nakamura. NAIST simultaneous speech-to-text translation system for IWSLT 2022. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 286–292, Dublin, Ireland (in-person and online), May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.iwslt-1.25. URL https://aclanthology.org/2022.iwslt-1.25. 54

Yashesh Gaur, Walter S. Lasecki, Florian Metze, and Jeffrey P. Bigham. The effects of automatic speech recognition quality on human transcription latency. In *Proceedings of the 13th International Web for All Conference*, W4A '16, 2016. ISBN 9781450341387. 97

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *ICML17*, 2017. 33

Xinwei Geng, Xiaocheng Feng, Bing Qin, and Ting Liu. Adaptive multi-pass decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 523–532, Brussels, Belgium, 10. Association for Computational Linguistics. doi: 10.18653/v1/D18-1048. 78

S. Ghannay, A. Caubrière, Y. Estève, N. Camelin, E. Simonnet, A. Laurent, and E. Morin. End-to-end named entity and semantic concept extraction from speech. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 692–699, 2018. doi: 10.1109/SLT.2018.8639513. 82

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019. 43

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. Aligned cross entropy for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 3515–3523. PMLR, 2020a. 43, 46, 58, 111

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. Aligned cross entropy for non-autoregressive machine translation. In *ICML20*, 2020b. 111

Arnab Ghoshal, Pawel Swietojanski, and Steve Renals. Multilingual training of deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference On*, pages 7319–7323. IEEE, 2013. 16

John Godfrey and Edward Holliman. Switchboard-1 release 2 LDC97S62. *Web Download. Philadelphia: Linguistic Data Consortium*, 1993. 22, 37

Google. Google speech to text api. https://cloud.google.com/speech-to-text, 2021. Accessed: 2021-03-15. 96, 97, 101

Allen L Gorin, Giuseppe Riccardi, and Jeremy H Wright. How may i help you? *Speech communication*, 23(1-2):113–127, 1997. 94

Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012. 5, 48, 58, 84

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 369–376. ACM, 2006. 11, 12, 16, 27, 29, 43, 46, 48

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (Icassp), 2013 Ieee International Conference On*, pages 6645–6649. IEEE, 2013. 43

František Grézl, Martin Karafiat, and Miloš Janda. Study of probabilistic and bottle-neck features in multilingual environment. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop On*, pages 359–364. IEEE, 2011. 16

Frantisek Grézl, Martin Karafiát, and Karel Vesely. Adaptation of multilingual stacked bottle-neck neural network structure for new language. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference On*, pages 7654–7658. IEEE, 2014. 16

František Grézl, Ekaterina Egorova, and Martin Karafiát. Study of large data resources for multilingual training and system porting. *Procedia Computer Science*, 81:15–22, 2016. 15

Jiatao Gu and Xiang Kong. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 120–133, 2021. 46

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *ICLR18*, 2018. 5, 43

Anmol Gulati et al. Conformer: Convolution-augmented transformer for speech recognition. In *Proc. Interspeech*, pages 5036–5040, 2020. 52, 106, 131

Pengcheng Guo, Florian Boyer, Xuankai Chang, Tomoki Hayashi, Yosuke Higuchi, Hirofumi Inaguma, Naoyuki Kamo, Chenda Li, Daniel Garcia-Romero, Jiatong Shi, et al. Recent developments on ESPnet toolkit boosted by conformer. *arXiv preprint arXiv:2010.13956*, 2020.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. Neural module networks for reasoning over text. In *ICLR20*, 2020. 7, 43

Parisa Haghani, Arun Narayanan, Michiel Bacchiani, Galen Chuang, Neeraj Gaur, Pedro Moreno, Rohit Prabhavalkar, Zhongdi Qu, and Austin Waters. From audio to semantics: Approaches to end-to-end spoken language understanding. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 720–726. IEEE, 2018. 78

Awni Hannun. The Label Bias Problem. https://awni.github.io/label-bias, 2019. 48

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014a. 43

Awni Hannun, Vineel Pratap, Jacob Kahn, and Wei-Ning Hsu. Differentiable weighted finite-state transducers. *arXiv preprint arXiv:2010.01003*, 2020. 107

Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *arXiv preprint arXiv:1408.2873*, 2014b. 52, 54, 59

Adi Haviv, Lior Vassertail, and Omer Levy. Can latent alignments improve autoregressive machine translation? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2637–2641, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.209. URL https://aclanthology.org/2021.naacl-main.209. 46

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. Improved neural machine translation with smt features. In *Thirtieth AAAI conference on artificial intelligence*, 2016. 53

Georg Heigold, Vincent Vanhoucke, Alan Senior, Patrick Nguyen, M Ranzato, Matthieu Devin, and Jeffrey Dean. Multilingual acoustic models using distributed deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference On*, pages 8619–8623. IEEE, 2013. 16

Jindřich Helcl, Jindřich Libovický, and Dušan Variš. CUNI system for the WMT18 multimodal translation task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 616–623, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6441. 69

Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990. 94, 95

Yosuke Higuchi, Keita Karube, Tetsuji Ogawa, and Tetsunori Kobayashi. Hierarchical conditional end-to-end asr with ctc and multi-granular subword units. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7797–7801. IEEE, 2022. 51

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012. doi: 10.1109/MSP.2012.2205597. 2, 48, 64

Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan. Advances in joint CTC-Attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. In *Proc. Interspeech 2017*, pages 949–953, 2017. doi: 10.21437/Interspeech.2017-1296. 67, 73

James Horlock and Simon King. Discriminative methods for improving named entity extraction on speech data. In *INTERSPEECH*, 2003. 82

Ke Hu, Tara N Sainath, Ruoming Pang, and Rohit Prabhavalkar. Deliberation model based two-pass end-to-end speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7799–7803. IEEE, 2020. 78

Liang Huang and David Chiang. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June 2007. Association for Computational Linguistics. 5, 64

Y. Huang, H. K. Kuo, S. Thomas, Z. Kons, K. Audhkhasi, B. Kingsbury, R. Hoory, and M. Picheny. Leveraging unpaired text data for training end-to-end speech-to-intent systems. In *ICASSP 2020*, 2020. 94

Hirofumi Inaguma, Yosuke Higuchi, Kevin Duh, Tatsuya Kawahara, and Shinji Watanabe. Orthros: Non-autoregressive end-to-end speech translation with dual-decoder. *arXiv preprint arXiv:2010.13047*, 2020a. 14

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Yalta, Tomoki Hayashi, and Shinji Watanabe. ESPnet-ST: All-in-one speech translation toolkit. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 302–311, Online, July 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.34. 5, 7, 64, 70, 126

Hirofumi Inaguma, Siddharth Dalmia, Brian Yan, and Shinji Watanabe. Fast-md: Fast multi-decoder end-to-end speech translation with non-autoregressive hidden intermediates. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 922–929. IEEE, 2021a. 46, 108

Hirofumi Inaguma, Brian Yan, Siddharth Dalmia, Pengcheng Guo, Jiatong Shi, Kevin Duh, and Shinji Watanabe. Espnet-st iwslt 2021 offline speech translation system. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 100–109, 2021b. 4

Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, Javier Jorge, Nahuel Roselló, Adrià Giménez, Albert Sanchis, Jorge Civera, and Alfons Juan. Europarl-ST: A multilingual corpus for speech translation of parliamentary debates. In *ICASSP20*, 2020. 37, 57, 112

Frederick Jelinek. *Statistical Methods for Speech Recognition*. MIT press, 1997. 43, 58

Ye Jia, Ron J. Weiss, Fadi Biadsy, Wolfgang Macherey, Melvin Johnson, Zhifeng Chen, and Yonghui Wu. Direct speech-to-speech translation with a sequence-to-sequence model. In Gernot Kubin and Zdravko Kacic, editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association*. ISCA, 2019. 94

Michael Johnson. Compositionality. *The Wiley Blackwell Companion to Semantics*, pages 1–27, 2020. 1

Patricia Johnson. Cohesion and coherence in compositions in malay and english. *RELC Journal*, 23(2):1–17, 1992. doi: 10.1177/003368829202300201. 1, 64

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2), 1994. 43

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, USA, 1st edition, 2000. ISBN 0130950696. 82

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP13*, 2013. 28, 43

Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. End-to-end speech translation with transcoding by multi-task learning for distant language pairs. *IEEE ACM Trans. Audio Speech Lang. Process.*, 28:1342–1355, 2020. 94

Shigeki Karita, Nanxin Chen, Tomoki Hayashi, and other. A comparative study on transformer vs RNN in speech applications. In *ASRU19*, 2019. 43

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A Smith. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *ICLR*, 2021. 46

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)*, pages 4835–4839. IEEE, 2017. 11, 43, 45, 49, 78

Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR14*, 2014. 37, 70

KM Knill, Mark JF Gales, Shakti P Rath, Philip C Woodland, Chao Zhang, and S-X Zhang. Investigation of multilingual deep neural networks for spoken term detection. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop On*, pages 138–143. IEEE, 2013. 15

Philipp Koehn and Josh Schroeder. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, 2007. 5, 64

Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133, 2003. URL https://aclanthology.org/N03-1017. 58

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 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*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. 2, 5, 7, 64

Francis Kubala, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. Named entity extraction from speech. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, pages 287–292. Citeseer, 1998. 82

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, 2018. 30, 37, 70, 126, 130

S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1), 1951. 97

G. Kumar, M. Post, D. Povey, and S. Khudanpur. Some insights from translating conversational telephone speech. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3231–3235, 2014. doi: 10.1109/ICASSP.2014.6854197. 70

Shankar Kumar, Yonggang Deng, and William Byrne. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(1): 35–76, 2006. 64

Chih-Hua Kuo. Cohesion and coherence in academic writing: From lexical choice to organization. *RELC Journal*, 26(1):47–62, 1995. doi: 10.1177/003368829502600103. 1, 64

Hong-Kwang J. Kuo, Zoltán Tüske, Samuel Thomas, Yinghui Huang, Kartik Audhkhasi, Brian Kingsbury, Gakuto Kurata, Zvi Kons, Ron Hoory, and Luis Lastras. End-to-End Spoken Language Understanding Without Full Transcripts. In *Interspeech 2020, 21th Annual Conference of the International Speech Communication Association*. ISCA, 2020. 94

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eigh-*

*teenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781. 83

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882, 2018. 64

Brenden M Lake. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, pages 9791–9801, 2019. 6, 77

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1030. URL https://aclanthology.org/N16-1030. 82, 83

Hang Le, Juan Pino, Changhan Wang, Jiatao Gu, Didier Schwab, and Laurent Besacier. Dual-decoder transformer for joint automatic speech recognition and multilingual speech translation. *arXiv preprint arXiv:2011.00747*, 2020. 78

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP18*, 2018. 38, 111

Alexander H. Levis, Neville Moray, and Baosheng Hu. Task decomposition and allocation problems and discrete event systems. *Automatica*, 30(2):203–216, 1994. ISSN 0005-1098. doi: 10.1016/0005-1098(94)90025-6. 1, 64

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. 107

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. Question and answer test-train overlap in open-domain question answering datasets. *arXiv preprint arXiv:2008.02637*, 2020. 94

Xian Li, Changhan Wang, Yun Tang, Chau Tran, Yuqing Tang, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. Multilingual speech translation with efficient finetuning of pre-trained models. *arXiv e-prints*, pages arXiv–2010, 2020a. 8

Xinjian Li, Siddharth Dalmia, Alan W. Black, and Florian Metze. Multilingual Speech Recognition with Corpus Relatedness Sampling. In *Interspeech 2019*, pages 2120–2124. ISCA, September 2019. doi: 10.21437/Interspeech.2019-3052. 4

Xinjian Li, Siddharth Dalmia, Juncheng Li, Matthew Lee, Patrick Littell, Jiali Yao, Antonios Anastasopoulos, David R Mortensen, Graham Neubig, Alan W Black, et al. Universal phone

recognition with a multilingual allophone system. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8249–8253. IEEE, 2020b. 4

Jindřich Libovický and Jindřich Helcl. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *EMNLP18*, 2018. 14, 33, 43, 46

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020a. 8

Yuchen Liu, Jiajun Zhang, Hao Xiong, Long Zhou, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong. Synchronous speech recognition and speech-to-text translation with interactive decoding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 8417–8424, 2020b. 78

Zhiyuan Liu, Yankai Lin, and Maosong Sun. Compositional semantics. In *Representation Learning for Natural Language Processing*, pages 43–57. Springer Singapore, Singapore, 2020c. ISBN 978-981-15-5573-2. doi: 10.1007/978-981-15-5573-2□. 1, 64

Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. Speech Model Pre-Training for End-to-End Spoken Language Understanding. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association*, 2019. 7, 94, 95, 96, 97, 98, 99, 100, 101, 102

Loren Lugosch, B. Meyer, Derek Nowrouzezahrai, and M. Ravanelli. Using speech synthesis to train end-to-end spoken language understanding models. *ICASSP 2020*, 2020. 101

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *EMNLP15*, 2015. 28

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1101. URL https://aclanthology.org/P16-1101. 83

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 591–598, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072. 83

Erik McDermott. *Discriminative training for speech recognition*. PhD thesis, Citeseer, 1997. 108

Joseph P. McKenna, Samridhi Choudhary, Michael Saxon, Grant P. Strimel, and Athanasios

Mouchtaris. Semantic complexity in end-to-end spoken language understanding. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association*, 2020. 95, 98, 99, 100

Clara Meister, Tim Vieira, and Ryan Cotterell. Best-first beam search. *Transactions of the Association for Computational Linguistics*, 2020. 58

Donald Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Inf. Retr.*, 2007. 97

Bernd T Meyer, Sri Harish Mallidi, Angel Mario Castro Martinez, Guillermo Payá-Vayá, Hendrik Kayser, and Hynek Hermansky. Performance monitoring for automatic speech recognition in noisy multi-channel environments. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 50–56. IEEE, 2016. 5, 7, 64

Yajie Miao, Mohammad Gowayyed, and Florian Metze. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop On*, pages 167–174. IEEE, 2015. 11, 13, 16, 18, 19, 39, 59, 108, 115

Yajie Miao, Mohammad Gowayyed, Xingyu Na, Tom Ko, Florian Metze, and Alexander Waibel. An empirical exploration of CTC acoustic models. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference On*, pages 2623–2627. IEEE, 2016. 17

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013. 2

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *LREC 2018*, 2018. 99

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. *Commun. ACM*, 61(5):103–115, apr 2018. ISSN 0001-0782. doi: 10.1145/3191513. URL https://doi.org/10.1145/3191513. 89

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. Transformers with convolutional context for ASR. In *arXiv Preprint arXiv:1904.11660*, 2019. 37

Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002. 2, 64

David R. Mortensen, Siddharth Dalmia, and Patrick Littell. Epitran: Precision G2P for Many Languages. In *Proceedings of the Eleventh International Conference on Language Resources*

*and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). 22

Sudha Morwal, Nusrat Jahan, and Deepti Chopra. Named entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing*, 1:15–23, 12 2012. doi: 10.5121/ijnlc.2012.1402. 83

Moses-SMT. multi-bleu.perl. *GitHub repository*, 2018. 38, 120

Markus Müller, Sebastian Stüker, and Alex Waibel. Language adaptive multilingual CTC speech recognition. In *Proc. SPECOM*, Hatfield, UK, 2017. 17

Mathias Müller, Annette Rios, and Rico Sennrich. Domain robustness in neural machine translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 151–164, Virtual, October 2020. Association for Machine Translation in the Americas. 79

Kenton Murray and David Chiang. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6322. URL https://aclanthology.org/W18-6322. 49, 79

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, 2018. 36

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, Xinyi Wang, and John Wieting. compare-mt: A tool for holistic comparison of language generation systems. *CoRR*, abs/1903.07926, 2019. URL http://arxiv.org/abs/1903.07926. 124

nvidia. cuDNN CTC loss. https://docs.nvidia.com/deeplearning/cudnn/api/index.html#cudnnCTCLoss_v8, 2022. 42

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, 2002. 5, 64

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *WMT18*, 2018. 38, 39, 112, 117

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. Fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (EMNLP): Demonstrations*, 2019. 37, 54

Shruti Palaskar, Vikas Raunak, and Florian Metze. Learned in speech recognition: Contextual acoustic word embeddings. In *ICASSP 2019-2019 IEEE International Conference on Acous-*

*tics, Speech and Signal Processing (ICASSP)*, pages 6530–6534. IEEE, 2019. 7

David D. Palmer and Mari Ostendorf. Improving information extraction by modeling errors in speech recognizer output. In *Proceedings of the First International Conference on Human Language Technology Research*, 2001. URL https://aclanthology.org/H01-1034. 82

V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *ICASSP 2015*, 2015. 99

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. 70, 98

Carolina Parada, Mark Dredze, and Frederick Jelinek. Oov sensitive named-entity recognition in speech. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011. 84

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. SpecAugment: A simple data augmentation method for automatic speech recognition. *Proc. Interspeech 2019*, pages 2613–2617, 2019. 2, 3, 13, 21, 22, 38, 70, 116, 130

Kyubyong Park and Jongseok Kim. g2pE. *GitHub repository*, 2019. 37, 112

Adam Paszke et al. PyTorch: An imperative style, high-performance deep learning library. *Proc. NeurIPS*, 2019. 108, 130

Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: A study of generalization via modularity. In *NIPS19*, 2019. 7, 43

Vijayaditya Peddinti, Guoguo Chen, Vimal Manohar, Tom Ko, Daniel Povey, and Sanjeev Khudanpur. JHU ASpIRE system: Robust LVCSR with TDNNS, iVector adaptation and RNN-LMS. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 539–546. IEEE, 2015. 5, 64

Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe. Branchformer: Parallel MLP-Attention Architectures to Capture Local and Global Context for Speech Recognition and Understanding. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022. 106

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018. 108

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*

*(EMNLP 2020): Systems Demonstrations*, pages 46–54, Online, 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.7. 108

N. Pham, Thai-Son Nguyen, Thanh-Le Ha, J. Hussain, Felix Schneider, J. Niehues, Sebastian Stüker, and A. Waibel. The IWSLT 2019 KIT speech translation system. In *International Workshop on Spoken Language Translation (IWSLT)*, 2019. 2, 5, 7, 64

Matt Post. A call for clarity in reporting BLEU scores. In *WMT18*, 2018. 55, 70, 113, 126

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. Improved speech-to-text translation with the fisher and callhome Spanish–English speech translation corpus. In *International Workshop on Spoken Language Translation (IWSLT 2013)*, 2013. 70

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011. 5, 7, 84

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for ASR based on lattice-free MMI. In *InterSpeech16*, 2016.

Senthil Purushwalkam, Maximilian Nickel, Abhinav Gupta, and Marc'Aurelio Ranzato. Task-driven modular networks for zero-shot compositional learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 7, 43

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. Glancing transformer for non-autoregressive neural machine translation. 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)*, pages 1993–2003, Online, August 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.155. URL https://aclanthology.org/2021.acl-long.155. 46

Yao Qian, Ximo Bian, Yu Shi, Naoyuki Kanda, Leo Shen, Zhen Xiao, and Michael Zeng. Speech-language pre-training for end-to-end spoken language understanding. *arXiv preprint arXiv:2102.06283*, 2021b. 94

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. A stack-propagation framework with token-level intent detection for spoken language understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019. 95

Lance A Ramshaw and Mitchell P Marcus. Text chunking using transformation-based learning.

In *Natural language processing using very large corpora*, pages 157–176. Springer, 1999. 84

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016*, 2016. 48, 49

Vikas Raunak, Vaibhav Kumar, and Florian Metze. On compositionality in neural machine translation. *NeurIPS Workshop, Context and Compositionality in Biological and Artificial Neural Systems*, 2019. URL https://arxiv.org/abs/1911.01497. 6, 77

Vikas Raunak, Siddharth Dalmia, Vivek Gupta, and Florian Metze. On long-tailed phenomena in neural machine translation. In *EMNLP (Findings)*, 2020. 54

Abhilasha Ravichander, Siddharth Dalmia, Maria Ryskina, Florian Metze, Eduard Hovy, and Alan W Black. NoiseQA: Challenge Set Evaluation for User-Centric Question Answering. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Online, April 2021. URL https://arxiv.org/abs/2102.08345. 107

Raj Reddy. Foundations and grand challenges of artificial intelligence: AAAI presidential address. *AI Magazine*, 9(4):9–9, 1988. 67

Yi Ren, Jinglin Liu, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. SimulSpeech: End-to-end simultaneous speech to text translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3787–3796, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.350. URL https://aclanthology.org/2020.acl-main.350. 108

Shruti Rijhwani, Antonios Anastasopoulos, and Graham Neubig. OCR Post Correction for Endangered Language Texts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5931–5942, Online, November 2020. Association for Computational Linguistics. 78

Anthony Rousseau, Paul Deléglise, and Yannick Estève. Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *LREC14*, 2014. 37, 112

A. Saade, A. Coucke, A. Caulier, J. Dureau, Adrien Ball, Théodore Bluche, D. Leroy, Clément Doumouro, Thibault Gisselbrecht, F. Caltagirone, Thibaut Lavril, and Maël Primet. Spoken language understanding on the edge. In *Energy Efficient Machine Learning and Cognitive Computing workshop, NeurIPS*, 2019. 95

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-autoregressive machine translation with latent alignments. *arXiv preprint arXiv:2004.07437*, 2020. 5, 14, 43, 46, 58

Tara N Sainath, Ruoming Pang, David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirkó Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, et al. Two-pass end-to-end speech recog-

nition. *Proc. Interspeech 2019*, pages 2773–2777, 2019. 52, 78

Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014. 43

Elizabeth Salesky, Matthias Sperber, and Alan W Black. Exploring Phoneme-Level Speech Representations for End-to-End Speech Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1835–1841, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1179. URL https://www.aclweb.org/anthology/P19-1179. 78

Elizabeth Salesky, Matthew Wiesner, Jacob Bremerman, Roldano Cattoni, Matteo Negri, Marco Turchi, Douglas W Oard, and Matt Post. The multilingual tedx corpus for speech recognition and translation. *arXiv preprint arXiv:2102.01757*, 2021. 55

Lahiru Samarakoon, Brian Mak, and Albert YS Lam. Domain adaptation of end-to-end speech recognition in low-resource settings. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 382–388. IEEE, 2018. 79

Ramon Sanabria and Florian Metze. Hierarchical multitask learning with ctc. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 485–490. IEEE, 2018. 13, 14, 51

George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. English conversational telephone speech recognition by humans and machines. *arXiv preprint arXiv:1703.02136*, 2017. 15

George Saon, Zoltán Tüske, and Kartik Audhkhasi. Alignment-length synchronous decoding for rnn transducer. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7804–7808. IEEE, 2020. 52

Stefano Scanzio, Pietro Laface, Luciano Fissore, Roberto Gemello, and Franco Mana. On the use of a multilingual neural network front-end. In *Ninth annual conference of the international speech communication association (InterSpeech)*. ISCA, 2008. 16

Tanja Schultz and Alex Waibel. Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication*, 35(1):31–51, 2001. 16

Hiroshi Seki, Takaaki Hori, Shinji Watanabe, Niko Moritz, and Jonathan Le Roux. Vectorized beam search for CTC-Attention-Based speech recognition. In *Proc. Interspeech 2019*, pages 3825–3829, 2019. doi: 10.21437/Interspeech.2019-2860. 53, 70

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

2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. 30, 38, 115

Jiatong Shi, Jonathan D. Amith, Xuankai Chang, Siddharth Dalmia, Brian Yan, and Shinji Watanabe. Highland puebla nahuatl–spanish speech translation corpus for endangered language documentation. In *The First Workshop on NLP for Indigenous Languages of the Americas (AmericasNLP)*, 2021. 4

Suwon Shon, Ankita Pasad, Felix Wu, Pablo Brusco, Yoav Artzi, Karen Livescu, and Kyu J. Han. SLUE: New benchmark tasks for spoken language understanding evaluation on natural speech. In *ICASSP*, 2022. 82, 85, 86, 129, 130

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, 2006. URL http://www.cs.umd.edu/~snover/pub/amta06/ter_amta.pdf. 125

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *NIPS13*, 2013. 7, 43

Kai Song, Kun Wang, Heng Yu, Yue Zhang, Zhongqiang Huang, Weihua Luo, Xiangyu Duan, and Min Zhang. Alignment-enhanced transformer for constraining nmt with pre-specified translations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8886–8893, 2020. 58

Pavel Sountsov and Sunita Sarawagi. Length bias in encoder decoder models and a case for global conditioning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1525, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1158. 76

Matthias Sperber and Matthias Paulik. Speech translation and the end-to-end promise: Taking stock of where we are. In *Association for Computational Linguistic (ACL)*, Seattle, USA, 2020. 7, 8, 77, 94

Matthias Sperber, Graham Neubig, J. Niehues, and Alexander H. Waibel. Attention-passing models for robust and data-efficient end-to-end speech translation. *Transactions of the Association for Computational Linguistics*, 7:313–325, 2019. 78

Andreas Stolcke, Frantisek Grezl, Mei-Yuh Hwang, Xin Lei, Nelson Morgan, and Dimitra Vergyri. Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference On*, volume 1, pages I–I. IEEE, 2006. 15, 16

Sebastian Stuker, Florian Metze, Tanja Schultz, and Alex Waibel. Integrating multilingual articulatory features into speech recognition. In *Eighth European Conference on Speech Communication and Technology*, 2003. 16

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014. 6, 26, 28, 29, 43, 64

Pawel Swietojanski, Arnab Ghoshal, and Steve Renals. Unsupervised cross-lingual knowledge transfer in DNN-based LVCSR. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 246–251. IEEE, 2012. 16

Christoph Tillmann and Hermann Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational linguistics*, 29(1):97–133, 2003. 5, 7, 64

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proceedings of 37th Annual Allerton Conference on Communication, Control and Computing (Alletron)*, 1999. 34

Sibo Tong, Philip N Garner, and Hervé Bourlard. An investigation of deep neural networks for multilingual speech recognition training and adaptation. In *Proc. of INTERSPEECH*, 2017. 16

Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. In *Proc. Interspeech 2017*, pages 3532–3536, 2017. doi: 10.21437/Interspeech.2017-1118. 78

Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Mari Ostendorf. Parsing speech: a neural approach to integrating lexical and acoustic-prosodic information. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 69–81. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1007. URL https://doi.org/10.18653/v1/n18-1007. 82

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. Neural machine translation with reconstruction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3097–3103, 2017. 78

Zoltán Tüske, George Saon, Kartik Audhkhasi, and Brian Kingsbury. Single headed attention based sequence-to-sequence model for state-of-the-art results on Switchboard-300. *arXiv preprint arXiv:2001.07263*, 2020.

Evelyne Tzoukermann and Corey Miller. Evaluating automatic speech recognition in translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 2: User Track)*, pages 294–302, Boston, MA, March 2018. Association for Machine Translation in the Americas. 64

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Infor-*

*mation Processing Systems*, pages 5998–6008, 2017. 2, 3, 5, 6, 12, 13, 21, 26, 32, 33, 34, 37, 38, 43, 51, 64, 68, 106, 112, 121, 131

Karel Veselỳ, Martin Karafiát, František Grézl, Miloš Janda, and Ekaterina Egorova. The language-independent bottleneck features. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 336–341. IEEE, 2012. 15, 16, 18

Ngoc Thang Vu, Florian Metze, and Tanja Schultz. Multilingual bottle-neck features and its application for under-resourced languages. In *Proc. 3rd Workshop on Spoken Language Technologies for under-Resourced Languages*, Cape Town; S. Africa, May 2012. MICA. 15

Ngoc Thang Vu, David Imseng, Daniel Povey, Petr Motlicek, Tanja Schultz, and Hervé Bourlard. Multilingual deep neural network based acoustic modeling for rapid language adaptation. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference On*, pages 7639–7643. IEEE, 2014. 16

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. Fairseq S2T: Fast speech-to-text modeling with fairseq. In *Proceedings of the 2020 Conference of the Asian Chapter of the Association for Computational Linguistics (AACL): System Demonstrations*, 2020a. 75

Changhan Wang, Anne Wu, and Juan Pino. Covost 2: A massively multilingual speech-to-text translation corpus. *arXiv preprint arXiv:2007.10310*, 2020b. 75

Chengyi Wang, Shuangzhi Wu, and Shujie Liu. Source dependency-aware transformer with supervised self-attention. *arXiv preprint arXiv:1909.02273*, 2019. 58

Chengyi Wang, Yu Wu, Shujie Liu, Zhenglu Yang, and Ming Zhou. Bridging the gap between pretraining and fine-tuning for end-to-end speech translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9161–9168, April 2020c. doi: 10.1609/aaai.v34i05.6452. 78

Dong Wang and Thomas Fang Zheng. Transfer learning for speech and language processing. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2015. 94

S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi. Hybrid CTC/Attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11 (8):1240–1253, 2017. doi: 10.1109/JSTSP.2017.2763455. 46, 47, 49, 50, 52, 54, 70, 71

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. ESPnet: End-to-end speech processing toolkit. In *Proc. Interspeech 2018*, pages 2207–2211, 2018. doi: 10.21437/Interspeech.2018-1456. 22, 23, 37, 55, 70, 107, 116, 130

Shintaro Harada Taro Watanabe. Neural machine translation with synchronous latent phrase

structure. *ACL-IJCNLP 2021*, page 321, 2021. 58

Ron J. Weiss, J. Chorowski, Navdeep Jaitly, Y. Wu, and Z. Chen. Sequence-to-sequence models can directly translate foreign speech. In *Interspeech 2017, 18st Annual Conference of the International Speech Communication Association*, 2017. 70, 77, 94

Orion Weller, Matthias Sperber, Christian Gollan, and Joris Kluivers. Streaming Models for Joint Speech Recognition and Translation. In *European Chapter of the Association for Computational Linguistic (EACL)*, 2021. 94

Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1137. 67

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 108

Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151:3–34, 2015. 97

Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. A survey on non-autoregressive generation for neural machine translation and beyond. *arXiv preprint arXiv:2204.09269*, 2022. 46

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*, 2016. 15

Brian Yan, Siddharth Dalmia, Yosuke Higuchi, Graham Neubig, Florian Metze, Alan W Black, and Shinji Watanabe. Joint ctc/attention for speech and machine translation. In *Submission to the Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022a. 4

Brian Yan, Patrick Fernandes, Siddharth Dalmia, Jiatong Shi, Yifan Peng, Dan Berrebbi, Xinyi Wang, Graham Neubig, and Shinji Watanabe. CMU's IWSLT 2022 dialect speech translation system. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 298–307, Dublin, Ireland (in-person and online), May 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.iwslt-1.27. URL https://aclanthology.org/2022.iwslt-1.27. 4, 46, 51

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. Tener: adapting transformer encoder for named entity recognition. *arXiv preprint arXiv:1911.04474*, 2019. URL https://arxiv.org/abs/1911.04474. 83

Yilin Yang, Liang Huang, and Mingbo Ma. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium, 10. Association for Computational Linguistics. doi: 10.18653/v1/D18-1342. 79

Kayo Yin, Patrick Fernandes, Danish Pruthi, Aditi Chaudhary, André FT Martins, and Graham Neubig. Do context-aware translation models pay the right attention? 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)*, pages 788–801, 2021. 46

Thomas Zenkel, Ramon Sanabria, Florian Metze, Jan Niehues, Matthias Sperber, Sebastian Stüker, and Alex Waibel. Comparison of decoding strategies for ctc acoustic models. In *Proc. Interspeech 2017*, pages 513–517, 2017. doi: 10.21437/Interspeech.2017-1683. URL http://dx.doi.org/10.21437/Interspeech.2017-1683. 13

Lufeng Zhai, Pascale Fung, Richard Schwartz, Marine Carpuat, and Dekai Wu. Using n-best lists for named entity recognition from chinese speech. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 37–40, 2004. 82

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, and Yang Liu. Neural machine translation with explicit phrase alignment. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1001–1010, 2021. 58

Pei Zhang, Niyu Ge, Boxing Chen, and Kai Fan. Lattice transformer for speech translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6475–6484, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1649. 5, 7

Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7829–7833. IEEE, 2020. 5

Chengqi Zhao, Mingxuan Wang, and Lei Li. NeurST: Neural speech translation toolkit. *arXiv preprint arXiv:2012.10018*, 2020. URL https://arxiv.org/pdf/2012.10018.pdf.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1349–1354, 2019. 108

Renjie Zheng, Junkun Chen, Mingbo Ma, and Liang Huang. Fused acoustic and text encoding for multimodal bilingual pretraining and speech translation. *arXiv preprint arXiv:2102.05766*,

2021. URL https://arxiv.org/pdf/2102.05766.pdf. 7, 77

Liyuan Zhou, Hanna Suominen, and Leif Hanlen. Evaluation data and benchmarks for cascaded speech recognition and entity extraction. In *Proceedings of the Third Edition Workshop on Speech, Language & Audio in Multimedia*, SLAM '15, page 15–18, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337496. doi: 10.1145/2802558.2814646. URL https://doi.org/10.1145/2802558.2814646. 84

Wei Zhou, Zuoyun Zheng, Ralf Schlüter, and Hermann Ney. On language model integration for rnn transducer based speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8407–8411. IEEE, 2022. 59

Job Zwiers. *Compositionality, concurrency, and partial correctness: proof theories for networks of processes, and their relationship*, volume 321. Springer Science & Business Media, 1989. 1