BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

SEMI-SUPERVISED SPEECH-TO-TEXT RECOGNITION WITH TEXT-TO-SPEECH CRITIC

ROZPOZNÁVÁNÍ ŘEČI DO TEXTU S ČÁSTEČNÝM DOHLEDEM A KRITIKEM ZALOŽENÝM NA PŘEVODU Z TEXTU DO ŘEČI

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR **MURALI KARTHICK BASKAR**
AUTOR PRÁCE

SUPERVISOR **doc. Ing. LUKÁŠ BURGET, Ph.D.**
ŠKOLITEL

BRNO 2023

Abstract

Sequence-to-sequence automatic speech recognition (ASR) models require large quantities of training data to attain good performance. For this reason, unsupervised and semi-supervised training in seq2seq models have recently witnessed a surge in interest. This work builds upon recent results showing notable improvements in semi-supervised training using cycle-consistency and related techniques. Such techniques derive training procedures and losses able to leverage unpaired speech and/or text data by combining ASR with text-to-speech (TTS) models.

This thesis first proposes a new semi-supervised modelling framework combining an end-to-end differentiable ASR→TTS loss with TTS→ASR loss. The method is able to leverage unpaired speech and text data to outperform recently proposed related techniques in terms of word error rate (WER). We provide extensive results analysing the impact of data quantity as well as the contribution of speech and text modalities in recovering errors and show consistent gains across WSJ and LibriSpeech corpora.

The thesis also discusses the limitations of the ASR↔TTS model in out-of-domain data conditions. We propose an enhanced ASR↔TTS (EAT) model incorporating two main features: 1) the ASR→TTS pipeline is equipped with a language model reward to penalize the ASR hypotheses before forwarding them to TTS; and 2) speech regularizer trained in unsupervised fashion is introduced in TTS→ASR to correct the synthesized speech before sending it to the ASR model. Training strategies and the effectiveness of the EAT model are explored and compared with augmentation approaches. The results show that EAT reduces the performance gap between supervised and semi-supervised training by absolute WER improvement of 2.6% and 2.7% on LibriSpeech and BABEL respectively.

Abstrakt

Modely pro automatické rozpoznávání řeči (ASR) vyžadují pro dosažení přijatelné přesnosti velké množství trénovacích dat. Z tohoto důvodu se v poslední době zvýšil zájem o trénování seq2seq modelů bez dohledu a s částečným dohledem. Tato práce vychází z nedávných výsledků, které ukázaly výrazné zlepšení trénování s částečným dohledem pomocí cyklické konzistence a souvisejících technik. Ty využívají trénovací postupy a kritéria schopná pomocí kombinace ASR s modely převodu textu na řeč (TTS) zužitkovat nesouvisející řečová a/nebo textová data.

Tato práce nejprve navrhuje nový rámec pro modelování kombinující diferencovatelné end-to-end kritérium ASR→TTS s kritériem TTS→ASR. Tato metoda dokáže využít nesouvisející řečová a textová data a překonat související techniky ve slovní chybovosti (WER). Práce obsahuje rozsáhlou sadu výsledků analyzujících vliv množství dat i vliv podílu řeči a textu na opravách chyb. Výsledky dokládají konzistentní zlepšení na korpusech WSJ a LibriSpeech.

Práce se rovněž zabývá omezeními modelu ASR↔TTS v podmínkách mimo doménu trénovacích dat (out-of-domain). Navrhujeme vylepšený model ASR↔TTS (EAT), zahrnující dva klíčové komponenty: 1) směr ASR→TTS je doplněn jazykovým model, který penalizuje hypotézy ASR před jejich vstupem do TTS; a 2) ve směru TTS→ASR je zavedena regularizace trénovaná bez dohledu tak, aby opravovala syntetizovanou řeč před vstupem do modelu ASR. Zkoumáme strategie trénování a účinnost modelu EAT a porovnáme jej s přístupy umělého zvyšování množství (augmentace) dat. Výsledky ukazují, že model EAT snižuje rozdíl v úspěšnosti mezi trénováním bez dohledu a trénováním s částečným dohledem absolutně o 2,6% WER na LibriSpeech datech a o 2,7% WER na BABEL datech.

Keywords

Automatic speech recognition, text to speech, semi-supervised training, cycle-consistency, unpaired speech and text data, regularization.

Klíčová slova

Automatické rozpoznávání řeči, převod textu na řeč, trénování s částečným dohledem, cyklická konzistence, nesouvisející řeč a textová data, regularizace.

Reference

BASKAR, Murali Karthick. *Semi-Supervised Speech-to-Text Recognition with Text-to-Speech Critic*. Brno, 2023. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. Lukáš Burget, Ph.D.

Rozšířený abstrakt

Modely pro automatické rozpoznávání řeči (ASR) vyžadují pro dosažení přijatelné přesnosti velké množství trénovacích dat. Z tohoto důvodu se v poslední době zvýšil zájem o trénování seq2seq modelů bez dohledu a s částečným dohledem. Tato práce vychází z nedávných výsledků, které ukázaly výrazné zlepšení trénování s částečným dohledem pomocí cyklické konzistence a souvisejících technik. Ty využívají trénovací postupy a kritéria schopná pomocí kombinace ASR s modely převodu textu na řeč (TTS) zužitkovat nesouvisející řečová a/nebo textová data.

Cíle této práce jsou:

1. End-to-end diferencovatelný tréninkový kanál integrací seq2seq ASR a TTS. Architektura je jednoduchá na konstrukci a umožňuje použití stávajícího ASR a modely TTS.

2. Synergujte jak nepárová řečová, tak textová data, abyste snížili chyby ASR zobecněním přes akustické variace i jazykové variace.

3. Pochopte cíl konzistence cyklu pro ASR a navrhněte jeho zlepšení zpracovávat různé datové domény analýzou výkonu na těžších testovacích sadách (např: BABEL-svahilština)

Tato práce nejprve navrhuje nový rámec pro modelování kombinující diferencovatelné end-to-end kritérium ASR→TTS s kritériem TTS→ASR. Tato metoda dokáže využít nesouvisející řečová a textová data a překonat související techniky ve slovní chybovosti (WER). Práce obsahuje rozsáhlou sadu výsledků analyzujících vliv množství dat i vliv podílu řeči a textu na opravách chyb. Výsledky dokládají konzistentní zlepšení na korpusech WSJ a LibriSpeech.

Práce se rovněž zabývá omezeními modelu ASR↔TTS v podmínkách mimo doménu trénovacích dat (out-of-domain). Navrhujeme vylepšený model ASR↔TTS (EAT), zahrnující dva klíčové komponenty: 1) směr ASR→TTS je doplněn jazykovým model, který penalizuje hypotézy ASR před jejich vstupem do TTS; a 2) ve směru TTS→ASR je zavedena regularizace trénovaná bez dohledu tak, aby opravovala syntetizovanou řeč před vstupem do modelu ASR. Zkoumáme strategie trénování a účinnost modelu EAT a porovnáme jej s přístupy umělého zvyšování množství (augmentace) dat. Výsledky ukazují, že model EAT snižuje rozdíl v úspěšnosti mezi trénováním bez dohledu a trénováním s částečným dohledem absolutně o 2,6% WER na LibriSpeech datech a o 2,7% WER na BABEL datech.

Nárůst výkonu v EAT ukázal důležitost společného tréninku ASR s mnoha způsoby, jako je řeč a text. Budou dvě možná rozšíření až 1) mají sdílený kodér pro společné trénování řeči a textu. 2) upsamplujte text sekvence, aby se podobala sekvenci řeči, nebo převzorkování sekvence řeči, aby se podobala textu sekvence.

Semi-Supervised Speech-to-Text Recognition with Text-to-Speech Critic

Declaration

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením doc. Lukáše Burgeta. Další informace mi poskytli Dr. Martin Karafiát and Dr. Shinji Watanabe. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Murali Karthick Baskar
November 10, 2023

Acknowledgements

I would like to express my deepest gratitude to my thesis advisor, Dr. Lukáš Burget, for his unwavering support, guidance, and encouragement throughout the entire research and writing process. His expertise and insight were invaluable to me and his patience and understanding was greatly appreciated.

I would also like to express my sincere respect to my co-advisor Honza Černocký for taking me under his tutelage, for being patient with all my procrastination and has provided constant support during my hard times. He has been a constant motivation and inspiration with his wide knowledge, extraordinary sense of humour, humble nature and he proof-read "the thing" !

I am also grateful to Dr. Martin Karafiát for being a friend and a mentor willing to listen and provide suggestions to my ideas during the course of my doctoral study. He has supported me in my collaborative research with Dr. Shinji Watanabe and Dr. Ramon Astudillo which became the main topic of this thesis.

I would also like to thank the reviewers of my thesis, for their valuable feedback and suggestions, which greatly improved the quality of this thesis, as well as the members of thesis committee.

I extend my appreciation to the FIT BUT, as well as the members of the thesis committee, for providing the resources and support necessary to complete this research.

I am also grateful to my friends Santosh, Mireia, Bhargav, Lucas, Johan, Karel Beneš, Katia, Katka, Bolaji, Honza Švec, Anya and Honza Brukner for their support, which helped me to persevere through the challenging times during the completion of this thesis.

Also, I owe a huge "Thank you!" to Renata Kohlova for being a well wisher and being very helpful in making my stay comfortable in Brno.

Finally, I would like to thank "time" for tailoring my life with benefits to pursue my doctoral studies.

Contents

Notations

Symbol	Description
t	time frame index in $1, ..., T$
l	Token index in text sequence in $1, ..., L$
$\mathbf{x}$	Single data point(vectors)if input features
y	Text sequence as output label
y_{to}	Text only data
$\hat{x}$	Speech sequence predicted by TTS
$\hat{y}$	Text sequence predicted by ASR
X	Matrix of input features with $X = x_1, ..., x_T$
Y	Matrix of output vectors $Y = y_1, ..., y_L$
$E(.)$	Expectation
exp()	Exponential of
$\mathcal{L}$	Loss function
$p(.)$	Probability
A	Alignments using Forward-backward algorithm
a	Attention context vector
H	Encoder hidden state matrix
c	Decoder hidden state vector
q	Quantization vector
α	Scalar weight
λ	Scalar weight
$f(x)$	Abstract encoder
$g(y)$	Abstract decoder
∇_{ASR}	Gradients belonging to ASR model
∇_{TTS}	Gradients belonging to TTS model
Θ	Neural network model parameters
$\mathcal{D}_{to}$	Dataset with text only data
$\mathcal{D}_{so}$	Dataset with speech only data
$\mathcal{D}_s$	Supervised Dataset with speech-text pairs
KL	Kullback-Leibler Divergence
$\arg\max$	Index of maximum element in a vector

Nomenclature

AE	AutoEncoder
AFV	ASR to Feature2Vector
ASR	Automatic Speech Recognition
BLSTM	Bi-directional Long Short-Term Memory
BRMN	Bi-directional Residual Memory Network
CAT	Cluster Adaptive Training
CBHG	1-D convolution bank + highway network + bidirectional GRU
CE	Cross Entropy
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classification
DNN	Deep Neural Network
EAT	Enhanced ASR-TTS
ELBO	Evidence Lower BOund
GCP	Generated Consistent Predictions
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GST	Global Style Tokens
HMM	Hidden Markov Model
ILM	Ideal Localization Mask
IPA	Internation Phonetic Alphabet
IPD	Interchannel Phase Difference
IRM	Ideal Ratio Mask
ISCA	International Speech Communication Association
KL	Kullback–Leibler
LHUC	Linear Hidden Unit Contribution
LLR	Log-Likelihood Ratio
LM	Language Model
LMP	Language Model Penalizer

LPM	Local Prior Matching
LSTM	Long Short-Term Memory
MFCC	Mel-Frequency Cepstral Coefficient
MMSE	Minimum Mean Squared Error
MSE	Mean Square Error
MSE	Mean Squared Error
MTL	Multi Task Learning
MTL	Multi-Task Learning
OOD	Out-of-Domain
RBM	Restricted Boltzmann Machine
REINFORCE	REward INcrement = Nonnegative Factor times Offset Reinforcement times Characteristic Eligibility
ReLU	Rectified Linear Unit
RMN	Residual Memory Network
RNN	Recurrent Neural Network
RNNLM	Recurrent Neural Network Neural Network Language Model
Seq2Seq	Sequence-to-sequence
SO	Speech Only
ST	Speech Only and Text Only
STFT	Short-time Fourier Transform
TO	Text Only
TTE	Text to ASR Encoder outputs
TTS	Text to Speech Synthesis
VAE	Variational Auto Encoder
WER	Word Error Rate
WPE	Weighted Prediction Error
WSJ	Wall Street Journal

Chapter 1

Introduction

1.1 Motivation

Contemporaneous sequence-to-sequence (seq2seq [Bahdanau et al., 2014]) automatic speech recognition (ASR) systems have shown outstanding recognition performance over conventional ASR. The salient and single modeling framework of seq2seq ASR unifies the acoustic model (AM) and language model (LM) components in conventional model architectures [Povey et al., 2011a, Burget et al., 2010, Povey et al., 2011b] into a single and salient learnable model. The seq2seq models contains a relatively large number of model parameters and hence require huge amounts of supervised training data (paired speech-text data). Procuring supervised training data for various domains such as environmental conditions, speaker variations and languages is time consuming and expensive in terms of man hours. On the contrary, unpaired or unsupervised speech and text data are available [Glass, 2012] in abundance. Various successful prior works [Karafiát et al., 2016] on unsupervised learning for ASR have used either unpaired speech [Swietojanski et al., 2014, Schneider et al., 2019] or unpaired text data [Xu et al., 2020a, Hsu et al., 2017] or both [Tjandra et al., 2017] but have not found promising results due to limited modelling capability. *The aspiration of this research is to propose a simplified framework to exploit both unpaired speech and the text data to largely reduce the performance gap between unsupervised and supervised learning compared to recent state-of-the-art (SoTA) models.*

1.2 Related works

The conventional ASR system design has posed major challenges in joint training with unpaired speech and text, resulting in numerous research works involving either unpaired speech or text data. Unpaired speech-based training [Khurana et al., 2021, Xu et al., 2020a] has been applied with different model training criterions [Schneider et al., 2019, Hsu et al., 2017] and has led to improved ASR performance [Oord et al., 2018]. On the contrary, unpaired text has been primarily used to build language models and later integrated with ASR during inference. With the advent of seq2seq ASR systems, ASR training became flexible to text injection. A few works have attempted to inject unpaired text during training by bringing the encoder space [Renduchintala et al., 2018] or decoder space [Hsu et al., 2020a] closer to the text distribution.

More recently, 'machine speech chain' [Tjandra et al., 2020, Tjandra et al., 2017] has been proposed to jointly train unpaired speech and text by cascading seq2seq ASR and

seq2seq text-to-speech (TTS) systems. The intuition behind speech chain is that it learns from ASR and TTS mutually. This line of work is particularly interesting as it yields a simple framework by cascading existing seq2seq ASR and TTS systems.

Irrevocably, the recent success of models holding huge numbers of parameters (in the order of billions) [Zhang et al., 2022, Microsoft,] has heightened the need for unsupervised training regimes. Hence, jointly utilizing unpaired speech and text data presents an incredible opportunity to learn and adapt to different data conditions.

1.3 Contributions and Structure of the Thesis

The primary goal of this dissertation is to *synergise the benefits of seq2seq ASR and seq2seq TTS systems in an end-to-end differentiable pipeline to improve ASR with the unpaired speech and text data.*

To this end, the thesis has three parts:

ASR↔TTS

An ASR↔TTS model is proposed in chapter 3 to jointly train with unpaired speech and text data. This model includes two training pipelines:

1. ASR→TTS, an end-to-end differentiable pipeline to train with unpaired speech data.
2. TTS→ASR pipeline to train with unpaired text data.

In simple terms, ASR↔TTS is a combination of a continuous input/output (speech) based autoencoder [Bengio et al., 2013] and a discrete input/output (text) based autoencoder. ASR↔TTS is motivated by the machine speech chain [Tjandra et al., 2020, Tjandra et al., 2017] model, and the following strategies help to distinguish our work from speech chain:

- Given a speech utterance, ASR generates multiple likely hypotheses and directs them to TTS for reconstruction of speech.
- The ASR→TTS objective applies REINFORCE loss to perform back-propagation continuously from TTS to ASR without hindrance to back-propagation of gradients.

This novel training scheme is beneficial as it considers multiple errors produced by the ASR model and allows the ASR to unsupervisedly learn from the penalty scores provided by the TTS. The following two chapters in this thesis focus on understanding each pipeline independently and addresses its issues.

ASR→TTS with Language Model Prior

The autoencoder (AE) model [Hinton and Salakhutdinov, 2006, Bengio et al., 2013, Zheng et al., 2014] has served as a convenient neural network architecture to learn better representations of data. The variational autoencoder (VAE) [Kingma and Welling, 2013] is a variant of the AE model which has shown great promise in learning better model representations. The design of the VAE is analogous to ASR→TTS, except in the following aspects:

- VAE inherits prior knowledge over the latent variables from standard normal distribution.
- Learning to match the trade-off with prior distribution acts as a regularizer term.

To incorporate these ideas into ASR→TTS, the LM is used as a prior and is integrated with TTS to improve the language related errors. ASR is now jointly trained with TTS and LM and hence named as ASR→TLM and is proposed in chapter 5 as an extension to ASR→TTS. The ASR→TLM learns to match the ASR decoder with the LM prior using cross-entropy objective.

Enhanced ASR↔TTS

The experimental analysis of the TTS→ASR pipeline shows that the synthetic speech generated by the TTS suffers due to domain mismatch. The primary reason for this poor synthesis quality is that TTS can be trained only with clean data.

TTS→AFV (ASR+feat2Vec) architecture is proposed in chapter 6 to address this domain mismatch issue by penalizing the incorrectly synthesized speech using a penalizer before feeding the synthesized speech to the ASR model. AFV refers to a combination of the ASR and feat2vec [Schneider et al., 2019] models. Feat2vec is an unsupervisedly trained speech model used to provide confidence scores for the synthesized speech.

Additionally, to capture better context information, transformer blocks are used in the encoder module of seq2seq ASR and TTS models. The proposed TTS→AFV is later integrated with the ASR→TLM pipeline and the unified model is named as enhanced ASR↔TTS (EAT) and is shown to be capable of handling domain mismatch.

Finally, chapter 6 summarizes the contributions of this thesis and highlights possible directions for future research.

For the sake of reproducibility, the ASR↔TTS model proposed in this thesis is available at `https://github.com/creatorscan/espnet-asrtts`

1.4 Claims of the Thesis

The original contributions of this thesis are as follows:

- Development of an end-to-end trainable integrated ASR and TTS system named ASR↔TTS to handle unpaired speech and text data.
- Extension to ASR↔TTS:
  - LM integration to fix the language related errors by adopting ideas from the variational autoencoder (VAE).
  - A proposal and integration of feat2vec model to reduce the domain mismatch issue in the TTS system.
  - Enhancements with transformer architecture, data annealing and data augmentation strategies.
- Empirical comparison with the state-of-the-art unsupervised learning techniques on the following tasks:
  - Wall Street Journal (WSJ) corpus
  - LibriSpeech corpus
  - BABEL-Swahili
- Analysis of the effect of the proposed model on the scenarios of only unpaired speech, only unpaired text and both unpaired speech and text data.

1.5 Related Publications

Portions of chapters 3, 5 and 6 have appeared in the following articles:

1. Baskar, M.K., Watanabe, S., Astudillo, R., Hori, T., Burget, L., Černocký, J. "Semi-Supervised Sequence-to-Sequence ASR Using Unpaired Speech and Text". In *Proc. Interspeech 2019*, pp. 3790-3794, 2019.

2. Baskar, M.K., Watanabe, S., Astudillo, R., Hori, T., Burget, L., Černocký, Jan. "Eat: Enhanced ASR-TTS for Self-Supervised Speech Recognition". In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6753-6757, 2021.

Chapter 2

Sequence-to-Sequence ASR

Learning to transcribe noisy, unsegmented sequence data is a ubiquitous problem in many real-world sequence learning tasks. ASR is a sequence learning problem $\mathbf{X} \rightarrow \mathbf{Y}$ that can be viewed as $p(\mathbf{Y} \mid \mathbf{X})$, where $\mathbf{Y}$ is the text label sequence and $\mathbf{X}$ is the input sequence of acoustic features. A hybrid approach has been used to handle this problem in ASR; by using hidden Markov model (HMM) [Rabiner and Juang, 1986] to capture the sequential information and neural networks to perform classification of the token labels (context dependent phoneme or grapheme states). While training the HMM segments, the classified labels are transformed to label sequence. However, these hybrid systems assume the independence assumption and fail to exploit the complete potential of neural networks for sequence labelling. Hybrid systems perform classification either by using the Gaussian Mixture Model (GMM) or Deep Neural Network (DNN) models.

Seq2seq ASR systems, on the contrary, simplify the sequence learning problem within a single framework and directly estimate $p(\mathbf{Y} \mid \mathbf{X})$. There are two important seq2seq ASR architectures: 1) connectionist temporal classification (CTC) [Graves et al., 2006] and attention-based encoder-decoder (AED) [Bahdanau et al., 2014].

2.1 From Hybrid to Seq2seq ASR

The conventional DNN-HMM is denoted as hybrid model and the seq2seq model denotes CTC or attention models.

2.1.1 DNN-HMM

The working procedure of ASR is formulated with Bayes decision theory, by predicting the most probable text sequence $\hat{\mathbf{Y}}$ from set of possible sequences $\mathcal{V}$:

$$\hat{\mathbf{Y}} = \arg\max_{\mathbf{Y} \in \mathcal{V}} p(\mathbf{Y}, \mathbf{X}) \tag{2.1}$$

The ASR is trained to learn the posterior distribution $p(\mathbf{Y}, \mathbf{X})$, by introducing the HMM state sequence $\mathbf{S} = \{s_t \in 1, 2, .., J \mid t = 1, 2, .., T\}$ computed for each state j and time t.

and factorizes $p(\mathbf{Y}, \mathbf{X})$ into three components:

$$\begin{aligned} &\arg\max_{\mathbf{Y}\in\mathcal{V}} p(\mathbf{Y}, \mathbf{X}) \\ &\qquad = \arg\max_{\mathbf{Y}\in\mathcal{V}} \sum_{s} p(\mathbf{X} \mid \mathbf{S}, \mathbf{Y})\, p(\mathbf{S} \mid \mathbf{Y})\, p(\mathbf{Y}) \\ &\qquad \approx \arg\max_{\mathbf{Y}\in\mathcal{V}} \sum_{s} p(\mathbf{X} \mid \mathbf{S})\, p(\mathbf{S} \mid \mathbf{Y})\, p(\mathbf{Y}) \end{aligned} \tag{2.2}$$

Here $p(\mathbf{X} \mid \mathbf{S})$, $p(\mathbf{S} \mid \mathbf{S}, \mathbf{Y})$ and $p(\mathbf{Y})$ denotes the acoustic, lexicon and language model (LM) respectively. The conditional independence assumption is followed in acoustic model, lexicon and LM. In acoustic model, the likelihood $p(x_t \mid s_t)$ is replaced with the posterior $p(s_t \mid x_t)$ by pseudo-likehoood trick.

$$\begin{aligned} p(\mathbf{X} \mid \mathbf{S}) &= \prod_{t=1}^{T} p(x_t \mid x_1, .., x_{t-1}, \mathbf{S}) \\ &\approx \prod_{t=1}^{T} \frac{p(s_t \mid x_t)}{p(s_t)} \end{aligned} \tag{2.3}$$

The above assumption limits the context capturing capability, but is mitigated with the help of DNN and Recurrent Neural Network (RNN). In lexicon modeling, $p(\mathbf{S} \mid \mathbf{Y})$ is factorized by chain rule and using first order Markov assumption:

$$\begin{aligned} p(\mathbf{S} \mid \mathbf{Y}) &= \prod_{t=1}^{T} p(s_t \mid s_1, .., s_{t-1}, \mathbf{Y}) \\ &\approx \prod_{t=1}^{T} p(s_t \mid s_{t-1}, \mathbf{Y}) \end{aligned} \tag{2.4}$$

$p(s_t \mid s_{t-1})$ represents the HMM state transition probability and $\mathbf{Y}$ is converted to a sequence of HMM states using the lexicon. The LM here is n-gram based and the probability over the text sequences $p(\mathbf{Y})$ is computed using probabilistic chain rule and conditional independence assumption. RNNLM [Mikolov et al., 2010] mitigate this assumption constraint but make the decoding complex. Combining these three models together leads to incoherent optimization and thus needs a single pipeline model, or so-called seq2seq models: CTC and Attention based encoder-decoder.

The need for seq2seq models rose due to the following problems in hybrid models:

- Multiple modeling steps are required for alignment and training. For instance, Gaussian Mixture Model (GMM)-HMM is required for learning alignments and DNN for label classification.
- Conditional independence assumption is applied to unify three different models namely GMM, DNN and n-gram language model. In addition to this these three models have different training objectives.

2.2 Connectionist Temporal Classification (CTC)

The CTC technique considers the alignment between input and output as latent and thus does not require explicit alignment information as in hybrid systems although the Markov

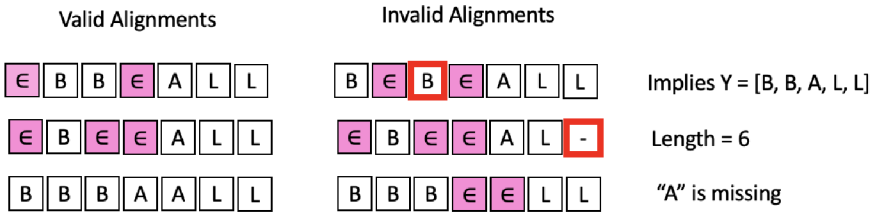


Figure 2.1: Valid and Invalid alignments for an input speech of input length $T = 7$ with the corresponding output $\mathbf{Y}$ with target length $L = 4$

conditions are used to generate the alignments by dynamic programming. The softmax probability scores from the neural network model are used to generate different hard alignment paths; aggregated to obtain soft alignment paths using the Baum-Welch algorithm. CTC still assumes that the output labels are independent of each other while computing hard alignments.

Given an input sequence ($\mathbf{X} = x_1, x_2, ..., x_T$) with T frames, the probability of each output label y_l from sequence $\mathbf{Y} = y_1, y_2, ..., y_L$ with L output labels is computed using the softmax output layer in the CTC network.

Aligning

The alignment between $\mathbf{X}$ and $\mathbf{Y}$ is generated by mapping each of y_l and x_t. Since the number of frames T is always higher than the number of output labels $\mathbf{Y}$, (that is characters or phonemes), two strategies are used during training to handle this mismatch:

- A blank token ϵ is used to mark the silence frames and will be completely ignored at the output
- Repeated tokens are allowed to learn patterns such as '*b a* ***l l***' without collapsing.

Due to the presence of ϵ before or after any output label y_l, a new alignment sequence Z is defined:

$$Z = [\epsilon, B, \epsilon, \epsilon, A, L, L] \tag{2.5}$$

Figure 2.1 shows the process of selecting the valid alignments. Some of the important properties of the CTC alignment process are as follows:

- The alignments Z between input frames and output tokens are monotonic, that is, no future output labels can be aligned to the past input frames.
- Valid alignments are determined when the input and output have the same length sequence. Final alignments are obtained by merging identical subsequent tokens and by removing the blanks, ϵ.
- CTC alignments follow a many-to-one approach, where one or many inputs can be mapped to a single output label.

Finally, $\mathbf{Y}$ is obtained by collapsing the Z sequence to the absolute length of the output sequence L. L is given by the groundtruth - number of characters.

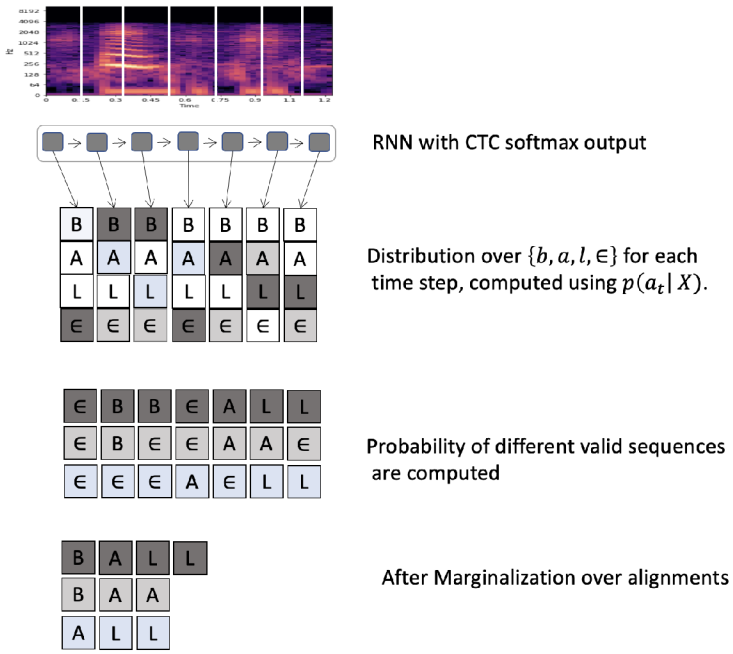


Figure 2.2: Training procedure using CTC objective

Training

The CTC training objective is to maximize the probability of predicting the correct output label given input acoustic frames. The posterior probability $p(\mathbf{Y} \mid \mathbf{X})$ is computed as:

$$p(\mathbf{Y} \mid \mathbf{X}) = \sum_{Z} \prod_{t=1}^{T} p_t(y_t \mid \mathbf{X}) \tag{2.6}$$

Here, $p(\mathbf{Y} \mid \mathbf{X})$ performs marginalization over all valid alignments Z. The alignments are differentiable as in equation 2.6 to allow backpropagation of gradients for weight updation and each alignment is computed step by step as in figure 2.2. CTC architecture is typically constructed using an RNN based encoder which provides the probability of $p(y_t \mid \mathbf{X})$ with the output token, y_t for each time step, t. RNN plays a major role in capturing the long term temporal context information from the input. Calculating CTC loss is computationally expensive as direct computation of scores for each alignment and marginalization is tedious. Hence, dynamic programming is applied by merging the alignments which arrive to the same output at a particular timestep.

Decoding

The CTC model is trained to output a highly probable output sequence $\mathbf{Y}^*$ for each input $\mathbf{X}$ using greedy decoding

$$\mathbf{Y}^* = \arg\max_{\mathbf{Y}} p(\mathbf{Y} \mid \mathbf{X}) \tag{2.7}$$

Given the alignments $[b, b, \epsilon]$ and $[b, b, b]$ as in figure 2.2, each of these has lower probability over alignment $[a, a, a]$, but the sum of their probabilities is higher than that of $[a, a, a]$. In this case, the naive beam search predicts $\mathbf{Y} = [a]$ as the likely prediction. However, the correct output must be $\mathbf{Y} = [b]$. Hence, the naive beam search is not applied in CTC.

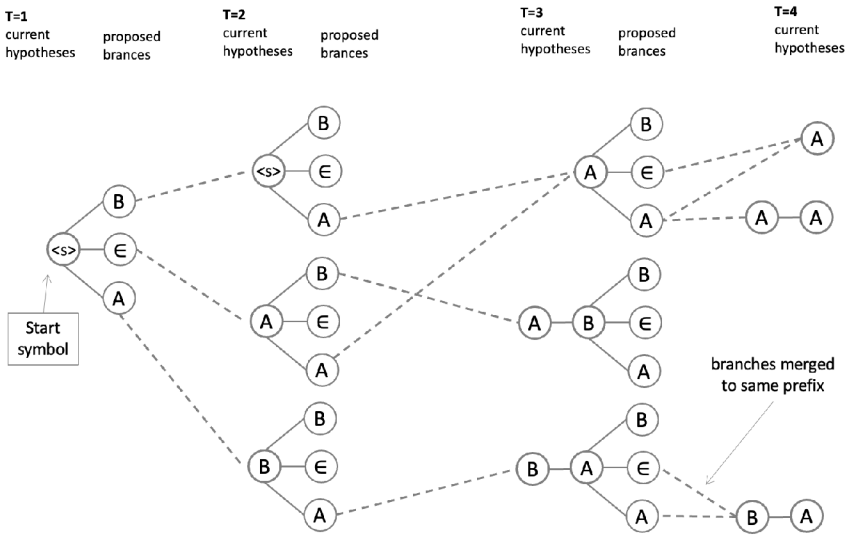


Figure 2.3: CTC prefix beam search algorithm with symbols a, b, ϵ and a beam size of 3

A prefix beam search algorithm can handle this issue by collapsing both the alignments to the same output $\mathbf{Y} = [b]$. Instead of striving to predict the most likely $\mathbf{Y}$, the prefix beam search aims to attain a better solution with minimal computation.

Figure 2.3 shows the prefix beam search decoding procedure. In this, instead of accumulating all the alignments in the beam, only the prefixes are stored after collapsing the repetitive symbols and removing blanks. The prefix scores are accumulated at each step based on the alignments mapped to them. For example, at $T = 3$, the prefix $[a]$ is proposed to have a child branch with a and the possible output prefixes at $T = 4$ are $[a]$ and $[a, a]$. The final prediction is computed by scaling the CTC score $p(\mathbf{Y} \mid \mathbf{X})$ with the language model score $p(\mathbf{Y})$:

$$\mathbf{Y}^* = \arg\max_{\mathbf{Y}} p(\mathbf{Y} \mid \mathbf{X}) \cdot p(\mathbf{Y})^{\alpha} \tag{2.8}$$

2.3 Attention Encoder-Decoder Model (AED)

The encoder-decoder framework [Bahdanau et al., 2014] serves as an alternative to the CTC model and represents a true seq2seq model by performing complex mapping between input speech and output text sequence. The model contains an encoder neural network with recurrent layers, to encode the entire input sequence into an fixed-length vector representation. This vector serves as an input to the decoder – another set of recurrent layers with a final softmax layer, which, in each recurrent iteration, predicts probabilities for the next symbol of the output sequence. This work deals with the task of ASR, where the seq2seq model is used to map a sequence of speech features onto a sequence of characters. In particular, we use an attention-based seq2seq model [Chorowski et al., 2014] in which the encoder encodes an input sequence into continuous low dimensional representation. The attention mechanism focuses on the relevant portion of the internal sequence in order to predict each next output symbol using the decoder. The model is typically trained to maximize the conditional likelihood (or minimize the cross-entropy) of the correct output symbols. For predicting the current character, the previous character (e.g. its one-hot encoding) from the ground truth sequence is typically fed as an auxiliary input to the decoder during train-

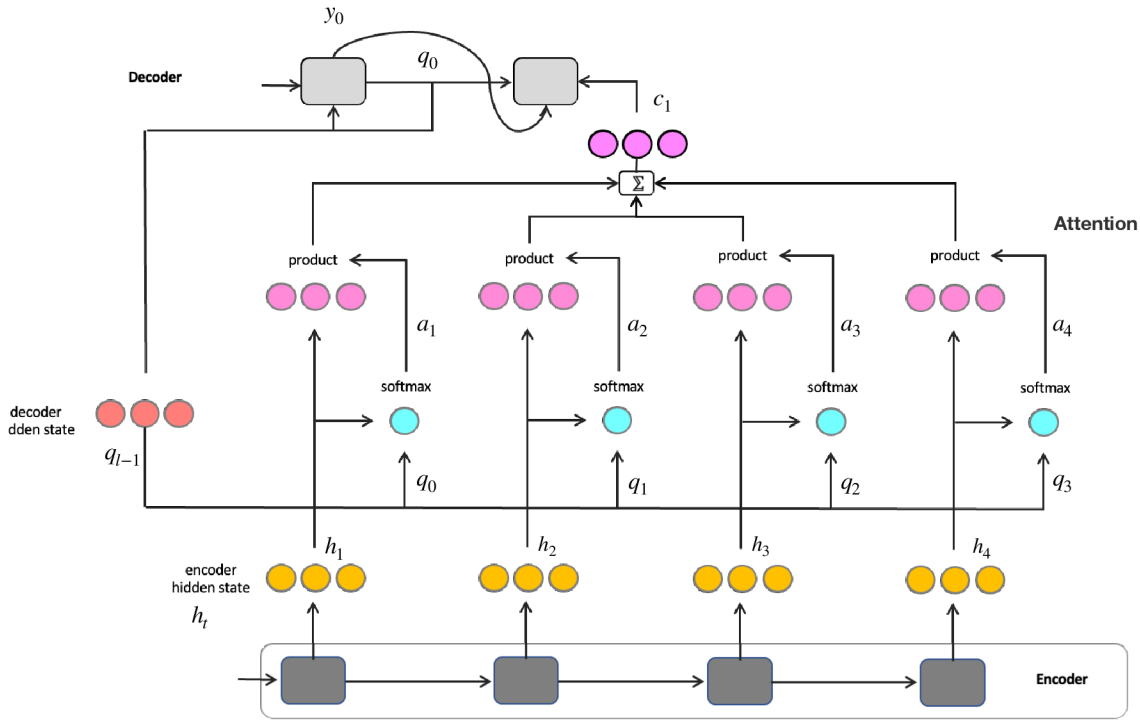


Figure 2.4: Diagrammatic view of attention-based encoder-decoder network (AED).

ing. This so-called teacher-forcing [Williams and Zipser, 1989] helps the decoder to learn an internal language model (LM) for the output sequences in an auto-regressive fashion. Using such a training strategy, the attention [Chorowski et al., 2014] based seq2seq model has been shown to absorb and jointly learn all the components of a traditional ASR (i.e. acoustic, pronunciation and language model).

In detail, the attention-based encoder-decoder (AED) [Chorowski et al., 2014] architecture is shown in figure 2.4. Here, the encoder $\mathbf{H} = \text{enc}(\mathbf{X})$ neural network provides internal representations $\mathbf{H} = \{h_t\}_{t=1}^{T}$ of an input sequence $\mathbf{X} = \{x_t\}_{t=1}^{T}$, where T is the number of frames in an utterance. In this work, the encoder is a recurrent network with bi-directional long short-term memory (BLSTM) layers [Hochreiter and Schmidhuber, 1997, Schuster and Paliwal, 1997]. To predict the l^{th} output symbol, the attention component takes the entire sequence $\mathbf{H}$ and the previous hidden state of the decoder q_{l-1} as inputs and produces per-frame attention weights:

$$\{a_{lt}\}_{t=1}^{T} = \text{Attention}(q_{l-1}, \mathbf{H}). \tag{2.9}$$

Here, q denotes the '*query*' and encoded input $\mathbf{H}$ contains the '*key*' and '*values*' to get the attention alignments. The attention mechanism is location-aware and the values of attention weights for frames are set proportionately to the corresponding focus needed. These frames are used to predict the current output and the weights are normalized to sum-up to one. The weighted average of the internal sequence H with attention weights serves as an attention summary vector, c_l.

$$c_l = \sum_t a_{lt} h_t. \tag{2.10}$$

The decoder is a recurrent network with unidirectional LSTM layers, which receives a_l along with the previously predicted output character y_{l-1} (e.g. its one-hot encoding) as inputs and estimates the hidden state vector

$$q_l = \text{dec}(c_l, q_{l-1}, y_{l-1}). \tag{2.11}$$

This vector is further subjected to an affine transformation (LinB) and softmax non-linearity to obtain the probabilities of the current output symbol y_l

$$s_l = \text{LinB}(q_l) \tag{2.12}$$

$$p(y_l \mid y_{1:l-1}, \mathbf{X}) = \text{Softmax}(s_l) \tag{2.13}$$

The probability of a whole sequence $y = \{y_l\}_{l=1}^{L}$ is

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{l}^{L} p(y_l | y_{1:l-1}, \mathbf{X}) \tag{2.14}$$

During the training, the model parameters are typically updated to minimize the cross-entropy (CE) loss for correct output y^*. This is particularly easy with the teacher forcing, when the symbol from the ground truth sequence is always used:

$$\mathcal{L}_{CE} = -\log p(y^* \mid \mathbf{X}) = -\sum_{l=1} \log\, p(y_l^* \mid y_{1:l-1}^*, \mathbf{X}). \tag{2.15}$$

Here, the previously predicted symbol is from groundtruth and therefore no alternative hypotheses need to be considered. A simple greedy search can be performed to decode the output sequence where the most likely symbol is chosen in each decoding iteration until the dedicated end-of-sentence symbol is decoded:

$$\hat{y}_{1:L} = \arg\max_{\hat{c}_{1:L}} \prod_{l} P(\hat{c}_l \mid \hat{c}_{1,\ldots,l-1,}\mathbf{X}). \tag{2.16}$$

During inference, the predicted characters $\overline{y}_{l-1}$ are fed back unlike training (where ground truth character $\overline{y}_{l-1}^*$ is used). The best path sequence $\overline{y}_{1:L}$ is then estimated from N-best paths (as traversing across all paths is not feasible for practical reasons). This procedure does not guarantee finding the most likely sequence. To find the optimal sequence, exploring multiple hypotheses generated by beam search usually provides better results. Here, each partial hypothesis in the beam search has its own hidden state, as it depends on the previously predicted symbols y_l in that hypothesis.

2.4 Joint CTC-Attention

The joint CTC-attention model [Watanabe et al., 2017a] takes the best out of CTC and AED models through a multi-task learning (MTL) mechanism while training and joint decoding during inference. Joint training with CTC is added to help enforcement of temporally monotonic behaviour in the attention alignments. The overall joint training objective function with MTL is a logarithmic linear combination of the CTC and AED training objectives with $\alpha \in [0, 1]$ as the interpolation weight

$$\mathcal{L}_{\text{MTL}} = \alpha \log p_{\text{CTC}}(\mathbf{Y}|\mathbf{X}) + (1 - \alpha) \log p_{\text{AED}}(\mathbf{Y}|\mathbf{X}), \tag{2.17}$$

Here, the CTC loss $p_{\text{CTC}}(\mathbf{Y}|\mathbf{X})$ is given by (2.6) and the AED loss $p_{\text{AED}}(\mathbf{Y}|\mathbf{X})$ is given by (2.15). During inference, a label-synchronous beam search is employed to predict the most probable label sequence $\hat{\mathbf{Y}}$

$$\hat{\mathbf{Y}} = \arg\max_{\mathbf{Y}\in\mathcal{V}} \{\alpha \log p_{\text{CTC}}(\mathbf{Y} \mid \mathbf{X}) + (1-\alpha) \log p_{\text{AED}}(\mathbf{Y} \mid \mathbf{X}) + \gamma \log p_{\text{LM}}(\mathbf{Y})\}, \tag{2.18}$$

where $\log p_{\text{LM}}(\mathbf{Y})$ is evaluated from an external neural language model with a scaling factor γ. The log probability of the hypothesized character sequence at each partial hypothesis can be calculated using a beam search. The AED partial hypothesis score can be accumulated recursively, while the CTC score can be computed using the CTC prefix probability [Graves et al., 2006, Watanabe et al., 2017b]. The look-ahead word-based LM can be used to obtain the partial hypothesis LM score [Hori et al., 2018].

2.5 Text To Speech (TTS)

In recent years, TTS architectures have undergone a significant change from statistical parametric synthesis (SPS) to the seq2seq approach. Seq2seq TTS simplifies SPS in three ways: 1) the extensively scripted rule based procedure (convert characters to acoustic units) in SPS is handled by a text encoder in seq2seq TTS, 2) SPS individually analyses text and speech, while seq2seq TTS jointly learns from text and speech input 3) Seq2seq TTS is autoregressive in nature (prediction is based on previous context) while SPS predicts speech frames independently. Tacotron is currently the most widely used seq2seq TTS model.

Tacotron and Tacotron2

Tacotron [Wang et al., 2017] is an encoder-decoder based TTS system with the following components an encoder, a decoder, and a post-processing network (postnet). Figure 2.5 shows the model architecture of Tacotron. The main component of Tacotron is the CBHG module. The acronym denotes a 1-D Convolutional layer, a highway network and a bi-directional GRU layer in its architecture. The CBHG is present inside both the encoder and decoder and helps to extract high-level features such as phonetic, prosodic and lexical information from the incoming input sequence. The prenet component inside the encoder applies a set of non-linear transforms over the incoming input text embedding sequences before passing to the bottleneck layer with the dropout component. The generalized output is passed to the CBHG module to reduce overfitting and mispronunciations.

A hyperbolic tangent (tanh) based attention component is present in the decoder with '*query*' being the recurrent layer output at each timestep. Multiple non-overlapping output frames (e.g. three in figure 2.5) are predicted by the decoder.

Tacotron2 [Shen et al., 2018] is a advancement over Tacotron with inclusion of an important component; post-net. The postnet converts the decoder output to a spectrogram and the spectrogram is later sent to a suitable vocoder to generate a speech waveform. The model includes the following components:

- Encoder: The input is a sequence of characters converted to 512-dimensional sequence of embeddings using a learnable embedding matrix. The embedding sequence is fed to a stack of three convolutional layers, a batch normalization layer and ReLU activations. The resulting output is encoded using a 512-dimensional BLSTM layer followed by a location sensitive attention mechanism to capture neighbouring context input.

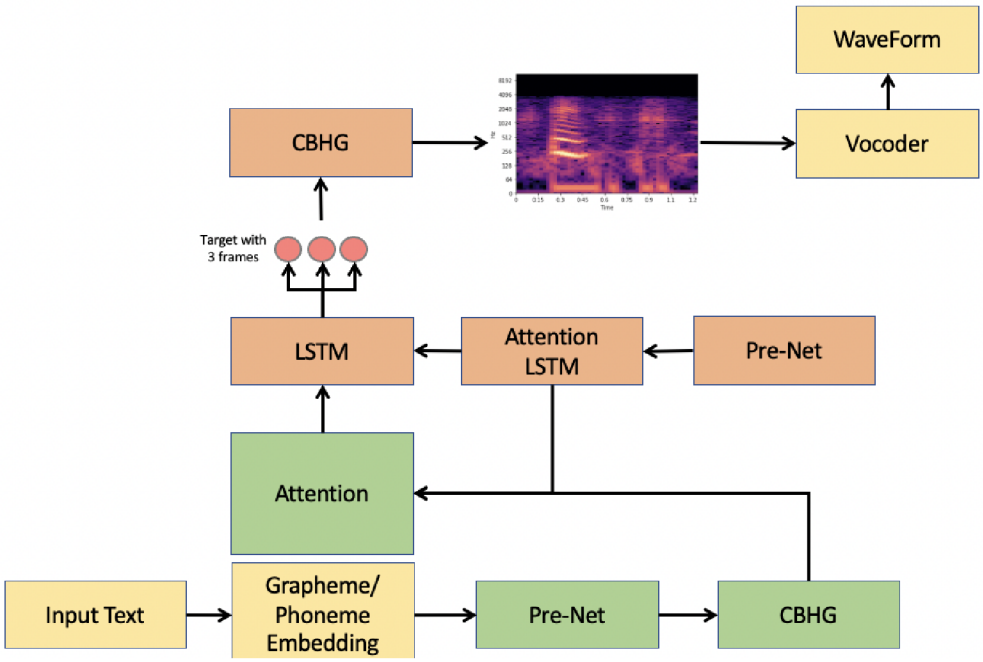


Figure 2.5: Tacotron - A text to speech synthesis model

- Decoder: The module contains prenet (two fully connected layers) and is fed with the encoded outputs. The output of the prenet is concatenated with the encoded attention context vector and directed to two LSTM layers with 1024 neurons. The resulting output is followed by a projection layer to predict the frame-level spectrogram output and the stop token.
- Postnet: The postnet contains five one-dimensional convolutional layers with 512 filters and kernel size of five, followed by the batch normalization and tanh nonlinearity activation. The postnet output acts as a residual and is summed to the prediction to improve the reconstruction output. Finally, the predictions are transformed to waveforms using either Wavenet [Van Den Oord et al., 2016] or Griffin-Lim [Griffin and Lim, 1984].

Figure 2.6 shows the working pipeline of tacotron2 in detail. Initially, the encoder converts the character sequence $\mathbf{Y} = [y_1, y_2, y_3, ..., y_L]$ into encoded outputs $\mathbf{H} = [h_1, h_2, h_3, ..., h_L]$ as:

$$\begin{aligned} \mathbf{H} &= Encoder(\mathbf{Y}) \\ a_{lt} &= Attention(m_{t-1}, \mathbf{H}, a_{l,t-1}) \end{aligned} \tag{2.19}$$

Encoded representations $\mathbf{H}$ are fed to the local-sensitive attention to estimate the attention weights a_{lt}. The attention weights are interpolated with H to get the fixed length context vector c_t for each decoder step. The decoder predicts the intermediate representations $m_1, m_2, ..., m_T$ for each time t auto-regressively from c_t and the previous decoder output x_{t-1}:

$$\begin{aligned} c_t &= \sum_l a_{lt} \cdot h_l \\ m_t &= LSTM(m_{t-1}, c_{t-1}) \end{aligned} \tag{2.20}$$

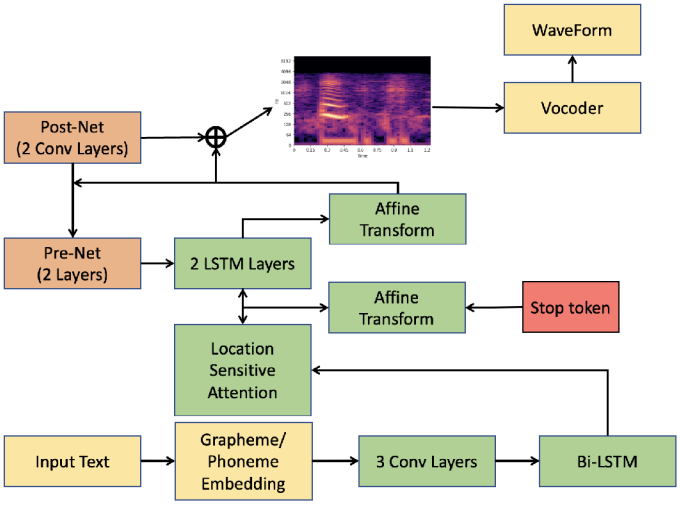


Figure 2.6: Tacotron2

The prediction of the spectrogram at the current time-step x_t is computed by feeding m_t and c_t to a linear layer (FF_{spec}) to get the 80-dimensional output:

$$x_t = FF_{spec}(c_t, m_t) \tag{2.21}$$

Another linear layer, FF_{stop} projects the decoder output m_t and attention output c_t down to one-dimensional output followed by a sigmoid activation to predict the stop token s_t:

$$s_t = \text{sigmoid}(FF_{stop}(c_t, m_t)) \tag{2.22}$$

The stop token reveals that the frame is the last frame of the sequence. This allows the model to be flexible to predict sequences of different lengths.

Multi-Speaker TTS

Multi-speaker TTS complements the Tacotron2 architecture with speaker embeddings. These are generated using a separately trained speaker identification model such as ivector [Doddipatla et al., 2017] or xvector [Hayashi et al., 2021] extractor. The input to the decoder component of Tacotron2 is modified to insert the speaker embedding vector e_s from a particular speaker s. A projection matrix $Proj$ is applied over e_s to match the dimensions of c_{t-1}.

$$\begin{aligned} e_s &= Extractor(\mathbf{X}) \\ m_t &= LSTM(m_{t-1}, c_{t-1} + Proj(e_s)) \end{aligned} \tag{2.23}$$

2.6 Learning from Unpaired Speech

Modeling the acoustics (eg: pitch, intensity, timbre, duration, spectral envelope and formants) in speech signal alone is the most difficult problem as they are non-stationary and sensitive to characteristics of the environment, speaker, channel and so on. The continuous flow of research [Hinton et al., 2012, Gales et al., 2008] in this field has led to progress in making the acoustic modeling practically usable at least for languages with a large amount

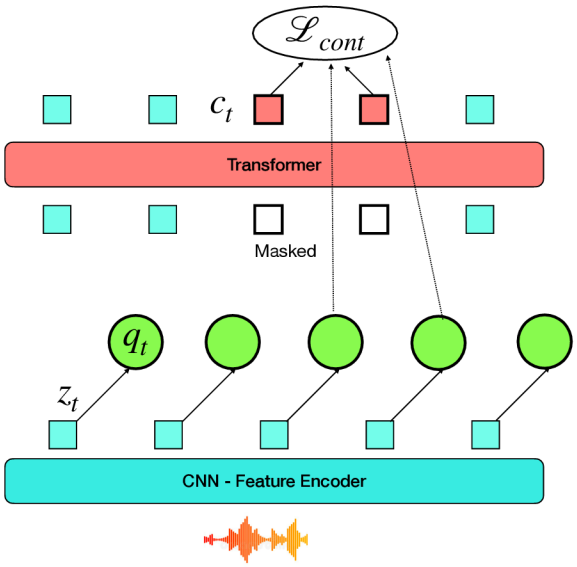


Figure 2.7: Wav2vec2 model architecture to perform self supervised training [Baevski et al., 2020].

of data. However, the field is still evolving to effectively model languages with less supervision and transfer learning has gained attention and success . Some of the major works in speech research deals with transfer across speakers, transfer across languages and transfer across model.[Seide et al., 2011].

2.6.1 Wav2Vec2 Model

The wav2vec2 model is designed to perform self supervised representation learning from unpaired speech data. The model architecture in figure 2.7 contains:

- Feature encoder: Processes the raw waveform input to latent representation z_t using a series of convolutional layers.
- Transformer: Contextual representations c_t are learned from the encoded representations z_t

The model training involves quantization, masking and contrastive learning. Quantization is done to convert the continuous latent representation z_t to quantized (discrete) targets q_t. The quantized targets represents the right code word from the codebook entries. The codebook is created using the Gumbel softmax quantization technique, as it creates diverse codebook entries and has a variable temperature parameter to adjust the diversity. The masking component masks the z_t along the temporal axis and is defined using two hyperparameters: p and M, having the following functionality:

- Randomly select p percentage of starting time indices i from the z_t
- Each index i and its consecutive M time steps are then masked.

Both masked and unmasked inputs are passed through the transformer and the whole architecture is trained to make the resulting output c_t close to the quantized target q_t using

contrastive loss $\mathcal{L}_{cont}$:

$$\mathcal{L}_{cont} = -\log \frac{\exp(sim(c_t, q_t)\ k)}{\sum_{\bar{q}\in\mathcal{Q}} \exp(sim(c_t, \bar{q})/k)}. \tag{2.24}$$

Here, k denotes the temperature parameter, sim is the cosine similarity between c_t and q_t.

2.6.2 Knowledge Distillation

Knowledge distillation was proposed as a theoretical work to perform model compression in [Buciluundefined et al., 2006] and as an application to ASR in [Li et al., 2014, Hinton et al., 2015]. Distillation refers to training a student network under the guidance of a well trained teacher network. The objective of distillation $\mathcal{L}_{\text{KD}}$ is to bring the output distribution of the student model $p_S(y_l \mid x_t)$ close to the output distribution of the teacher model $p_L(y_l \mid x_t)$ using KL divergence (equivalent to cross-entropy).

$$\mathcal{L}_{\text{KD}} = -\sum_{l=1}^{L} p_L(y_l \mid \mathbf{X}) \cdot \log p_S(y_l \mid \mathbf{X}). \tag{2.25}$$

However, the softmax output distribution of the teacher model may have peaky behavior for correct class labels while suppressing the rest of the class labels to zero . To circumvent this and allow rich information beyond ground truth labels, a temperature term τ is introduced to scale the logits before softmax computation:

$$p_L(y_l \mid \mathbf{X}) = \frac{\exp(\frac{z_l}{\tau})}{\sum_k \exp(\frac{z_k}{\tau})}, \tag{2.26}$$

where, z_l are original teacher model posteriors.

2.6.3 Multilingual training

Language evolves from one form to another resulting in different languages across the world and it is natural to believe that they share some common patterns. For example, many consonants and vowels are shared across languages, defined by universal phoneme sets such as the International Phonetic Alphabet (IPA). Even some parts of a word might share the same meaning, for example "nose" in English is similar to "nashi" in Sanskrit, and "inji" in Tamil shares a similar sound with "gingee" (ginger) in English. This sharing between human languages has been utilized explicitly and implicitly to improve statistical strength in multilingual conditions, delivering better models than those trained on monolingual data, especially for low-resource languages. A recent research conducted by Tremblay[1] claim that the basic tendency of humans to learn a new language by associating it with their mother tongue is what is exploited in transfer learning. This advantage has been demonstrated in a multitude of research fields [Thrun, 1995], although our work simply focuses on speech recognition.

A simple approach to sharing data is to define a common phonetic alphabet across all languages. The universal phoneme set is either derived in a data-driven way, or obtained from the IPA by merging the phonemes from different languages. This phoneme set is then used set to obtain multilingual decision trees and tied-state targets for training the

[1]https://www.languagemagazine.com/how-does-mother-tongue-affect-second-language-acquisition/

multilingual neural network [Sim and Li, 2008]. During decoding, language-specific language models and lexicons are used for each language separately. This method is successful compared to mono-lingual systems but is restricted to similar languages.

The multilingual training strategies from conventional hybrid systems have been adopted to seq2seq ASR systems. Multilingual bottleneck features extracted from conventional hybrid models were used to train a seq2seq ASR system in [Cho et al., 2018] and showed substantial improvements. [Karafiát et al., 2016], also applied similar strategy to obtain a multilingual model with additional finetuning to the target language.

2.6.4 Data Augmentation

Data augmentation pivots data sparsity issue from exploiting unsupervised data to manipulating available data with distorted versions. Various data augmentation techniques have demonstrated consistent improvement for ASR [Park et al., 2019, Povey et al., 2011c]. This simple way of obtaining supervised training signal helps us to improve the baseline system, which in turn generates pseudo-labels with higher quality. Speed perturbation, volume perturbation and spectral masking techniques also play a major role in augmentation strategy.

2.6.5 Pseudo-labeling

In the pseudo-labeling technique [Khurana et al., 2021, Xu et al., 2020a], the baseline system is initially trained with the available supervised data. The baseline model is used to predict labels on the unlabelled data. The confidence predictions (pseudo-labels) are chosen assuming that they are correct and are augmented with the supervised data during training. If the noise in pseudo-labels is sufficiently low, the ASR model can benefit from this additional training data to obtain improved accuracy. [Xu et al., 2020a], repeat the pseudo-label generation and the augmented training steps and obtain continuous improvements in both.

2.7 Learning from Unpaired Text

The above discussed solutions efficiently use unpaired speech data to deal with conditions with small amount of paired data. In hybrid ASR systems application of unpaired text data is done by modeling it separately using a language model. The language model (LM) is integrated with the ASR only during inference. This technique does not utilize the unpaired text data during training due to the complex nature of ASR architecture. On the other hand Seq2seq ASR allows training with unpaired text data in multiple ways. Local prior matching [Hsu et al., 2020a] jointly trains LM with seq2seq ASR by reducing the distance between predictions of ASR and LM outputs using KL divergence. A multimodal objective used in [Renduchintala et al., 2018] extends the shared encoder concept by upsampling the input text sequence to match the corresponding speech sequence.

Another extreme condition not widely focused on in the literature is training an ASR jointly with unpaired speech and text data. In this approach, the ASR is taught to unify both speech and text within a common subspace. The shared encoder approach analysed in [Karita et al., 2018b] aligns and maps the text representations and speech representations into a joint space to impose textual knowledge in the encoder. An extension to the shared encoder is carried out using an adversarial training objective in [Drexler and Glass, 2018]

to improve the ASR performance. Although these techniques did not improve over the existing semi-supervised learning approaches, they created a major impact in proving the potential of using unpaired speech and text data.

2.8 Directions of this Thesis

In this thesis, the AED based seq2seq ASR approach is used as it has the advantage over hybrid ASR to easily integrate unpaired speech and text data. A closer look at the works discussed above shows that:

- the major effort is towards designing a better model architecture, and
- the model abstraction (objective/strategy) is fuelled by the autoencoder architecture

In our work, the autoencoder architecture is simplified by connecting the seq2seq ASR and TTS models. This approach directs the focus to proposing a better training objective. This idea is also explored in [Tjandra et al., 2017] but without using an efficient training objective. The experimental results in our work show that the proposed work provides better performance over the existing best performing models. Chapter 3 describes the consistency algorithm followed by the introduction of the proposed ASR$\leftrightarrow$TTS architecture to train with unpaired speech and text data. Extensive analysis is conducted using the WSJ corpus to showcase the contribution of training with speech and text independently. The final system is tested with LibriSpeech and compared with the related works. The performance with LibriSpeech is further improved in chapter 5 by improving the training using LM predictions. Finally, the thesis explores the effect of the proposed model under realistic low-resource data conditions using BABEL-Swahili in chapter 6.

Chapter 3

ASR↔TTS: Cycle Consistency

Consistency is all I ask

–Tom Stoppard

Requiring parallel data for training a seq2seq ASR system is extortionate. The virtually unlimited presence of speech only (SO) or text only (TO) data should be leveraged to reduce recognition errors. This can be done by obtaining pseudo-supervised labels from pre-trained ASR and TTS models. While pre-trained TTS helps to synthesize speech from TO data, the pre-trained ASR can be leveraged to predict pseudo-labels from SO data [Xu et al., 2020a, Synnaeve et al., 2019, Wang et al., 2007]. While this allows obtaining pseudo-supervision, it still does not completely exploit the pre-trained models to aid the ASR performance.

To address this issue, a "speech chain" approach was used in [Tjandra et al., 2017] to jointly train the ASR with the TTS model in a self-supervised fashion. A sequence of ASR and a text-to-encoder (TTE) model [Hori et al., 2019] is analogous to speech chain, and both works showed that connecting ASR with TTS/TTE can handle unpaired data to reduce ASR recognition errors. This thesis borrows ideas from the above mentioned techniques to further improve the ASR by efficiently exploiting the TTS to handle unpaired data using consistency training. In our work, the TTS acts as a teacher in guiding the ASR to learn from its incorrect predictions.

This chapter describes a self-supervised ASR training paradigm named "ASR-TTS" that exploits the unpaired speech or text data to improve speech recognition performance. Its design is based on integrating ASR and TTS modules into a single architecture and performing end-to-end differentiable training. The model involves two training pipelines, the ASR→TTS and the TTS→ASR, for a given an input of speech, x_{so} and text, y_{to} from the datasets $\mathcal{D}_{so}$ of SO and $\mathcal{D}_{to}$ of TO, respectively.

- ASR→TTS: The SO data samples are exploited using the ASR→TTS pipeline as in figure 3.1. Here, the x_{so} is first fed into ASR and the predicted text sequence y is sent to TTS to generate speech $\hat{x}_{so}$.
- TTS→ASR: The TO data samples are exploited using the TTS→ASR as shown in figure 3.2. This is simply a reversed version of ASR→TTS where an input text sequence y_{to} is sent to TTS to generate speech $\hat{x}$ which is further processed with ASR to predict text sequence $\hat{y}_{to}$.

Both pipelines are trained separately using a consistency training algorithm [Baskar et al., 2019]. To jointly train both pipelines, a cycle-consistency training algorithm is applied. The proposed ASR-TTS approach is evaluated using the standard LibriSpeech corpus [Panayotov et al., 2015] by varying the amount of supervision and unsupervision. The efficacy of the model is substantiated by comparing it with existing unsupervised techniques.

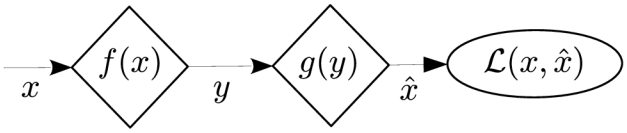


Figure 3.1: Consistency training procedure for training with input speech only data x

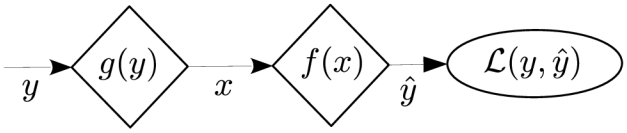


Figure 3.2: Consistency training procedure for training with input text only data y

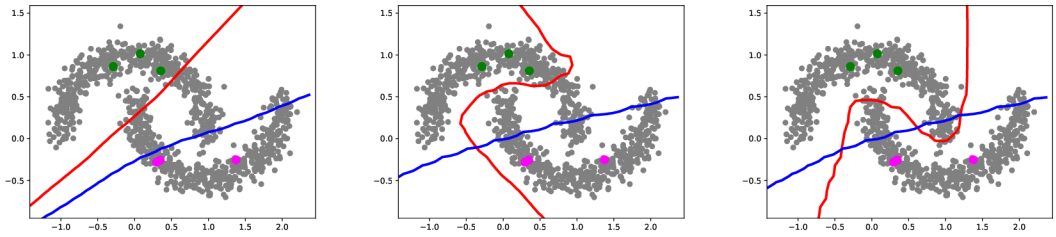
3.1 Consistency training

Consistency training aims at learning from unknown things by linking them to things which are known. Its objective provides a meta-supervision that does not operate on the data directly, but rather focuses on how the data must behave. This objective has been applied to a wide range of applications requiring unsupervised learning such as text classification [Xie et al., 2019] and image recognition [Verma et al., 2019, Tarvainen et al., 1780]. The generalized consistency training objective is computed by performing the following steps as in figure 3.1 and 3.2:

- Given an input x, a function $f(x)$ is used to generate the output y. Another function $g(y)$ intakes y to predict $\hat{x}$. This is shown in figure 3.1.
- Consistency loss strives to minimize the divergence between two distributions $p(x)$ and $p(\hat{x})$ which gradually allows the function $f(x)$ to learn and correct from its own imperfections.
- While the above process helps to model the unpaired input x, the reverse process (figure 3.2) shows the procedure to train with unpaired input y.

The output labels $\hat{x}$ and $\hat{y}$ used in this training procedure are the same as the inputs or are derived from the inputs as in figures 3.1 and 3.2 respectively. This falls under the category of self-supervised learning.

Figure 3.3, shows the importance of training with consistency loss using a toy dataset named moons. The task is to discriminate the two classes using some/lesser supervised data and lots of unsupervised data. Here, modelling the discriminator (simple feedforward network) with less supervised data (blue line) fails to discriminate the two classes while training with unsupervised data using a consistency objective gradually learns a non-linear hyper-plane (red-line) to discriminate the classes.



(a) (b) (c)

Figure 3.3: Decision boundaries with supervised (blue line) and consistency (red line) training with increasing number of training iterations. Plot (a) shows that during the initial training phase have an no decision boundary to classify the two classes., (b) intermediate training phase and (c) final phase with consistency objective correctly separating the data.

3.2 Supervised seq2seq modeling

ASR: The conventional supervised seq2seq ASR model computes a cross-entropy loss, $\mathcal{L}_{\text{ASR}}$, by using input speech (the Mel filterbank features) x, and predicts the text token sequence, y in an auto-regressive fashion:

$$p(\hat{y} \mid x) = \prod_{l=1}^{L} p(\hat{y}_l \mid \hat{y}_{1:l-1}, x). \tag{3.1}$$

The loss is thus given as:

$$\mathcal{L}_{\text{ASR}} = -\log p(\hat{y} \mid x) = -\sum_{l=1}^{L} p(\hat{y}_l \mid \hat{y}_{1:l-1}, x). \tag{3.2}$$

TTS: Tacotron2 described in section 2.5 is used as the TTS model in this work. Tacotron2 receives the character sequence $y = \{y_l\}_{l=1}^{L}$ as input, predicts the speech (filterbank) features $x = \{x_t\}_{t=1}^{T}$ with a regression layer and uses the corresponding supervised training objective, $p(\hat{x} \mid y)$ The loss of the TTS system is composed of three different loss terms:

$$\mathcal{L}_{\text{TTS}} = \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{L}_1} + \mathcal{L}_{\text{BCE}}, \tag{3.3}$$

where the mean square error $\mathcal{L}_{\text{MSE}}$ ensures the predicted Mel spectrogram estimates $\hat{x}$ are closer to the genuine Mel spectrogram x. $\mathcal{L}_{\text{L}_1}$ is also used as an auxiliary loss to match the actual regression estimate x to the groundtruth and acts robust to outliers [Pesme and Flammarion, 2020] compare to $\mathcal{L}_{\text{MSE}}$. $\mathcal{L}_{\text{BCE}}$ is the binary cross entropy loss for the end-of-sentence prediction. The MSE and L_1 components of the loss can be interpreted as negative log-probabilities of the speech features for Gaussian and Laplace distributions for constant scale parameters, i.e.

$$\mathcal{L}_{\text{TTS}} = -\log p(\hat{x} \mid y) = -\sum_{t=1}^{T} \log p(\hat{x}_t \mid \hat{x}_{1:t-1}, y).$$

Considering the MSE only,

$$-\log p(\hat{x}_t \mid \hat{x}_{1:t-1}, Y) \propto \|\hat{x}_t - \mathbf{g}(y, \hat{x}_{1:t-1})\|^2 \tag{3.4}$$

where $\hat{x}_t - \mathbf{g}(y, \hat{x}_{1:t-1})$ is the TTS model's auto-regressive estimate at time t.

3.3 ASR→TTS: Speech only (SO) data training

The functions $f(x)$ and $g(y)$ in general figure 3.1 are replaced by ASR and TTS models as in figure 3.4a and the final consistency objectives are adapted to suit the ASR training with speech only data. The resulting model is denoted as ASR→TTS pipeline, where the filterbank features x_{so} from Speech Only (SO) data are converted to text y using ASR and this text output is converted back to filterbank features $\hat{x}_{so}$ using TTS.

3.3.1 Jointly training ASR and TTS

Jointly training ASR and TTS with TTS objectives has the following issues and the ASR→TTS makes two major contributions:

Issue 1: The central problem in designing this pipeline is that the text bottleneck eliminates much information from speech, for example speaker identity. This makes the TTS generate input features common to all speakers.

Solution 1: To mitigate this issue, the TTS receives text y along with speaker dependent vectors e_s in ASR→TTS as shown in figure 3.4a. This allows the TTS to generate speaker dependent filterbank features. Aside from being computationally more intensive, this requires solving the problem of passing speaker characteristics along with the the text (see figure 3.4a). In order to do so, inspired by [Tjandra et al., 2017], the TTS model is augmented with speaker vectors obtained from an x-vector network [Snyder et al., 2018]. In this way the MSE criterion of $\mathcal{L}_{\text{TTS}}$ (3.4) used here is:

$$\mathcal{L}_{\text{MSE}} = -\log p(\hat{x} \mid \hat{y}, e_s) \tag{3.5}$$

where $e_s = \mathbf{f}(x^s)$ is the x-vector for speaker s implemented using function $\mathbf{f}(.)$. Note that x-vectors are designed to retain speaker characteristics but not the general structure of the speech signal. In that sense, the model can not learn to copy x directly from input to output.

Issue 2: Jointly training ASR and TTS as in figure 3.4a is also constrained due to another issue: end-to-end differentiability. The text sequence predicted by ASR is discrete in nature and this hinders the backpropagation of gradients from TTS to ASR as shown in figure 3.4b. These gradients carry essential information about the score of the hypotheses and can guide the ASR to learn from its erroneous predictions.

Solution 2: The consistency training objective in the ASR→TTS pipeline incorporate an expectation likelihood function to impose differentiability in the pipeline. The resulting ASR→TTS objective is defined as:

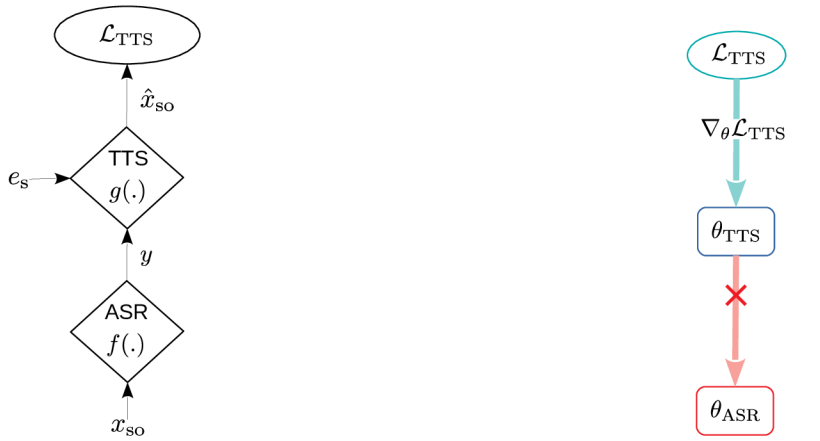
$$\mathcal{L}_{\text{ASR}\rightarrow\text{TTS}} = \mathbb{E}_{p(\hat{y}|x_{so})}\{\mathcal{L}_{\text{TTS}}\} \tag{3.6}$$

Computing an expectation over all possible hypotheses $\hat{y}$ is intractable and hence an approximation is made over a finite set of N hypotheses $\hat{y}_1, \hat{y}_2, .., \hat{y}_N$:

$$\mathcal{L}_{\text{ASR}\rightarrow\text{TTS}} \approx \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}_{\text{TTS}} \cdot p(\hat{y} \mid x_{so}) \tag{3.7}$$

The approximation is derived using a score function estimator named REINFORCE [Ranzato et al., 2015]. The intuition behind this joint training with REINFORCE is two-fold:

- the gradients can be propagated from TTS to ASR through the discrete text tokens without hindrance as in figure 3.6b.
- this enables the discrete the gradients back-propagated from the TTS to suppress the distribution of negative samples while simultaneously boosting the distribution of positive samples.



(a) Forward flow from ASR to TTS

(b) Back-propagation of gradients

Figure 3.4: Simplified representation of the forward and backward propagation during consistency training for unpaired speech data. In figure (b), the red arrow denotes that the gradient cannot be propagated and the blue arrow denotes that gradient propagation is possible.

The forward and backward propagation procedure using the REINFORCE objective for SO training is designed as shown in figures 3.6a and 3.6b respectively. These figures show a modification to allow consistency training with the TTS objective.

3.3.2 REINFORCE - Score function gradient estimator

The score function estimator called REINFORCE, helps to define gradients for the non-differentiable parts of the training pipeline. Figure 3.5 details the working procedure of REINFORCE by moving the positive samples from the distribution to improve the scores. For instance, when the TTS predicts a score (MSE from the predicted spectra) for a single ASR hypothesis, it is actually allowed to predict a probability distribution over TTS scores by predicting multiple ASR hypotheses. This is similar to the expected loss function where instead of computing loss for one hypothesis, it is computed for multiple hypotheses. In figure 3.6a, the ASR is allowed to predict multiple hypotheses y_n where $n = 1, 2, 3, 4$:

$$y = \text{REAL} : \text{ ground-truth sequence}$$
$$\hat{y}^1 = \text{REAL}, \hat{y}^2 = \text{REAR}, \hat{y}^3 = \text{REAL}, \hat{y}^4 = \text{ROARER} : \text{ ASR predictions}$$

Here, the TTS score is computed using the mean squared error $\mathcal{L}_{\text{TTS}} = \|\hat{x}^n - x\|^2$ between the predicted input feature $\hat{x}_{so}$ and the groundtruth x_{so}. The score errors are low for

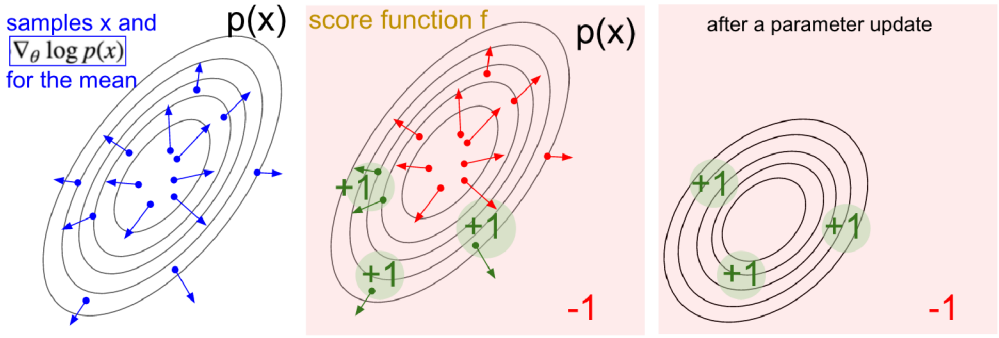


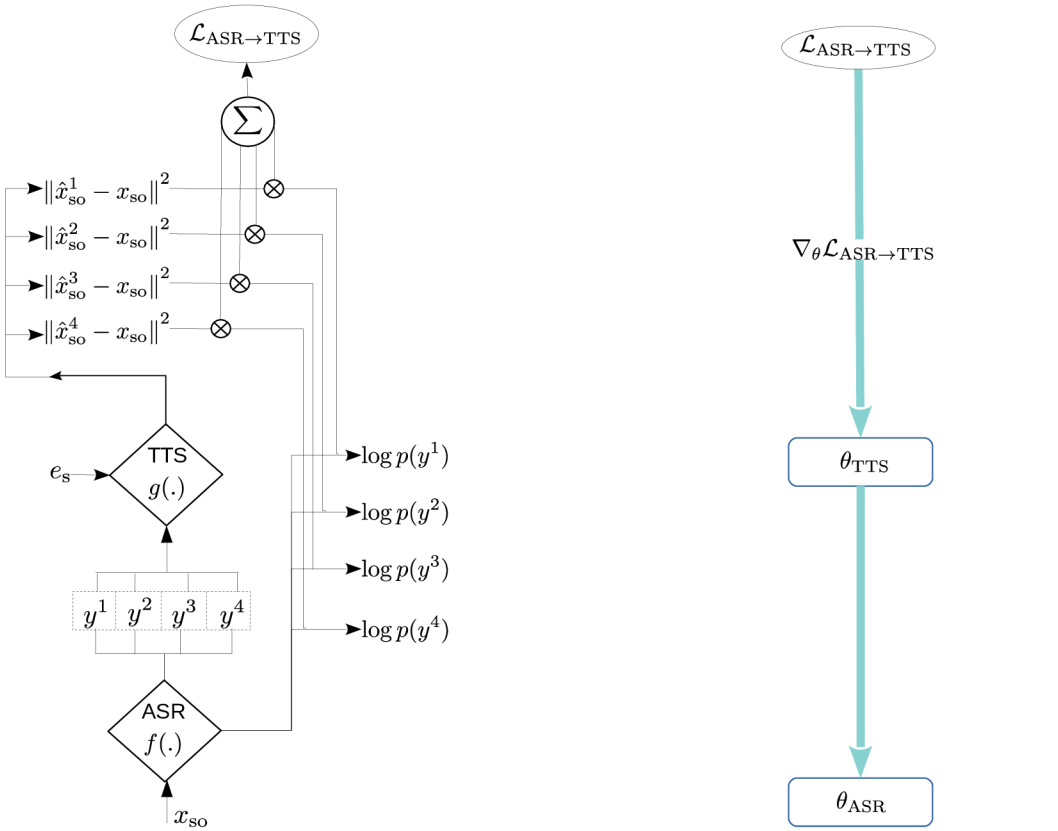
Figure 3.5: Graphical view of the REINFORCE - score function gradient estimator. Left: Gaussian distribution $p(x)$ and a few samples from it (blue dots). Plot of the gradient of the log probability $\log p(x)$ with respect to the Gaussian's mean parameter. The blue arrows indicate the directions in which the mean of the distribution should be moved to increase the probability of that sample. Middle: Score function f giving -1 everywhere except +1 in some small regions. Right: after parameter update, the green arrows and the reversed red arrows move towards left and towards the bottom. Samples from this distribution will now have a higher expected score. In this work, $p(x) = p(\hat{y}|x)$ and $\theta = \theta_{\text{ASR}}$. The figure is adopted from Karpathy's blog on reinforcement learning using REINFORCE algorithm and policy gradients.[1]

correct predictions $\hat{y}^1$ and $\hat{y}^3$ and high for incorrect predictions $\hat{y}^2$ and $\hat{y}^4$. Based on the TTS score allotted for each hypothesis $\hat{y}^n$, the hypotheses '$\hat{y}_2$ = REAR' and '$\hat{y}_4$ = ROARER' acts as negative samples, making hypotheses '$\hat{y}_1$ = REAL' and '$\hat{y}_3$ = REAL' with high TTS scores more likely. Conventional back-propagation is then performed to update the ASR model parameters based on this loss. The gradients of the positive samples are weighted higher while the negative samples are provided lower weights. The REINFORCE based objective $\mathcal{L}_{\text{ASR}\rightarrow\text{TTS}}$ allows the ASR model to consider more variance in hypotheses (random sampling) to provide more correct predictions and fewer errors.

$$\mathcal{L}_{\text{ASR}\rightarrow\text{TTS}} = \mathbb{E}_{p(\hat{y}|x_{so})}\{\mathcal{L}_{\text{TTS}}\} \tag{3.8}$$

For SO training, the ASR parameters θ_{ASR} only are updated and TTS parameters θ_{TTS} are fixed. The following equations provides the derivation for gradient computation for $\mathcal{L}_{\text{ASR}\rightarrow\text{TTS}}$:

$$\begin{aligned}
\nabla_\theta \mathcal{L}_{\text{ASR}\rightarrow\text{TTS}} = \nabla_{\theta_{\text{ASR}}} \mathbb{E}_{p(\hat{y}|x_{so})}\{\mathcal{L}_{\text{TTS}}\} &= \frac{1}{N}\sum_{n=1}^{N} \nabla_{\theta_{\text{ASR}}}\, p(\hat{y} \mid x_{so}) \cdot \mathcal{L}_{\text{TTS}} \\
&= \frac{1}{N}\sum_{n=1}^{N} \nabla_{\theta_{\text{ASR}}}\, p(\hat{y} \mid x_{so}) \cdot \mathcal{L}_{\text{TTS}} \\
&= \frac{1}{N}\sum_{n=1}^{N} p(\hat{y} \mid x_{so}) \frac{\nabla_{\theta_{\text{ASR}}}\, p(\hat{y} \mid x_{so})}{p(\hat{y} \mid x_{so})} \cdot \mathcal{L}_{\text{TTS}} \\
&= \frac{1}{N}\sum_{n=1}^{N} p(\hat{y} \mid x_{so}) \nabla_{\theta_{\text{ASR}}} \log p(\hat{y} \mid x_{so}) \cdot \mathcal{L}_{\text{TTS}} \\
&= \mathbb{E}_{p(\hat{y}|x_{so})}\{\mathcal{L}_{\text{TTS}}\, \nabla_{\theta_{\text{ASR}}} \log\, p(\hat{y} \mid x_{so})\}
\end{aligned}$$



(a) Forward flow from ASR to TTS

(b) Back-propagation of gradients

Figure 3.6: Simplified representation of the forward and backward propagation during consistency training for unpaired speech data with REINFORCE. In figure (b), the gradients smoothly travel from TTS to ASR.

To reduce the variance in the gradient computation using REINFORCE, a bias term $\mathbf{B}(x)$ [Ranzato et al., 2015] is introduced to control overall additive shift in loss and perform reduction control over the variance:

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\text{ASR}\to\text{TTS}} &= \mathbb{E}_{p(\hat{y}|x_{so})}\{(\mathcal{L}_{\text{TTS}} - \mathbf{B}(x_{so}))\, \nabla_{\theta_{\text{ASR}}}\, p(\hat{y} \mid x_{so})\} \\ &= \mathbb{E}_{p(\hat{y}|x_{so})}\{\mathbf{R}(\hat{y}, x_{so})\, \nabla_{\theta_{\text{ASR}}}\, p(\hat{y} \mid x_{so})\}. \end{aligned} \tag{3.9}$$

The bias term $\mathbf{B}(x)$ is calculated as the mean value of x_{so} for each sample. The resulting expectation is exponentially large on sentence length and to simplify the expectation over all possible hypotheses, an approximation over a limited set of hypotheses is assumed:

$$\nabla_\theta \mathcal{L}_{\text{ASR}\to\text{TTS}} \approx \frac{1}{N} \sum_{n=1}^{N} \mathbf{R}(\hat{y}, x_{so})\, \nabla_{\theta_{\text{ASR}}} \log\, p(\hat{y}^n \mid x_{so}) \tag{3.10}$$

In practical terms, using REINFORCE amounts to sampling multiple sentences from the ASR distribution and backpropagating each of them as if they were the ground truth, but weighted by the MSE reconstruction loss.

Algorithm 1 Consistency algorithm with REINFORCE

Require: Speech input $x_{so} \in \mathcal{D}_{\text{so}}$ from speech only (SO) dataset $\mathcal{D}_{\text{so}}$, paired speech utterance with text $(x_s, y_s) \in \mathcal{D}_{\text{s}}$ from supervised dataset $\mathcal{D}_{\text{s}}$, pre-trained ASR θ_{ASR} and TTS θ_{TTS} models, hyper-parameter α, number of samples N, learning rate γ_t

repeat

$x_{\text{so}} \in \mathcal{D}_{\text{so}}$. // Unsupervised training: Sample a speech sequence

$\hat{y}^1_{so}, \hat{y}^2_{so}, ..., \hat{y}^N_{so} \leftarrow p(.|x_{so}, \theta_{\text{ASR}})$ // Generate N text sequences according to ASR

for $n = 1, 2, .., N$ **do**

$\hat{x}^n_{\text{so}} \leftarrow p(.|\hat{y}^n_{so}, \theta_{\text{TTS}})$

$r^n = \mathcal{L}_{\text{TTS}}(x_{\text{so}}, \hat{x}_{\text{so}}) - \mathbf{B}(x_{so})$ // Reward for the n^{th} sequence

end for

$\mathcal{L}_{\text{so}} \leftarrow -\frac{1}{K} \sum_{n=1}^{N} r^n \cdot \log\ p_(\hat{y}_n|x_{so})$

$x_{\text{s}}, y_{\text{s}} \in \mathcal{D}_{\text{s}}$ // Supervised training: Sample a speech with corresponding text

$\hat{y} \leftarrow p(.|x_s, \theta_{\text{ASR}})$

$\mathcal{L}_{\text{s}} \leftarrow -y_s \cdot \log p(\hat{y}|x_s)$

$\mathcal{L} \leftarrow \mathcal{L}_s + \alpha \cdot \mathcal{L}_{so}$ // Final objective

if update TTS **then**

$\theta_{TTS} \leftarrow \theta_{TTS} + \gamma \cdot \nabla_{\theta_{TTS}} \mathcal{L}_{so}$ // TTS update

end if

$\theta_{ASR} \leftarrow \theta_{ASR} + \gamma \cdot \nabla_{\theta_{ASR}} \mathcal{L}$ // ASR update

until convergence

3.4 TTS→ASR: Text only (TO) data training

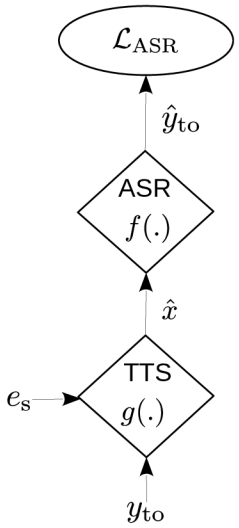
The consistency training pipeline used for TO training is obtained by reversing the SO training pipeline in figure 3.4a. Figure 3.7a shows the training pipeline where a given unpaired text or text only (TO) data y_{to} is sent as input to the TTS model to generate filterbank features $\hat{x}_{to}$. The TTS model is speaker-dependant as it uses an auxiliary input: x-vector e_s where the speaker identity s is randomly sampled for each utterance. The resulting point estimate output for an utterance obtained from TTS is:

$$\hat{x}_{to} = \arg\max_{c} \{p(x_c \mid y_{to}, e_s)\}. \tag{3.11}$$

The generated filterbank features are then sent to ASR to predict text sequence $\hat{y}_{to}$. The final objective is a conventional cross-entropy loss:

$$\mathcal{L}_{\text{TTS}\rightarrow\text{ASR}} = -\log p_{\text{ASR}}(\hat{y}_{to} \mid \hat{x_{to}}). \tag{3.12}$$

Backpropagation of gradients through TTS→ASR as shown in figure 3.7b is simpler than in the ASR→TTS case; the gradients can traverse from TTS to ASR without any hindrance as the output of TTS is continuous output and hence the pipeline is end-to-end differentiable. The resulting computation graph is simpler than in the ASR→TTS case. This pipeline acts as an autoencoder and allows simple training of unpaired text only data. The procedure to train the TTS→ASR pipeline is summarized in algorithm 2.



(a) Forward flow from TTS to ASR

(b) Back-propagation of gradients

Figure 3.7: Simplified representation of the forward and backward propagation during consistency training for unpaired text data. In figure (b), the blue arrows denote the free flow of gradients from ASR To TTS

Algorithm 2 Consistency algorithm: Training with text only (TO) data

Require: Text input $x_{to} \in \mathcal{D}_{\text{to}}$ from TO dataset $\mathcal{D}_{\text{to}}$, Parallel speech utterance with text $(x_s, y_s) \in \mathcal{D}_{\text{s}}$ from supervised dataset $\mathcal{D}_{\text{s}}$, pre-trained ASR θ_{ASR} and TTS θ_{TTS} models, hyper-parameter α, learning rate γ_t

repeat

- $y_{\text{to}} \in \mathcal{D}_{\text{to}}$. // Sample a text sequence
- $x_{\text{to}} \leftarrow \text{TTS}(x_{\text{to}})$
- $\hat{y}_{\text{to}} = \text{ASR}(x_{to})$
- $\mathcal{L}_{\text{to}} \leftarrow -y_{to} \cdot \log\, p_(\hat{y}_{to}|\hat{x}_{to})$
- $x_{\text{s}}, y_{\text{s}} \in \mathcal{D}_{\text{s}}$ // Sample speech with corresponding text
- $y_{\text{s}} \leftarrow \text{ASR}(x_{\text{s}})$
- $\mathcal{L}_{\text{s}} \leftarrow -y_s \cdot \log\, p(\hat{y}|x_s)$
- $\mathcal{L} \leftarrow \mathcal{L}_s + \alpha \cdot \mathcal{L}_{to}$ // Final objective
- **if** update TTS **then**
  - $\theta_{TTS} \leftarrow \theta_{TTS} + \gamma \cdot \nabla_{\theta_{TTS}} \mathcal{L}_{to}$ // TTS update
- **end if**
- $\theta_{ASR} \leftarrow \theta_{ASR} + \gamma \cdot \nabla_{\theta_{ASR}} \mathcal{L}$ // ASR update

until convergence

3.5 ASR↔TTS: SO and TO data training

Cycle-consistency training is a dual consistency training from both $f(x)$ to $g(y)$ and from $g(y)$ to $f(x)$. This procedure does not require joint representation learning to learn the correspondence between two domains such as speech and text. Instead, it cycles between

two or more input samples to validate matching patterns. The flowchart in figure 3.8 presents the working procedure of both SO and TO training using the ASR↔TTS model.

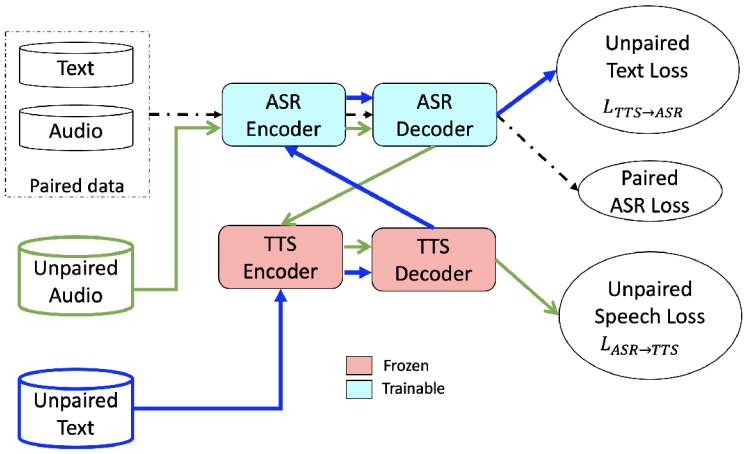


Figure 3.8: Flowchart of ASR↔TTS model

In case both unpaired speech and text are available, both ASR→TTS and TTS→ASR can be trained jointly as in algorithm 3. The final loss $\mathcal{L}_{\text{both}}$ is a linear interpolation of the loss functions defined in equations (3.8) and (3.12):

$$\mathcal{L}_{\text{both}} = \alpha \cdot \mathcal{L}_{\text{ASR}\to\text{TTS}} + (1-\alpha) \cdot \mathcal{L}_{\text{TTS}\to\text{ASR}} \tag{3.13}$$

where α is a hyper-parameter set by default to $\alpha = 0.5$. The importance of using the cycle-consistency objective in this work is as follows:

- The encoder component of the ASR model can learn the acoustic information from the SO data training using the ASR→TTS pipeline.
- The decoder component of the ASR model which acts as an intrinsic language model will be better optimized from the TO data training using the TTS→ASR pipeline.

Thus, cycling between ASR→TTS and TTS→ASR acts complementary to each other and helps to improve ASR performance.

3.5.1 Why joint training with a differentiable model ?

A natural way to exploit SO and TO data is by using ASR and TTS in a disjoint setting. In this case, the ASR is used to predict pseudo-labels from SO data, and TTS is used to generate pseudo-synthesized speech from TO data. The predicted text requires post-processing to remove the errors [Xu et al., 2020a] before it can be used as pseudo-supervision to train a new ASR. The synthesized speech can be directly used as additional pseudo-supervised data along with supervised data to train an ASR system. While this method does not require tedious post-processing as ASR is generating the pseudo text labels, it still requires a well-trained TTS system with the ability to generate natural sounding speech. Table 3.1 shows the performance of text and speech based pseudo-labels using the WSJ corpus.

Algorithm 3 Cycle consistency algorithm: SO and TO data training

Require: Speech input $x_{so} \in \mathcal{D}_{\text{so}}$ and text input $x_{to} \in \mathcal{D}_{\text{to}}$ from SO dataset and TO dataset $\mathcal{D}_{\text{so}}$ and $\mathcal{D}_{\text{to}}$ respectively, Paired speech utterance with text $(x_s, y_s) \in \mathcal{D}_{\text{s}}$ from supervised dataset $\mathcal{D}_{\text{s}}$, pre-trained ASR θ_{ASR} and TTS θ_{TTS} models, hyper-parameter α, β, learning rate γ_t

repeat

$\mathcal{L}_{\text{so}} \leftarrow -\frac{1}{K}\sum_{k=1}^{K} \log p_(y_k|x_{so}) \cdot r_k$ // Compute consistency loss using algorithm1

$\mathcal{L}_{\text{s}} \leftarrow -y_S \cdot \log p(\hat{y}|x_s)$ // Compute consistency loss using algorithm2

$\mathcal{L} \leftarrow \mathcal{L}_s + \alpha \cdot \mathcal{L}_{so} + \beta \cdot \mathcal{L}_{to}$ // cycle consistency objective

if update TTS **then**

$\theta_{TTS} \leftarrow \theta_{TTS} + \gamma \cdot \nabla_{\theta_{TTS}} \mathcal{L}_{so}$ // TTS update

end if

$\theta_{ASR} \leftarrow \theta_{ASR} + \gamma \cdot \nabla_{\theta_{ASR}} \mathcal{L}$ // ASR update

until convergence

Model	%WER
Baseline	31.1
Pseudo-speech	30.1
Pseudo-text	29.2
Nondifferentiable-ASR→TTS	28.0
Differentiable-ASR→TTS	26.8

Table 3.1: Recognition performance in %WER of different model training types on the eval-92 test set using WSJ-si284 (67 hours) as unpaired speech and text data. WSJ-si84 (14 hours) is used as supervision. The pseudo-supervised models are built using data augmentation (supervised data + pseudo speech/text). The jointly trained ASR→TTS directly uses the SO data in addition to supervised data.

Furthermore, jointly training ASR and TTS is still possible without being end-to-end differentiable and it can be done by having a weighted sum of ASR and TTS objectives:

$$\mathcal{L} = \mathcal{L}_{\text{ASR}}(\hat{y}, y^{'}) + \mathcal{L}_{\text{TTS}}(\hat{x}_{so}, x_{so}). \tag{3.14}$$

Here, the character token sequence $y^{'}_{so}$ is obtained by picking the top-1 character based on the probability $p(\hat{y}_{so}|x_{so})$:

$$y^{'}_{so} = \prod_{l=1}^{L} \arg\max_{j} \log p(\hat{y}_{lj}|x_{so}) \tag{3.15}$$

The results of disjoint ASR→TTS with consistency training are shown in table 3.1. The empirical study shows that similarly to pseudo-labelling, our approach improves over the baseline. However, training ASR and TTS jointly improves over pseudo-label models. The end-to-end differentiable ASR→TTS performs better to non-differentiable ASR→TTS by absolute 1.2% WER.

3.6 Comparison to existing works

Sequence-to-sequence (seq2seq) ASR training directly maps a speech input to an output character sequence using a neural network [Graves and Jaitly, 2014, Bahdanau et al., 2016, Chan et al., 2016], similar to those used in machine translation [Bahdanau et al., 2014, Sutskever et al., 2014]. The model requires a considerable amount of paired speech and text to learn alignment and classification [Amodei et al., 2016, Prabhavalkar et al., 2017], which limits its use in under-resourced domains. On the other hand, unpaired speech and text can be obtained for most domains in large quantities making training with unpaired data particularly relevant for seq2seq models.

Recent works have shown that the problem of seq2seq training in low-resource conditions can be addressed using unpaired data. These methods can be classified into three categories according to the type of data used. First are methods utilizing only unpaired speech for unsupervised training. In this category, [Tjandra et al., 2018] proposes an end-to-end differentiable loss integrating ASR and TTS models by the straight-through estimator. The work in [Hori et al., 2019] also proposes an end-to-end differentiable loss integrating ASR and a text-to-encoder (TTE) model. Both works show that connecting ASR with TTS/TTE can process unpaired speech data as well as reduce ASR recognition errors.

3.6.1 Unpaired Speech training with Text-to-Encoder (TTE)

Figure 3.9 shows the working model of the TTE using the consistency training objective.

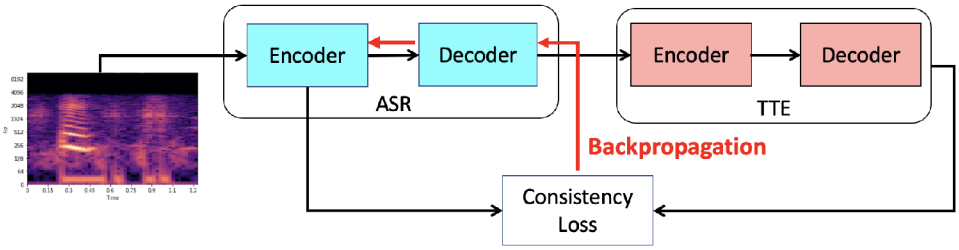


Figure 3.9: Self-supervised learning between ASR encoder outputs and its predictions using TTE.

Here, the unpaired speech is used for encoder-state-level self supervised training as in wav2vec2 described in chapter 2. The encoder state sequences of the ASR model are generated and used to compute the self supervised loss. Note here that the encoder states are predicted by the TTE model instead of waveform or spectral features as in TTS. This approach reduces the mismatch between the original and the reconstruction by avoiding the problem of missing para-linguistic information in an end-to-end differentiable manner. The self-supervised loss is computed based on expected loss approximated with sampling-based method. The sampling is done to obtain multiple sentences from the ASR model to generate encoder state sequences. Finally, the average loss of all the sequences is used to backpropagate the error to the ASR model via the REINFORCE algorithm as in our approach (see section 3.3.2). This allows to update the ASR system when the TTE is used to compute the loss.

The TTE model can be simply viewed as a TTS model trained to predict the encoded speech, $\mathbf{H}$ in equation (6.3), instead of speech $\mathbf{X}$ directly. Since the encoded speech holds

some speech characteristics, it is possible to define the following loss based on the consistency criterion. Similarly to [Hori et al., 2019], we use policy-gradient to back-propagate through the expectation in our unpaired speech loss (section 3.3) and update the ASR parameters:

$$\mathcal{L}_{\text{ASR}\rightarrow\text{TTE}} = E_{p_{\text{ASR}}(\mathbf{Y}|\mathbf{X})}\{\mathcal{L}_{\text{TTE}}\}, \tag{3.16}$$

where $\mathcal{L}_{\text{TTE}}$ is the same as $\mathcal{L}_{\text{TTS}}$, but $\mathbf{X}$ is replaced by the encoded speech $\mathbf{H}$:

$$\mathcal{L}_{\text{TTE}} = \text{MSE}(\hat{h}_t, h_t) + \mathcal{L}_{L1}(\hat{h}_t, h_t) E_{p_{\text{ASR}}(\mathbf{Y}|\mathbf{X})}\{\mathcal{L}_{\text{TTE}}\}. \tag{3.17}$$

This loss penalizes the ASR system for transcriptions that, once transformed back into an estimate of the encoded speech $\hat{\mathbf{H}}$ by the TTE, differ from the original encoded speech $\mathbf{H}$. Such loss does not require having the correct text output y^* and is end-to-end differentiable. TTE has the disadvantage of having to train a specific network to predict $\mathbf{H}$. The encoded speech may also already eliminate some of the speech characteristics making the CC loss less powerful. Henceforth, in our work, we to use the loss between filterbank features instead of the encoded representations.

Table 3.2 compares the performance of the TTS loss introduced in our work, to the TTE loss in [Hori et al., 2019]. The analysis involved a LibriSpeech setup using 100 hours of paired data with 180 and 360 hours of unpaired data for a fair comparison. The results shows that the TTS approach improves %WER over TTE by 2.5% relative (20.7% to 20.1%) on 360 hours of unpaired data and 2.9% relative (19.9% to 19.4%) on the 180 hours set.

Table 3.2: Comparison between TTE [Hori et al., 2019] and TTS using the LibriSpeech corpus on test-clean set

Unpaired speech (# Hours)	Model	%CER	%WER
180	TTE	8.8	20.7
180	TTS	8.7	20.1
360	TTE	8.6	19.9
360	TTS	8.4	19.4

3.6.2 Unpaired Text Training

Backtranslation-style TTE→ASR

[Hayashi et al., 2018] uses this idea of TTE for unpaired text training. Here, the backtranslation technique from machine translation [Sennrich et al., 2015] is adopted into attention based E2E-ASR model to exploit large amounts of unpaired text data. Figure 3.10 shows the self-supervised training of ASR with TTE using unpaired text data. In the TTE→ASR approach, the ASR is pre-trained with paired training data, and the final layer of the ASR encoder is used to extract hidden activation outputs. These are used as labels for training TTE using paired data. The unpaired text data is passed through the pre-trained TTE to extract hidden activations which are fed into the ASR decoder. Retraining both the ASR encoder and decoder resulted in better performance compared to only retraining the ASR decoder. The TTS→ASR used in our work (section 3.4) simplifies this pipeline in two ways:

- Backtranslation style TTE→ASR is complex and does not use filterbank features rich in speech characteristics.

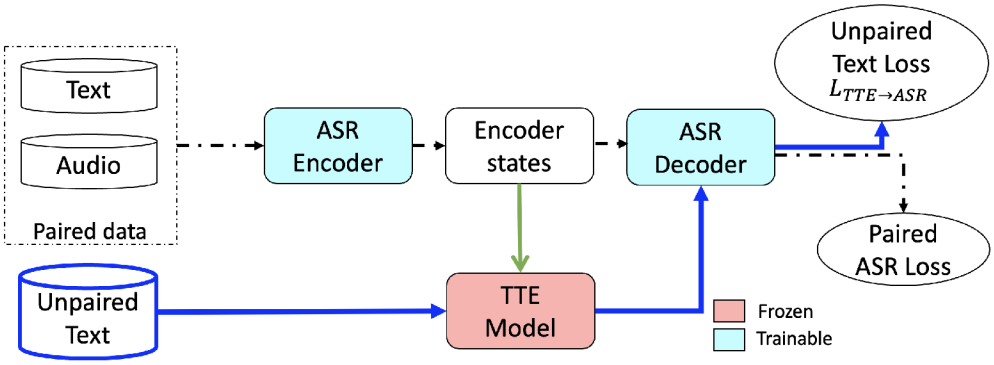


Figure 3.10: Overview of backtranslation style unpaired text data training

- Paired data is mandatory in this approach and must be used in a multi-task manner to recover the connection between ASR encoder and decoder broken by the TTE encoder outputs.

In addition to this work, other notable works have used unpaired text data in ways different from proposed work. For instance, [Renduchintala et al., 2018] attempts to upsample the text sequence similar to speech and feeds it as additional input along with the speech sequence to a shared encoder. The model learns to bring the encoder representation closer to the text representation. In addition to these works, [Liu et al., 2019] used a language model (LM) built with unpaired text-only data to jointly train with an ASR.

Criticizing LM

The criticizing LM is trained along with ASR as an adversary, where the ASR acts as generator and the criticizing LM as discriminator. The discriminator is trained to distinguish real text from ASR transcriptions. Here, the criticizing LM intakes unpaired text or ASR transcriptions as input and outputs the scalar quality score s . The model is trained to assign higher scores to real text and lower scores to ASR transcriptions.

The Wasserstein generative adversarial network (WGAN) training objective is applied to reduce the mismatch between real and hypotheses text distribution.

3.6.3 Unpaired Speech and Text

Machine Speech Chain

In the third category, both unpaired speech and text data are exploited. Machine speech chain [Tjandra et al., 2017] is the first approach to use a single pipeline to train using both unpaired speech x and text y data. Figure 3.11 shows two training pipelines that cascade seq2seq ASR and seq2seq TTS models in different ways. The supervised data is used to train the ASR and TTS independently via teacher forcing. The unpaired speech features are passed into a frozen ASR model to generate the text predictions which are redirected to a TTS model to synthesize speech features $\hat{x}$. The unpaired text input is fed to the frozen TTS model to predict the speech features and then fed to the ASR model to obtain text predictions $\hat{y}$. Here, both the TTS and the ASR model parameters are updated. The idea behind training these two pipelines jointly is that ASR and TTS and can mutually teach other with the inclusion of reconstruction loss to the unlabelled data. The training

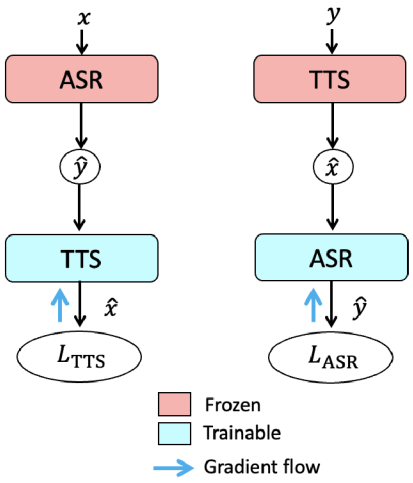


Figure 3.11: Overview of machine speech chain architecture. ASR to TTS pipeline for unpaired speech training. TTS to ASR pipeline for unpaired text training.

objective of the speech chain model is:

$$\mathcal{L}_{\text{Chain}} = \mathcal{L}^{ut}_{\text{ASR}} + \mathcal{L}^{us}_{\text{TTS}} + \mathcal{L}^{p}_{\text{ASR}} + \mathcal{L}^{p}_{\text{TTS}}, \tag{3.18}$$

Here $\mathcal{L}^{ut}_{\text{ASR}}$ denotes the ASR loss during training with unpaired text data and TTS loss with unpaired speech training is denoted by $\mathcal{L}^{us}_{\text{TTS}}$. Our proposed ASR↔TTS differs from the machine speech chain in the following ways:

- We treat the TTS model as the scorer and evaluate the ASR hypothesis based on these scores. Our focus is to improve ASR and hence the TTS model is not updated reducing the training complexity.
- The REINFORCE training objective is applied, which intakes several ASR hypotheses for loss computation and check variations in predictions to learn effectively.
- The ASR↔TTS training pipeline is end-to-end differentiable with the help of REINFORCE and allows backpropagation of gradients from TTS to the ASR model.

To overcome the constraints of speech chain, the authors in [Tjandra et al., 2020] introduced Gumbel-softmax loss to train unpaired speech data. Gumbel-softmax is applied over the ASR outputs to approximate categorical distribution with continuous distribution and the gradients are computed using the reparameterization trick. However, the unsupervised data and supervised data used in these works are obtained from the same WSJ corpus. Moreover, our experiments showed that REINFORCE generalizes well to small domain variations due to sampling multiple hypotheses from ASR compared to the Gumbel-softmax approach.

Maximum mean Discrepency (MMD)

In [Karita et al., 2018b], it is proposed to reduce the dissimilarity between encoded speech and text using the maximum mean discrepancy (MMD) algorithm. Speech and text are encoded using encoders in ASR and TTS models. The speech encoder and text decoder in ASR are shared with the TTS model, and both are jointly trained using unpaired speech

and text. In comparison to our work, MMD is inferior in performance as learning the matching distribution between speech and text is difficult.

A few other notable works [Drexler and Glass, 2018, Karita et al., 2018b] use adversarial training to bring the speech and text encoded representations closer. In [Drexler and Glass, 2018], speech encoder and text encoder outputs are adversarially matched to feed the text decoder. The speech representations are brought closer to text encodings using the discriminator. The authors integrate the denoising autoencoder objective for both text and speech with adversarial loss between speech and text encoder.

Position of our work

Compared to the above existing works, our work proposes improvements over recent related approaches and integrates some of them into a single loss. In addition to this, the cycle-consistency objective is successful outside of the speech processing area, and has sparkled significant progress in other fields such as machine translation [He et al., 2016], which derive end-to-end differentiability by using the N-best approach. Moreover, improvements are observed in image processing tasks using cycle-consistency with an adversarial objective [Zhu et al., 2017].

Chapter 4

Cycle Consistency Experiments

4.1 Database selection

The proposed ASR↔TTS (see chapter 3) framework requires a special way of data selection to handle unsupervised and supervised experiments. The LibriSpeech [Panayotov et al., 2015] and Wall Street Journal (WSJ) [Paul and Baker, 1992] corpora were used in this work. Different subsets of data from WSJ and LibriSpeech are used during training and the evaluation data is set as eval-92 for WSJ and test-clean and test-other for the LibriSpeech corpus.

4.1.1 Training - Paired datasets

The supervised data comes from the following sources:

- WSJ-si84 (14h)
- WSJ-si284 (67h)
- Complete WSJ corpus (84h)
- For analysis on how much paired data is required, the WSJ corpus is split into 2, 5 and 10 hours subsets (2h, 5h, 10h).
- LibriSpeech data (100h)
- LJSpeech - single speaker dataset for TTS (24h)

4.1.2 Training - Unpaired datasets

The unsupervised scenario is extensively analysed using the various sets and data sizes:

- WSJ-si284 (67h)
- Complete WSJ corpus (84h)
- 360 hours of LibriSpeech data
- 500 hours of LibriSpeech data
- 860 hours of LibriSpeech data

Table 4.1: List of characters used in all our experiments

Character	ID
<unk>	1
!	2
„	3
&	4
’	5
(	6
)	7
	8
,	9
-	10
.	11
/	12
:	13
;	14
<NOISE>	15
<space>	16
?	17
A	18
B	19
C	20
D	21
E	22
F	23
G	24
H	25
I	26
J	27
K	28
L	29
M	30
N	31
O	32
P	33
Q	34
R	35
S	36
T	37
U	38
V	39
W	40
X	41
Y	42
Z	43
{	44
}	45

The data selection is carried out meticulously such that there is no overlap between supervised and unsupervised examples.

4.1.3 Testing - Evaluation datasets

The models are evaluated using both WSJ and Librispeech test sets.

- WSJ-eval 92 (1.2h)
- Test-clean from Librispeech (5h)
- Test-other from Librispeech (5.3h)

4.2 Feature extraction

83-dimensional features containing 80 dimensional Mel filter-bank energies appended with 3-dimensional pitch features are extracted from speech waveforms for both WSJ and LibriSpeech. x-vectors [Snyder et al., 2018] are extracted from a TDNN based model pretrained with around 2000 hours of Voxceleb1 and Voxceleb2 data. These x-vectors act as auxiliary information to provide the speaker characteristics of each utterance while training the TTS system.

4.3 ASR and TTS architectures

The encoder-decoder network described in chapter 2 utilizes location aware attention [Bahdanau et al., 2016]. Tacotron architecture [Shen et al., 2018] is used to build a TTS model. A complete description of ASR and TTS models is presented in tables 4.2 and 4.3. The learning rate decay is based on the validation performance computed using the character error rate (minimum. edit distance). ESPnet [Watanabe et al., 2018] is used to implement and execute all the experiments.

4.4 Language Models

45 characters in table 4.1 along with <s> as start of symbol and </s> end of symbol are chosen as text tokens for both the WSJ and LibriSpeech datasets. Two different types of language models are built to decode the ASR models which are:

- The training data containing 1662964 lines of text with a 65001 vocabulary size is used to build a word based language model [Hori et al., 2017].
- 116500 lines of training data are obtained from the 360 hours LibriSpeech subset to build a character level language model.

4.5 Training and Decoding

A multi-task objective [Watanabe et al., 2017b] is employed during training and decoding by providing equal weightage of $\alpha = 0.5$ to CTC and attention (see section 2.4 and eqn. 2.15). Shallow fusion [Cho et al., 2018] is used to integrate the RNNLM during decoding. The scaling factors for language model probabilities are empirically chosen based on the model

Table 4.2: ASR Model configuration

Encoder	
Type	Bi-LSTM [Hochreiter and Schmidhuber, 1997]
# Layers	8
# Dimensions	320
# Projection dim	320
Decoder	
Type	LSTM
# Layers	1
# Dimensions	320
Attention	
Type	Location based
Conv. channels	10
Conv. filters	100
# Dimension	320
Window size	5
Subsampling factor	4
Training Configuration	
Optimizer	AdaDelta [Zeiler, 2012a]
# Epochs	20
Batch size	14

performance. Unsupervised training was performed after conventional supervised training of each model. Cycle-consistency utilized five samples in (3.7). A small amount of paired data was also used to regularize the model during the unsupervised stage. The eval92 test set was kept for evaluations. In LibriSpeech, the 100 hour set is used as paired data and the 360 hours set as unpaired data as in [Hori et al., 2019].

4.5.1 Pre-trained models

The ASR and TTS models are initially pre-trained with pre-defined amount of supervised data from WSJ (14h) and LibriSpeech (100h). The proposed ASR↔TTS training starts with the following pre-trained models:

- **pre-ASR-14h**: ASR supervisedly trained with 14 hours WSJ-si84 corpus
- **pre-ASR-100h**: ASR supervisedly trained with 100h of Librispeech
- **pre-TTS-100h**: TTS is first trained with the LJSpeech [Ito and Johnson, 2017] (see section 4.1). This model is re-trained with the multi-speaker corpus from LibriSpeech (100h). Directly training with 100h did not help as TTS requires a lot of supervised data.

Table 4.3: TTS Model configuration

Encoder	
Type	Bi-LSTM
# Layers	1
# Dimensions	512
# Embedding dim	512
Conv. filters	5
Decoder	
Type	LSTM
# Layers	2
# Dimensions	1024
Pre-net layers	2
Prenet units	256
Postnet layers	5
Postnet channels	512
Postnet filters	5
Attention	
Type	Location based
Conv. channels	32
Conv. filters	15
# Dimension	128
# Heads	4
Window size	5
Training Configuration	
Optimizer	Adadelta
# epochs	200
batch size	32
subsample	4
learning rate	1e-3
dropout rate	0.5
zoneout	0.1
eps (Denominator value in optimizer)	1e-6

4.5.2 Baseline experiments

In this section, the ASR models are prepared with different amounts of supervised data. The model configuration mentioned in table 4.2 is used to train on varying amounts of training data to maintain consistency. All the baseline models are decoded without a language model. Figures 4.1 and 4.2 show the %WER performance of models trained with different amounts of supervision. Baseline models trained using WSJ show drastic improvement from 68.2 %WER to 41.5 %WER by increasing from 2 hours to 5 hours as in

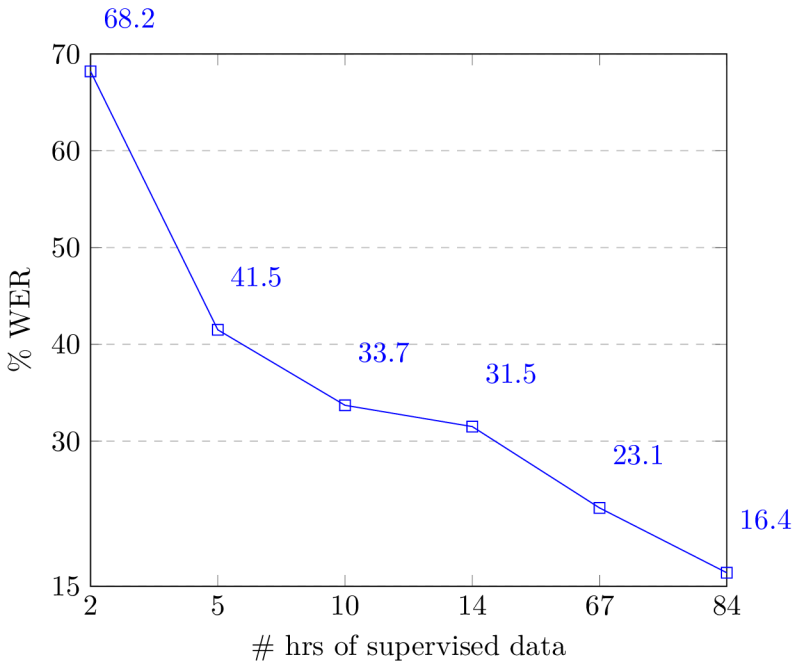


Figure 4.1: %WER of ASR model trained with different amounts of supervised data evaluated on the eval-92 test set.

figure 4.1. In the 5 hours to 14 hours range, the gains slowly decay, finally reaching 31.5 %WER. The performance starts to increase after 14 hours and reaches 16.4 %WER with 84 hours of training data. The performance of supervised ASR model trained by varying the amount of LibriSpeech data as in figure 4.2. The 500 hours subset contains only data matching 'test-other' set domain and hence shows gains in test-other but degradation in test-clean. There is consistent improvement in the other subsets: 100, 360, 460, 860 and 960 hours of LibriSpeech data.

4.6 Experimental Analysis on WSJ

Preliminary experiments are performed using the small WSJ dataset. In this section, the model is tested with varying amounts of unpaired data: 14 hours and 67 hours, along with a certain amount of semi-supervision: 2, 5, 10 and 14 hours of data. The ASR model is initialized with the pre-ASR-14h model as described in section 4.5.1. The effects of the amounts of paired and unpaired data (table 4.4) are shown on unpaired 'SO', 'TO' and 'SO+TO' data using the are shown on ASR→TTS, TTS→ASR and ASR↔TTS pipelines respectively.

4.6.1 Unpaired Speech Only (SO) Training

The experiments are performed starting with 2 hours of parallel data and 14 hours of unpaired data. As shown in table 4.4 with only 2 hours of parallel data, the model's %WER performance improves from 68.2 in figure 4.1 to 49.8 by only adding 14 hours of unpaired speech and improved to 51.9 with 67 hours of unpaired speech . The reason behind improvement through training with SO data is two fold:

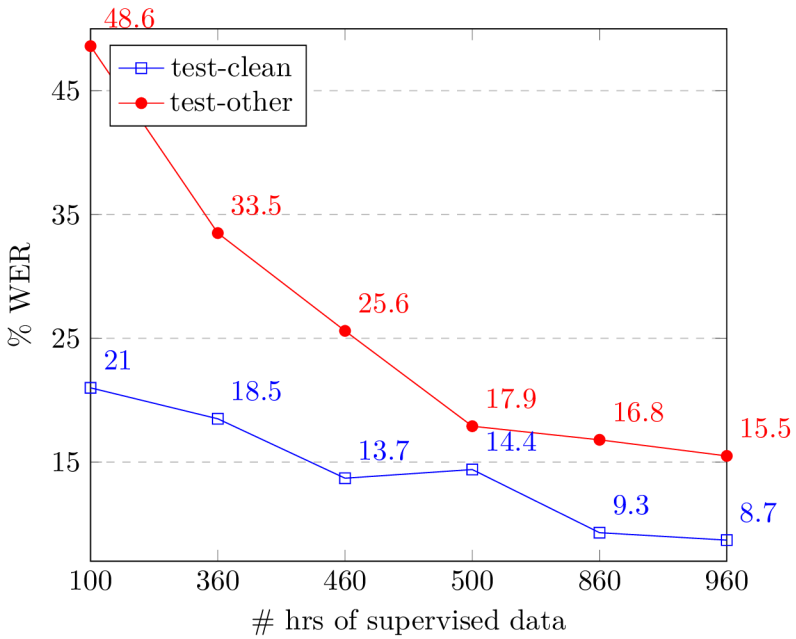


Figure 4.2: % WER of ASR models trained with different amounts of supervised data from LibriSpeech evaluated on test-clean and test-other datasets.

Table 4.4: %WER performance analysis using varying amounts of paired data (supervision) and unpaired data on the eval-92 test set using WSJ. The 'Data Type' refers to unpaired data used and it can be either speech only, text only or both ('SO, TO or SO+TO')

Unpaired data		# Hours of Paired data			
# Hours	Data Type	2	5	10	14
-	Baseline	68.2	41.5	33.7	31.5
14	SO	49.8	39.9	29.8	-
14	TO	63.0	43.6	34.6	-
14	SO+TO	43.7	35.5	28.3	-
67	SO	51.9	38.8	28.4	28.0
67	TO	39.6	36.8	29.6	27.1
67	SO+TO	41.4	34.2	27.7	26.2

- The model trained with 2 hours of paired data acts as a weak initialization for SO training leading to performance degradation compared to the TO model. However, SO training helps compared to the baseline performance. The results show that the weakly initialized (paired 2h) models benefit from TO training compared to SO training.

- The acoustic information is well learnt by the encoder component of the ASR system. The SO training helps to correct the errors due to incorrect pronunciations. For instance the first sequence in figure 4.3 shows that *'volunerable'* is corrected as *'volnerable'*. Although this term has imperfect spelling, it is acoustically more similar to *'vulnerable'*. Similar errors such as *'wecan'* and *'warmed'* are corrected as *'wekends'* and *'warmenth.*

REFERENCE: **wheat** may be **vulnerable** .. **weekend's warmth** analysts said
BASELINE: **weak** may be **volunerable** .. **we can's warmed** analysts said
ASR↔TTS: **week** may be **volnerable** .. **we kends warmenth** analysts said

REFERENCE: a slimmed down firestone can **prosper** merely
BASELINE: a slimmdown fire stonet can **prace bere** merely
ASR↔TTS: is slim down fire stoned can **prosper** merely

REFERENCE: he's **done** very well for ***** stockholders
BASELINE: eased **don't** very well for stock holders
ASR↔TTS: eased **done** very well for stock holders

REFERENCE: mr. polo also **owns** the fashion company
BASELINE: mr<UNK> polo also **owned** the fashion company
ASR↔TTS: mr<UNK> polo also **owns** the fashion company

REFERENCE: but other observers aren't so **pessimistic**
BASELINE: but other observers aren't so **pessibistic**
ASR↔TTS: but other observers aren't so **passimistic**

REFERENCE: investors are a conservative **lot** these days she says
BASELINE: investors are a conservative **lat** these days she says
ASR↔TTS: in deneral investors are a conservative **lot** these days she says

Figure 4.3: Hand-picked sequences from the eval-92 test set containing the reference text and the corresponding baseline model predictions. The ASR↔TTS predictions are compared with the baseline and the reference.

- The alignment between the encoder and decoder units is improved to with the aid of unsupervised speech data as shown in right plot in figure 4.5. The alignment abnormalities in baseline plot are rectified with SO training.

4.6.2 Unpaired Text Only (TO) training

Overfitting is observed in the case of SO training, while in the case of TO data, the model improved consistently from 63% (paired 14 hours) to 39.6% (paired 67 hours). This suggests that the TO training provides a major contribution in a very low-resource scenario. The TO training strengthens the decoder component of ASR which improves the implicit language model learnt inside the decoder. This is observed in figure 4.3, where *'prosper'* is predicted as *'prace bere'* by the baseline model. Although, both words are acoustically similar, the spelling is incorrect; this is overcome with the aid of TO data training which predicts *'prosper'* correctly. Figure 4.5 shows that the alignment learned during TO training is better than with SO training.

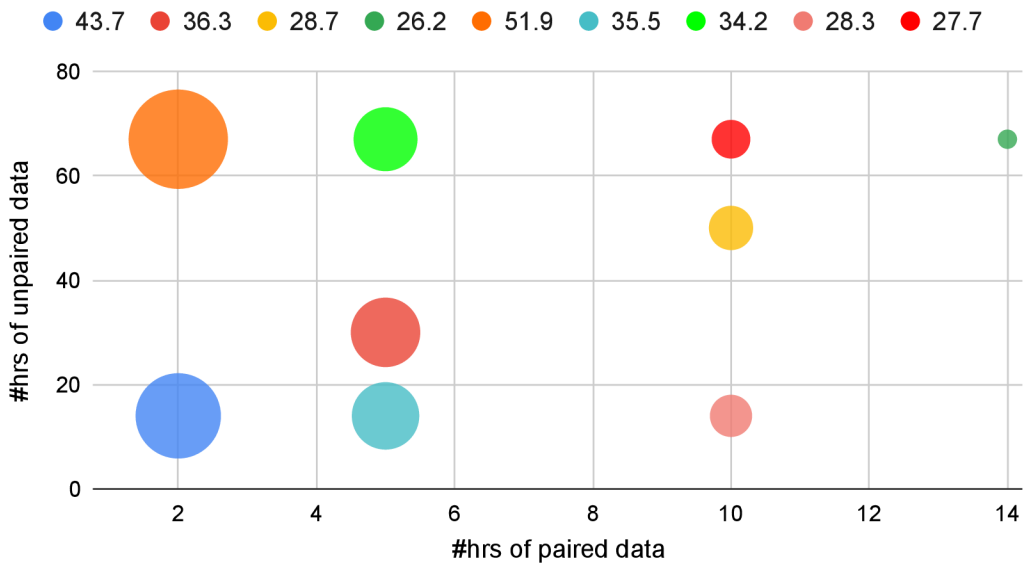
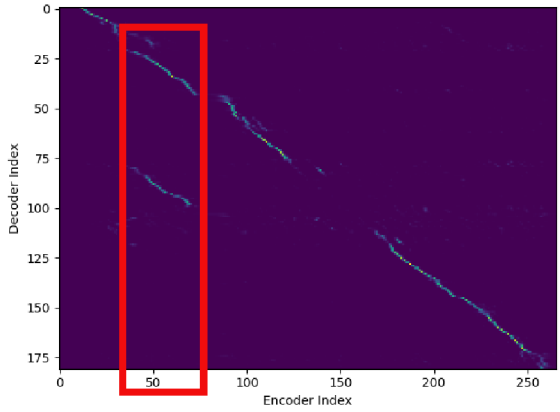


Figure 4.4: Bubble plot shows the impact of %WER by varying the amount of paired data in comparison to amount of the unpaired data

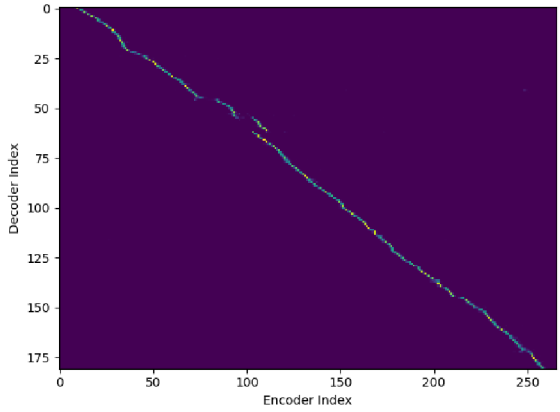
4.6.3 Unpaired Speech Only and Text Only (SO+TO) training

Interestingly, by including both unpaired speech and text of 14 hours, the %WER improved to 43.7, and with 67 hours of data the model obtained 41.4 %WER (for 2h of paired data) . The pattern emerging here is that, under a very low-resource scenario, the model benefits from large amounts of text. However, adding more speech only data leads to a slight degradation in performance. This pattern is not observed with 5, 10 and 14 hours of paired data conditions as increasing the amount of speech data from 14 hours to 67 hours improved by ∼1% absolute WER. With 14 hours of supervision and 67 hours of unpaired speech, text and both, the %WER improved to 28.0, 27.1 and 26.2, respectively.

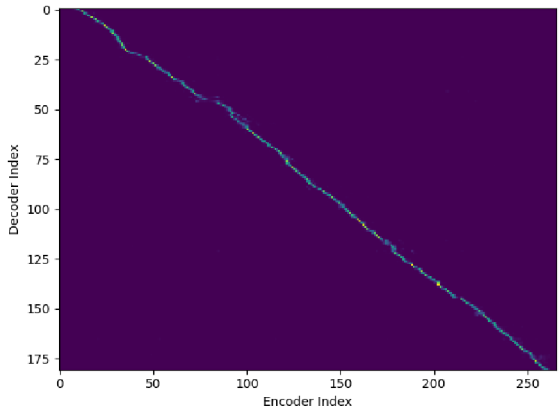
In table 4.4, the amount of paired data is varied and the amount of unpaired data is fixed as 14 hours and 67 hours. Figure 4.4 shows the effect of ASR↔TTS performance of linearly increasing the sizes of both paired and unpaired data. Here, the unpaired data sizes are 14, 30, 50 and 67 hours of SO+TO data, while the paired data sizes are 2, 5, 10 and 14 hours. In this plot, the size of the bubble represents the improvement in performance in-terms of %WER. The blue bubble denotes the 43.7 %WER performance obtained with 2 hours of paired data and 14 hours of unpaired data. Increasing the training data size to 5 and 30 hours of paired data and unpaired data respectively leads to 36.3 %WER as noted by red bubble. With 50 hours of unpaired data and 10 hours of paired data, the model attains better gains - see the yellow bubble (28.7 %WER). Finally, the 67 hours of unpaired data and 14 hours of paired data result in 26.2 %WER as plotted in the green bubble.



(a) ASR model trained with 100 hrs of LibriSpeech supervised training data. The red box shows the selected portion of encoder indices (x1=25, x2=75), where the decoder indices portion (y1=80, y2=100).



(b) ASR→TTS trained with 360 hrs of unpaired speech data.



(c) ASR↔TTS trained with 360 hrs of unpaired speech and text data.

Figure 4.5: Attention weights showing the alignment learnt between encoder (x-axis) and decoder (y-axis) units by supervised and unsupervised training. The alignment plot is obtained by forward propagating the a single utterance in dev-clean evaluation set. Each pixel denotes the attention weight a_{lt} of the t timestep for the l target label as in eqn (3.1).

4.7 Analysis of Attention alignment

A location aware attention module is a component of our seq2seq ASR model as described in section 2.3. On an example sentence, the encoder output contains 256 timesteps, and the decoder output contains 175 labels. In figure 4.5a, the encoder indices $(25, 75)$ are mapped to decoder indices $(80, 100)$. The attention weights to the same set of encoder indices $(25, 75)$ are also assigned to higher for the other set of decoder indices $(26, 50)$. Monotonicity is not learnt by this attention module due to lack of input samples. Retraining the same seq2seq ASR with unpaired speech data using ASR→TTS provides alignment weights which follow monotonicity as is in figure 4.5b. While, ASR→TTS helps to fix the major erroneous alignments, it fails to fix certain segments which are corrected with unpaired text training using the TTS→ASR model. Figure 4.5c, shows that the output of ASR↔TTS training with both unpaired speech and text data provides alignment weights monotonically aligning all encoder outputs to decoder labels.

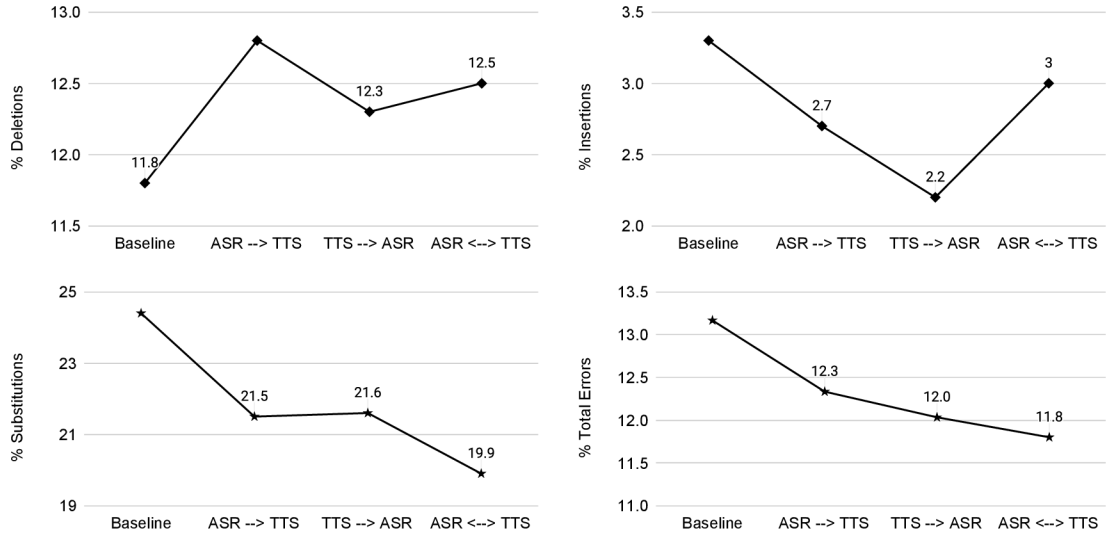


Figure 4.6: Analysis of deletions, insertions, substitutions and total error rates on eval-92 test set

4.8 Analysis of Deletions, Insertions and Substitutions

In this section, the performance improvements obtained using ASR↔TTS are analysed based on the percentage of substitution, deletion and insertion errors. Figure 4.6 shows that unpaired speech training using ASR→TTS improves the insertion and substitution significantly but degrades in deletion rate. The deletion rate is worse compared to baseline for the ASR→TTS, TTS→ASR and ASR↔TTS models. TTS→ASR significantly reduces the insertion errors and shows slight degradation in substitution rate. Being jointly trained with unpaired speech and text ASR↔TTS results in complementary behaviour in improving the substitution rate.

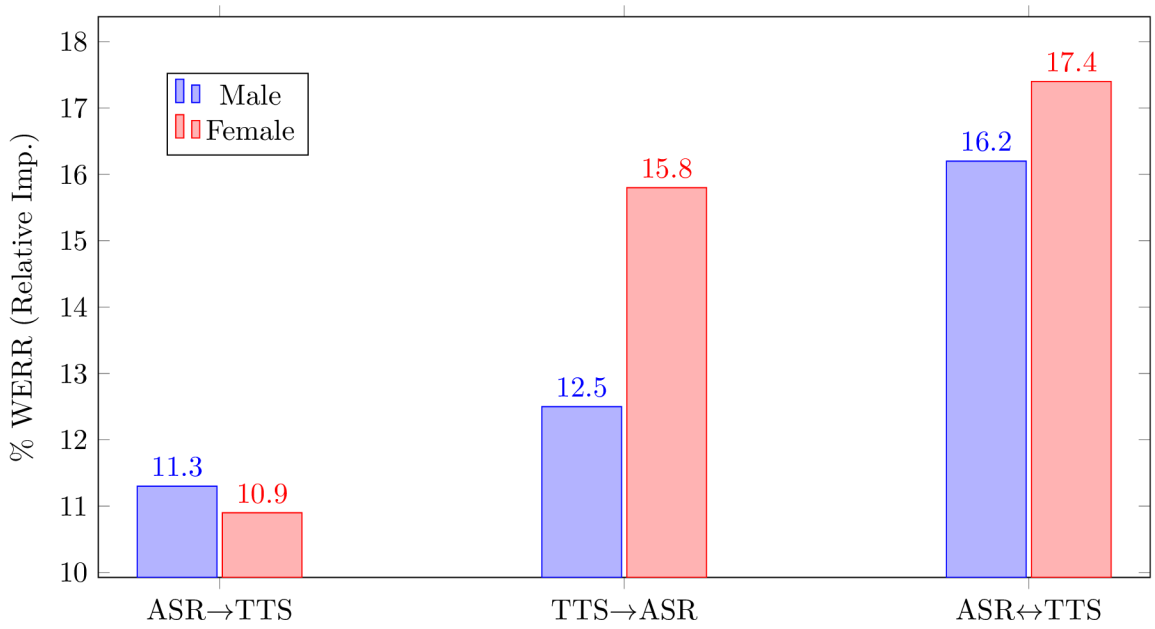


Figure 4.7: Relative improvement of WER for male and female speakers using ASR model trained using proposed ASR↔TTS pipeline. The baseline performance is 32.5 %WER for male speakers and 30.4%WER for female speakers.

4.9 Analysis of Speaker Type

Figure 4.7 shows that ASR→TTS attains high relative improvement in WER (WERR) for male compared to female speakers. With the inclusion of TTS→ASR, the % XWERR of female speakers over male speakers showed a 15.8% improvement. This comparison shows that unpaired text injection helps to improve female speakers and unpaired speech improves male speakers. ASR↔TTS based joint training helps to normalize the % XWERR improvements of both male and female speakers, by attaining 16.2% and 17.4% respectively

4.9.1 Effect of CTC and Attention

ASR↔TTS training involves re-training a pre-trained model using a multi-task objective. Thus, the parameters related to both CTC and attention are involved during training and decoding. In figure 4.8 , the performance is analysed by enabling and disabling the effect of CTC during training and decoding. The CTC scaling factor α is set as 1.0 and the attention is hence disabled during training and decoding. In this case only the CTC parameters are updated this results in,37.9 %WER. This is absolute 5% degradation over the baseline 32.2 %WER. In contrast, enabling only attention during training by setting $\alpha = 0.0$ updates only the parameters of attention-related components. Decoding using attention only requires careful selection of maxlen and minlen hyper-parameters and it achieves 26.3 %WER. The attention-only training resulted in better gains over the baseline 33.7 %WER.

Comparing attention and CTC shows that re-training only the parameters related to attention helps in improving the ASR performance with the ASR↔TTS method. Based on these observations, two variations of combinations of CTC and attention were analysed:

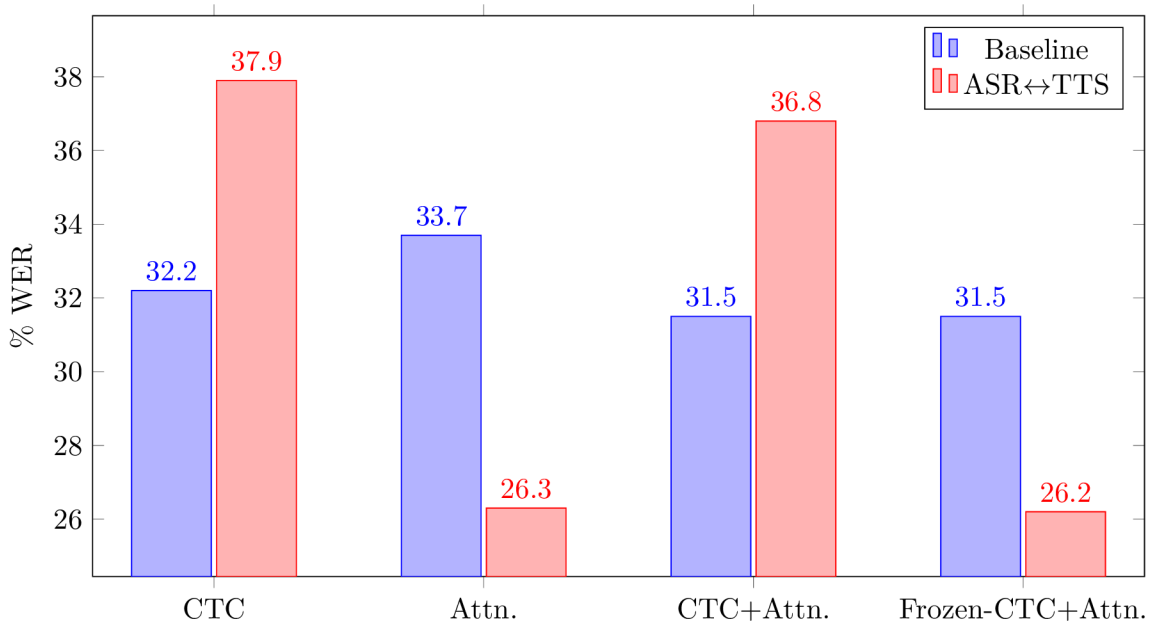


Figure 4.8: % WER of ASR model trained using our proposed ASR↔TTS pipeline by changing the effect of CTC and attention (attn.)

- **CTC+attn.**: The attention objective is combined with CTC with equal weight during training and decoding. This improves over the CTC-only model by reducing the %WER from 37.9 to 36.8. However, it is observed that combining attention with CTC did not help and the CTC still suffers with ASR↔TTS training.
- **Frozen-CTC+attn.**: To mitigate the effect of CTC, the parameters of CTC are frozen or not updated during training. The CTC components are only used during decoding to exploit the pre-trained information in CTC. ASR↔TTS shows minor improvement from attention only training by attaining 26.2 %WER. While the inclusion of the frozen-CTC's improvement can be treated as noise, it helps to minimize the effect of minlen and maxlen parameters during decoding.

4.10 Experiments on LibriSpeech

Table 4.5, shows the results of using both unpaired speech and text data from WSJ-SI284 and LibriSpeech 360 hours across the literature. In the WSJ corpus, the model achieves a %WER of 20.3 with RNNLM and 26.1 without RNNLM. This leaves a relative difference of 37.1% compared to Oracle's performance of 16.4%. The Oracle result is our baseline performance using test data WSJ-SI284 for training. Note that the performance on WSJ-SI284 is inferior to our previously reported baseline [Baskar et al., 2018]. This is due to a difference in architecture necessary to fit ASR and TTS models into the GPU.

In the case of LibriSpeech, the table 4.5 shows that training with unpaired audio and text data can achieve a %WER of 17.5, leading to 32.5% relative difference compared to the Oracle performance. Table 4.5 also shows that our approach using only unpaired text gains 18.0% relative improvement over the backtranslation-TTE [Hayashi et al., 2018] approach. Complementary gains of absolute 0.9% were observed by integrating RNNLM

Table 4.5: Unsupervised ASR performance across best results in the literature. Type refers to type of unpaired data used 'SO/TO/SO+TO'. The oracle result in WSJ table denotes the model trained with 84 hours of training data (WSJ-SI84 and WSJ-SI284)

. In LibriSpeech case, the oracle model is trained with 460 hours of training data.

WSJ-SI84 (paired) + WSJ-SI284 (unpaired)				
Model	Type	RNNLM	%CER	%WER
Speech chain [Tjandra et al., 2017]	SO+TO	-	9.9	-
Adversarial [Drexler and Glass, 2018]	SO+TO	yes	-	24.9
this work	SO+TO	-	9.1	26.1
this work	SO+TO	yes	**7.8**	**20.3**
Oracle	-	-	5.5	16.4
Oracle [Baskar et al., 2018]	-	yes	2.0	4.8
LibriSpeech 100 h (paired) + 360 h (unpaired)				
Backtranslation-TTE [Hayashi et al., 2018]	TO	-	10.0	22.0
this work	TO	-	8.0	17.9
Criticizing-LM [Liu et al., 2019]	TO	yes	9.1	17.3
this work	TO	yes	8.0	17.0
Cycle-TTE [Hori et al., 2019]	SO	yes	9.9	19.5
this work	SO	yes	7.8	16.8
MMD [Karita et al., 2018a]	Both	yes	8.4	18.0
this work	SO+TO	-	7.6	17.5
this work	SO+TO	yes	**7.6**	**16.6**
Oracle [Hori et al., 2019]	-	-	4.6	11.8

with this approach and the result is compared with criticizing-LM [Liu et al., 2019]. The effectiveness of our approach using only unpaired speech only data is shown in table 4.5 attaining 16.8% WER. Jointly training unpaired speech and text provided modest gains with a %WER improvement from 16.8 to 16.6. From these results, one can infer that training with unpaired SO data has major benefits over TO data on large corpus such as LibriSpeech.

4.11 Summary

This chapter presented a new approach to exploiting the information in unpaired speech and/or unpaired text to improve the performance of seq2seq ASR systems. It is showed that under low-resource conditions such as WSJ corpus the performance improvements are higher compared to a corpus with a sufficient amount of data such as LibriSpeech. It is also showed that integrating unpaired speech and text, both as a pipeline loss and through shallow integration with a RNNLM, provides additional gains and competitive results. Future work will focus on cycle-consistency approaches where ASR and TTS do not have matching conditions. Preliminary experiments show that this is a limitation of current systems.

Chapter 5

ASR→TLM: Language Model Prior for ASR→TTS

Consistency training acts as an effective technique to alleviate a lack of sufficient data for deep learning models. In chapter 3, the importance of consistency training for handling unpaired speech and text data was discussed. In particular, the ASR→TTS pipeline in the ASR↔TTS model handling the unpaired speech data faces language related errors as shown in figure 4.3. Intuitively, it seems that integration of LM with the ASR→TTS pipeline during training will help to address these errors. In this chapter, a novel consistency training pipeline named ASR→TLM is proposed by drawing inspiration from a variational auto-encoder (VAE) [Kingma and Welling, 2013]. Here, TLM refers to the integration of TTS and LM. The chapter initially discusses the relation between ASR→TTS pipeline and variational auto-encoder (VAE), proposes modifications to the architecture and loss objective and tests it empirically in the unpaired speech domain, deriving a generative model of speech.

5.1 Prior Work and Motivation

Prior to this work, studies such as [Hou et al., 2017, Jha et al., 2018] have imposed the consistency objective into VAE. Consistency is treated as a regularizer in [Hou et al., 2017] to make the output capture the necessary spatial characteristics of the input. A non-adversarial consistency objective is used to dis-entangle the latent space of VAE in [Jha et al., 2018]. While these studies used consistency as an auxiliary objective to VAE, they do not study the relation between both training objectives. To the best of our knowledge, our work is the first attempt to show the relation between consistency objective and the VAE, and also to fetch motivation from VAE to improve our ASR→TTS.

The primary motivation for inserting the notion of VAE into the consistency based ASR→TTS objective is to improve the representations learned by the ASR. In this work, the ASR→TLM treats the LM as a penalizer for language errors and contains the final loss objective analogous to the VAE objective. The proposed ASR→TLM model has the following credits:

- Prior knowledge is incorporated analogous to VAE with the aid of LM
- Including an LM based penalty over the ASR hypothesis is practically valid as it can scale predicted probabilities based on penalty score.

- The importance of the LM during training is exploited. The gradients from the LM score can rectify the imperfections in ASR hypotheses directly by acting complementarily to the penalty scores from the TTS model.
- The TTS penalty overcomes the acoustic errors while the LM penalty corrects the language-related errors.

Section 5.2 introduces the VAE framework and its training objective. Section 5.3, describes the ASR→TLM training scheme which is an extension of the proposed ASR→TTS presented in chapter 3. Finally, the section 5.5 tabulates the results on LibriSpeech and shows the efficacy of ASR→TLM compared to the previous approach.

5.2 Variational Auto-encoder (VAE)

VAE is a simple auto-encoding based encoder-decoder model. The encoder $f(.)$ is a non-linear neural network generating a latent variable z. The decoder $g(.)$ is another neural network with or without its parameters being shared with the encoder. The latent variable z is decoded by the decoder to reconstruct the input x. A simple reconstruction loss such as mean squared error (L2) or mean absolute error (L1) can be used to train this model. The nature of VAE differs from a conventional autoencoder in defining the training objective, as VAE encourages the latent variables to have a Gaussian distribution. This is done by minimizing a Kullback-Leibler (KL) divergence objective between the distribution of latent variables and the normal distribution.

5.2.1 Definition

Given an input x, the VAE introduces $q\left(z \mid x\right)$ which acts as a parametric model of the true-posterior $p(z \mid x)$. This is parameterized also by the encoder component $f(.)$ and hence the true posterior is denoted as:

$$\begin{aligned} p(z \mid x) &= \frac{p(x \mid z) \cdot p(z)}{p(x)} \\ \log\ p(z \mid x) &= \log \frac{p(x \mid z) \cdot p(z)}{p(x)} \\ &= \log p(x|z) + \log\ p(z) - \log p(x) \end{aligned} \tag{5.1}$$

Here, $p(x)$ is denoted as the evidence or normalization constant, $p(z)$ is the prior and $p(x|z)$ is the likelihood function. Estimating $p(z \mid x)$ using Bayes rule is difficult and hence variational inference using a family of distributions called $q\left(z \mid x\right)$ is used and the variational parameters ϕ are learnt using stochastic gradient descent. Henceforth, the estimation of the posterior $p(z \mid x)$ is tractable by introducing distribution $q(z|x) = \mathcal{N}(z; x, \mu, \sigma)$, where μ and σ are the mean and standard deviation. Standard normal distribution $\mathcal{N}(\mu, \sigma)$ is used. Usually, p(x)can then be expressed as:

$$\log p(x) = \log \int p(x,z)\, dz = \log \int p(x,z) \cdot \frac{q(z|x)}{q(z|x)}\, dz = \log(\mathbb{E}_q\{\frac{p(x,z)}{q(z|x)}\}) \tag{5.2}$$

This can be further expanded using Jensen's inequality and KL divergence to calculate the evidence lower bound (ELBO). In case of VAE, KL divergence is used and denoted as:

$$\begin{aligned}\mathcal{D}_{\mathcal{KL}}\{q(z|x)\|p(z|x)\} &= \mathbb{E}_{z\sim q}\{\frac{q(z|x)}{p(z|x)}\} \\ &= \mathbb{E}_{z\sim q}\{\log q(z|x) - \log p(z|x)\} \\ &= \mathbb{E}_{z\sim q}\{\log q(z|x) - \log\{\frac{p_\theta(x|z)\cdot p(z)}{p(x)}\}\} \\ &= \mathbb{E}_{z\sim q}\{\log q(z|x) - \log p_\theta(x|z) - \log p(z) + \log p(x)\}.\end{aligned} \tag{5.3}$$

Rewriting the above equation to get the variational lower bound or ELBO due to the presence of an evidence term results in:

$$\log p(x) - \mathcal{D}_{\mathcal{KL}}[q(z|x)\|p(z|x)] = \mathbb{E}_{z\sim q}[\log p(x|z) + \log p(z) - \log q(z|x)] \tag{5.4}$$

$$\log p(x) - \mathcal{D}_{\mathcal{KL}}\{q(z|x)\|p(z|x)\} = \mathbb{E}_{z\sim q}\{\log p_\theta(x|z)\} - \mathcal{D}_{\mathcal{KL}}\{q(z|x)\|p(z)\}. \tag{5.5}$$

Here, the objective is to maximize the evidence or marginal likelihood term $p(x)$. This is added with an error term $\mathcal{D}_{\mathcal{KL}}[q_\phi(z|x)\|p(z|x)]$ which tries to bring the distribution of latent variables closer to distribution $q(z|x)$. Therefore, reducing the error or learning the encoder to produce latent variables matching the distribution leads to maximizing the ELBO. Finally, the ELBO can be defined as:

$$\text{ELBO} = \mathcal{L}(x,q) = \log p(x) - \mathcal{D}_{\mathcal{KL}}\{q(z|x)\|p(z|x)\}. \tag{5.6}$$

For p(x) ,we can write:

$$\log p(x) \geq \mathcal{L}(x,q), \tag{5.7}$$

as the KL divergence is non-negative. Based on this assumption, (5.5) can also be used to define the ELBO as:

$$\text{ELBO} = \mathcal{L}(x,q) = \mathbb{E}_{z\sim q}\{\log p(x|z)\} - \mathcal{D}_{\mathcal{KL}}\{q(z|x)\|p(z)\} \tag{5.8}$$

5.2.2 VAE Training

Here, the $\mathbb{E}_{z\sim q}\{\log p(x|z)\}$ is the reconstruction error obtained at the end of the decoder. The KL divergence $\mathcal{D}_{\mathcal{KL}}[q(z|x)\|p(z)$ acts as a penalty or regularizer to ensure that the latent variables produced by the encoder stay within the defined distribution. During the training process, the objective is to maximize the marginal likelihood $\log p(x)$, this is done by indirectly maximizing the evidence lower bound (ELBO) as:

$$\log p(x) \geq \mathbb{E}_{z\sim q}\{\log p(x|z)\} - \mathcal{D}_{\mathcal{KL}}\{q(z|x)\|p(z)\} \tag{5.9}$$

Henceforth the objective ELBO can be generally defined as:

$$\log p(x) \geq \mathbb{E}_{z\sim q}[\log p(x|z) + \log p(z) - \log q(z|x)] \tag{5.10}$$

In summary, the joint distribution between input x and latent variable z, $p\,(x,z)$ is jointly optimized with $q\,(z \mid x)$, and to perform this joint optimization, the ELBO objective is

used. The variational parameters ϕ belonging to the encoder are instrumental in learning $q(z|x)$ and the decoder related parameters such as θ are also optimized. To ensure that the gradients from the decoder propagates till the encoder, a reparametrization trick is used which removes the hindrance due to sampled latent vectors between encoder and decoder.

There are many learning methods that optimize ELBO. An important difference between methods is whether the parameters for $q(\mathbf{Y} \mid \mathbf{X})$ are learnt from scratch at each iteration (e.g. EM or stochastic variational inference) or a single function is learnt during the entire training. The latter case yields the variational autoencoder (VAE [Kingma and Welling, 2013].

5.2.3 Relation ASR→TLM

The advantage in VAE is that it just implies using the stochastic gradient ascent (or momentum, ADAM, etc) in eq. (5.10) (usually reformulated to show the encoder and decoder more clearly). However in this form it it clear that the VAE loss closely resembles the loss for cycle consistency used in $\mathcal{L}_{\text{ASR}\to\text{TTS}}$. We can reformulate equation (5.10) as

$$\mathcal{L}_{\text{VAE}} = -\text{ELBO} = -E_{q(\mathbf{Y}|\mathbf{X})}\left\{\log p(\mathbf{X} \mid \mathbf{Y}) + \log p(\mathbf{Y}) - \log q(\mathbf{Y} \mid \mathbf{X})\right\} \tag{5.11}$$

or in the more commonly displayed form using the KL divergence as

$$\mathcal{L}_{\text{VAE}} = -E_{q(\mathbf{Y}|\mathbf{X})}\left\{\log p(\mathbf{X} \mid \mathbf{Y})\right\} + \text{KL}\left\{q(\mathbf{Y} \mid \mathbf{X}) \,||\, p(\mathbf{Y})\right\} \tag{5.12}$$

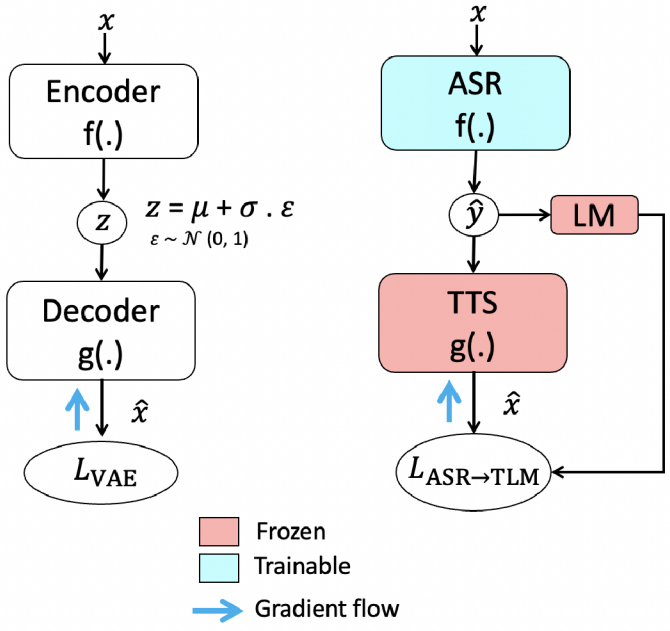


Figure 5.1: Flowchart showing training pipelines of the VAE and ASR↔TLM models

5.3 ASR→TLM

In this chapter, we propose ASR→TLM, a training pipeline interconnecting ASR with penalizers such as TTS and language model (LM). This model takes inspiration from the VAE

training objective and modifies the ASR→TTS pipeline proposed in chapter 3 to accommodate the language model penalty. The intuition is that during the initial course of training, the ASR tends to make some errors which will impose erroneous reconstruction by TTS. Apart from having a TTS penalizer to mitigate this problem as discussed in ASR→TTS (chapter 3), an additional penalty which directly targets the predicted hypotheses can be used. The ability of the LM to direct a correct token with a certain probability from the previously predicted tokens makes it a viable option to penalize the ASR hypotheses. The proposed ASR→TLM uses a pre-trained LM which intakes the predicted token sequence from ASR and outputs its probabilities. Here, the LM acts as a prior in defining the token sequences which can be learnt using unpaired text or TO data.

5.3.1 Relation to VAE

In this section, the encoder $f(.)$ and decoder $g(.)$ are generalized to simplify the comparison between VAE and ASR→TTTS. Given an input x, the encoder component $f(x)$ produces an output y. The decoder component $g(y)$ reconstructs back the input x. The parameters θ optimized here are based on the encoder component. The gradients for VAE loss defined in (5.11) are computed as:

$$\nabla_\theta \mathcal{L}_{\text{VAE}} \approx - \sum_{\mathbf{Y} \sim q(\mathbf{Y}|\mathbf{X})} T(\mathbf{X}, \mathbf{Y}) \nabla_\theta \log q(\mathbf{Y} \mid \mathbf{X}; \theta) \tag{5.13}$$

where

$$T(\mathbf{X}, \mathbf{Y}) = \log p(\mathbf{X} \mid \mathbf{Y}) + \log p(\mathbf{Y}) - \log q(\mathbf{Y} \mid \mathbf{X}) \tag{5.14}$$

In comparison, the gradients of $\mathcal{L}_{\text{ASR}\rightarrow\text{TTTS}}$ (3.8 equation from chapter 3) are:

$$\nabla_\theta \mathcal{L}_{\text{ASR}\rightarrow\text{TTS}} = \sum_{n=1}^{N} \log p_{\text{TTS}}(\mathbf{X} \mid \mathbf{Y}) \, \nabla_{\theta_{\text{ASR}}} \log \, p_{\text{ASR}}(\mathbf{Y} \mid \mathbf{X}) \tag{5.15}$$

Now, if we assume

- $q(\mathbf{Y} \mid \mathbf{X})$ is an ASR system,
- $p(\mathbf{X} \mid \mathbf{Y})$ is a TTS system and
- And we ignore L1 or end-of-utterance loss terms for TTS loss for simplicity

then it is apparent that the $\log p(\mathbf{Y})$ is missing in equation (5.15). Including the language model $p(\mathbf{Y})$ and rewriting (5.15) results in:

$$\nabla_\theta \mathcal{L}_{\text{ASR}\rightarrow\text{TLM}} \approx - \sum_{\mathbf{Y} \sim p_{\text{ASR}}(\mathbf{Y}|\mathbf{X})} T(\mathbf{X}, \mathbf{Y}) \nabla_{\theta \text{ASR}} \log p_{\text{ASR}}(\mathbf{Y} \mid \mathbf{X}) \tag{5.16}$$

with

$$T(\mathbf{X}, \mathbf{Y}) = \log p_{\text{TTS}}(\mathbf{X} \mid \mathbf{Y}) + \log p_{\text{LM}}(\mathbf{Y}) - \log p_{\text{ASR}}(\mathbf{Y} \mid \mathbf{X}) \tag{5.17}$$

This is in fact learning a generative model of speech that maximizes $\log p(\mathbf{X})$ by a variational autoencoder. It corresponds to the cycle consistency loss to which we add a Kulback-Leibler regularization term using the language model.

$$T(\mathbf{X}, \mathbf{Y}) = \beta_1 \log p_{\text{TTS}}(\mathbf{X} \mid \mathbf{Y}) + \beta_2 \log p_{\text{LM}}(\mathbf{Y}) - \log p_{\text{ASR}}(\mathbf{Y} \mid \mathbf{X}) \tag{5.18}$$

This new prior term using LM introduces smoothing of the distributions this is equivalent to flattening or sharpening the uncertainty in the language model and ASR distributions. In all our experiments $\beta_1 \rightarrow 1.0$ and the β_2 is varied based on the LM and the training dataset.

5.4 Experimental Setup

In this section, the experiments are performed using a fixed set of paired and unpaired data. Analysis is carried out to compare the SO training using ASR→TLM with language models trained with different datasets.

5.4.1 Database selection

Experiments are performed using WSJ and LibriSpeech corpora and later evaluated with eval-92 for WSJ and test-clean from LibriSpeech test sets. The primary component in training ASR→TLM is the language model which contains a vocabulary of 78 different characters. Two different language models, LM-1 and LM-2, are built:

- **LM-1**: 1662964 lines of character level text sequences are used to train an RNNLM. The text is specific to WSJ and does not overlap with the WSJ-SI84 and WSJ-SI284 datasets.
- **LM-2**: The data for training this RNNLM is obtained from the left-out set of LibriSpeech. The training data contains 40418261 lines of character level text and does not overlap with the 960 hours of LibriSpeech dataset.

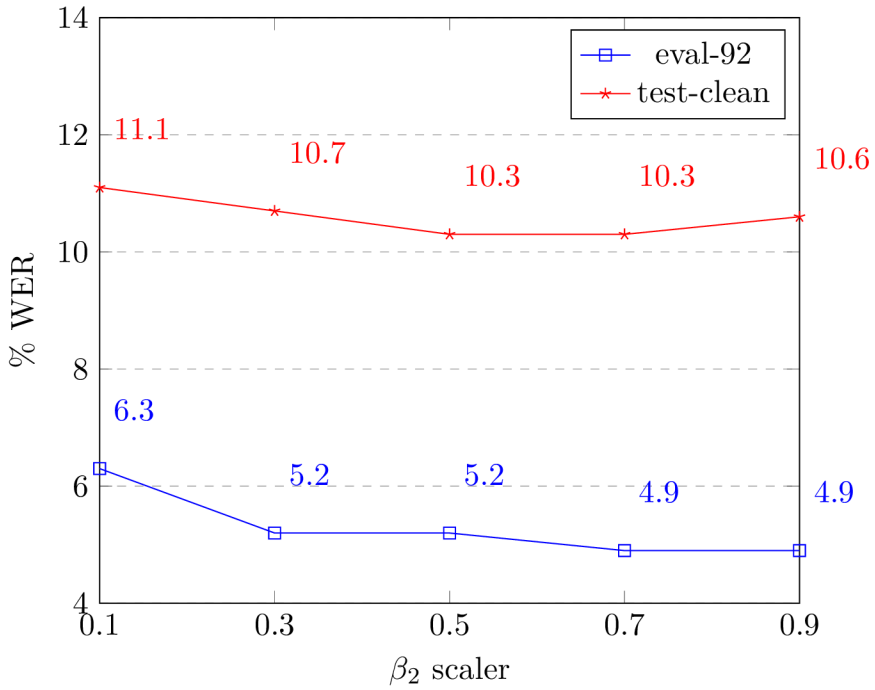


Figure 5.2: % WER of ASR→TLM-2 when decoding using shallow fusion with LibriSpeech RNNLM (SF-Lib) by varying the LM penalty factor β_2

5.4.2 Training

The training objective contains two hyper-parameters, α and β. The number of hypotheses predicted from ASR is set to $N = 4$ and the LM probabilities are obtained for all N hypotheses. The LM is constructed with 2 layers of uni-directional LSTM each with 650 units. The batchsize is kept at 1024 and the maximum length of each utterance is kept as 150 characters. The training is done for 20 epochs using the Adam optimizer. Early stopping is done if the perplexity fails to improve within 3 epochs. Two sets of experiments are done:

- **ASR→TLM-1**: The ASR→TLM which uses LM-1 during training
- **ASR→TLM-2**: The ASR→TLM which uses LM-2 during training

During the course of training with both models, the parameters of TTS and LM are frozen without being updated. Only the ASR parameters in the model are updated.

5.5 Results and analysis

All results on WSJ and LibriSpeech using the proposed ASR→TLM showed faster convergence when compared to the ASR→TTS model. Initial experiments are conducted on both WSJ and LibriSpeech by varying β_2 from 0.0 to 1.0 the performance is shown in figure 5.2. For both datasets,the results show consistent gain until β reaches 0.7.

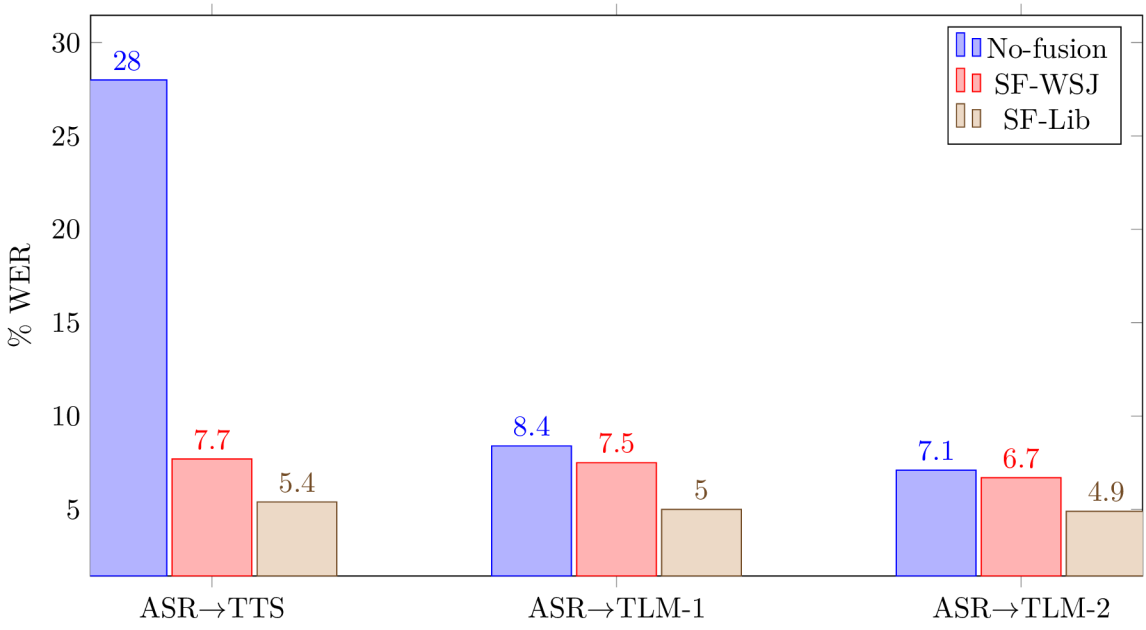


Figure 5.3: % WER of ASR model trained using our proposed ASR↔TTS pipeline and evaluated on eval-92 test set with shallow fusion using LM-1 (SF-WSJ) and LM-2 (SF-Lib) respectively.

5.5.1 Analysis on WSJ

The proposed ASR→TLM is compared with the ASR→TTS model on the WSJ corpus. The training data contains 14 hours of supervision and 67 hours of SO data. Figure 5.3 shows

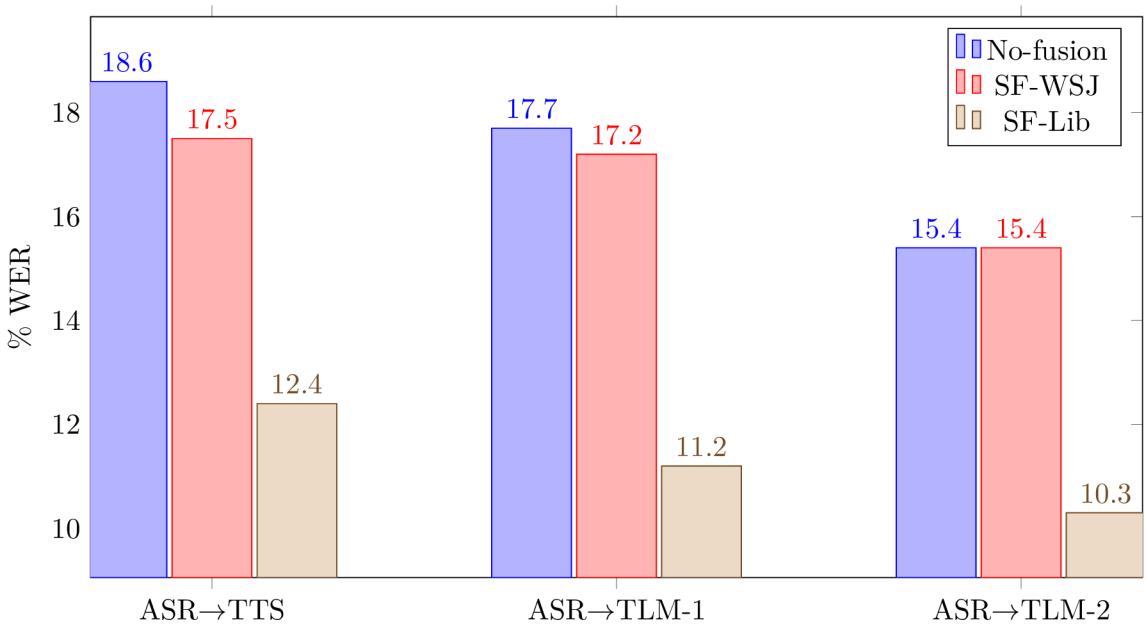


Figure 5.4: % WER of ASR model trained using our proposed ASR↔TTS pipeline and evaluated on test-clean set with shallow fusion using LM-1 (SF-WSJ) and LM-2 (SF-Lib) respectively.

that the ASR→TLM-1 model definitely enjoys the benefit of the LM-1 penalty by improving from 28.0% WER to 8.4% WER. Directly using LM-1 during decoding with shallow fusion provides 7.7% WER and is better than using it for the ASR→TLM-1 model. The primary reason for this scenario is that during shallow fusion, the LM encounters predictions of well trained ASR, while LM-1 in ASR→TLM-1 is used from the initial training stage and does not affect the final hypotheses. Shallow fusion with LM-2 attains better gain with 5.4% WER, which is absolute 2.3% WER improvement over LM-1. The drastic improvement is due to the presence of LM-2 with better perplexity over LM-1 in the development set of WSJ.

The ASR→TLM training proves to be complementary to shallow fusion with the same LM, that is LM-1. This is observed in figure 5.3, where the WER of ASR→TLM-1 improves from 8.4% to 7.5%. Shallow fusion with LM-2 achieves 5% WER due to the same effect as observed in ASR→TTS. ASR→TLM-2 attains 15.4% relative improvement over ASR→TLM-1 which showcases the importance of using LM trained with a significant amount of text data. Decoding ASR→TLM-2 with shallow fusion attains 5% relative improvement over model without fusion. However, ASR→TLM-1 attains 10% relative improvement with shallow fusion. This is because ASR→TLM-2 uses a bigger LM-2 over LM-1 during training and decoding with LM-1 did not provide noticeable gains as observed in ASR→TLM-1.

5.5.2 Analysis on LibriSpeech

The ASR→TLM-1 and ASR→TLM-2 did observe gains over the baseline ASR→TTS model on LibriSpeech as shown in figure 5.4. This experiment was conducted with LibriSpeech to show the impact of LM-1 when used along with 360 hours of SO data. Moreover, a larger model such as LM-2, was used to improve the ASR→TLM performance. Shallow fusion

of ASR→TTS with LM-2 attained 33.3% relative improvement, while using ASR→TLM-2 only gained 17.2% relative improvement over the ASR→TTS with no fusion. This shows the limitation of ASR→TLM-2 and indicates that the complete advantage of LM-2 can be exploited by training with of SO data. Morever, the advantage of LM-2 is still incorporated in ASR→TLM-2 with shallow fusion by attaining 10.3% WER, which shows the complementary nature of ASR→TLM training.

5.6 Conclusion

Usage of an LM has always played a major role in providing gain in recognition performance irrespective of the change in ASR architecture. The seq2seq models have also greatly benefited from shallow fusion with LMs. In addition to usual use of LM only during decoding the ASR→TLM proposed in this chapter , integrates the LM with ASR during training. This allows the ASR to learn from its own errors this claim is substantiated by the recognition performance on WSJ and LibriSpeech corpora. The experimental results show that the ASR→TLM proves to be complementary to the shallow fusion technique as different parts of errors are handled by incorporating LM during training and decoding. The LM penalizer acts as a regularization term in ASR→TLM and helps to improve over ASR→TTS.

Chapter 6

Enhanced ASR↔TTS (EAT)

The ASR↔TTS model was proposed in chapter 3 with experiments in chapter 4 and its improved counterpart ASR→TLM was described in chapter 5. The recognition performance of these models showed significant gains when evaluated using WSJ and LibriSpeech datasets. However, attempts to evaluate the ASR↔TTS against other datasets did not lead to reasonable gains. Existing works on semi-supervised learning such as machine speech chain [Tjandra et al., 2017] and other unsupervised training methods [Wang et al., 2020a, Rossenbach et al., 2020] have also evaluated the model performance only with certain datasets such as WSJ or LibriSpeech. Evaluating using these datasets projects the importance of the model only under in-domain conditions but fails to showcase the importance of the approach on other low-resource languages or under data mismatched conditions. This chapter discusses the issues of ASR↔TTS under mismatched conditions and provides suitable solutions. This chapter also provides improvements to ASR and TTS model architecture and their training schedules.

6.1 Out-of-domain (OOD) condition

The primary focus in this chapter is to understand the model behaviour when the SO+TO data does not match the data used to pre-train the ASR and TTS model. To facilitate this study, a low-resource language Swahili recorded in real-time conditions is obtained from the BABEL corpus. Swahili script uses ASCII characters and can be easily used with models pre-trained with LibriSpeech (English). Hence, BABEL-Swahili is treated as mismatched data or OOD data.

An empirical analysis of ASR↔TTS with BABEL-Swahili is conducted separately for ASR→TTS and TTS→ASR pipelines. The results in table 6.1 show that the TTS→ASR pipeline degrades in performance when compared to ASR→TTS.

Problem with TTS→ASR pipeline

In ASR→TTS TTS operates in the training mode. The auto-regressive training in TTS uses teacher-forcing and allows the ground-truth labels as previous tokens (filter-bank features) to predict the current token. This enables the TTS to provide reasonable reconstructed features even when OOD data is used. In the case of TTS→ASR, the TTS operates in decoding mode where the teacher-forcing is not possible. Instead, TTS relies on its own predictions to obtain the previous tokens and passes them to the decoder to predict the current token. Since the TTS is pre-trained with English speech, it learns the acoustic

context variations from an in-domain dataset such as LibriSpeech, exposing the pre-trained TTS to Swahili which contains acoustic contexts different from English. Swahili is chosen as OOD as it uses ASCII characters and is easier to plug into an existing English model.

A straight forward approach to solving this problem is to train a TTS with OOD data. However the TTS training requires a significant amount of training data (atleast 1000 hours) to obtain reasonable reconstructions. Using an exotic language such as Swahili, which contains a very limited amount of supervised data, hampers building a robust TTS for Swahili. Multilingual TTS [Zhang et al., 2019] should be a simple solution to building a speech synthesizer for low-resource languages but it is still in research stages and not flexible to adaptations using consistency training.

Table 6.1: Comparison of recognition performance of different ASR models trained with BABEL-Swahili.

Model	% WER
Topline - 40 hrs	52.4
ASR↔TTS	66.2
TTS→ASR	74.1
ASR→TTS	67.6
Baseline	71.4

Proposed Solutions

To counter this limitation, a well trained TTS must be adapted for a target language such as Swahili, with a limited amount of supervised data. Instead of directly adapting the TTS model with OOD data, an auxiliary model is used to guide the TTS to learn the new acoustic context variations in the OOD data. This reduces the burden of TTS by learning the target directly from the data.

- A hyper-parameter α is introduced to attenuate the influence of the ASR encoder by scaling the attention-encoded context. This allows reducing the focus on acoustic information when the synthesized speech quality is poor, effectively alternating between ASR and a more language-model-like behaviour.
- A new text-only training pipeline named TTS→AFV is proposed integrating feat2vec[1] [Baevski et al., 2020] to the existing TTS→ASR pipeline in chapter 3. Here, AFV refers to the ASR and feat2vec model. The feat2vec model acts as a penalizer to TTS outputs. The contrastive loss objective in feat2vec is integrated along with the existing TTS→ASR pipeline. This model encounters the incorrect TTS predictions, and guides them to produce reasonable reconstruction outputs. The proposed TTS→ASR model addresses TO data effectively.

6.1.1 Handling TTS problem with OOD data

In this section, a quick analysis is performed to show the problem faced by TTS during prediction of TTS outputs when fed with OOD data. Figure 6.1 plots the distribution of three different types of input text to show how a domain change in text leads to a change

[1]feat2vec is the variant of wav2vec2 with input as filterbank features

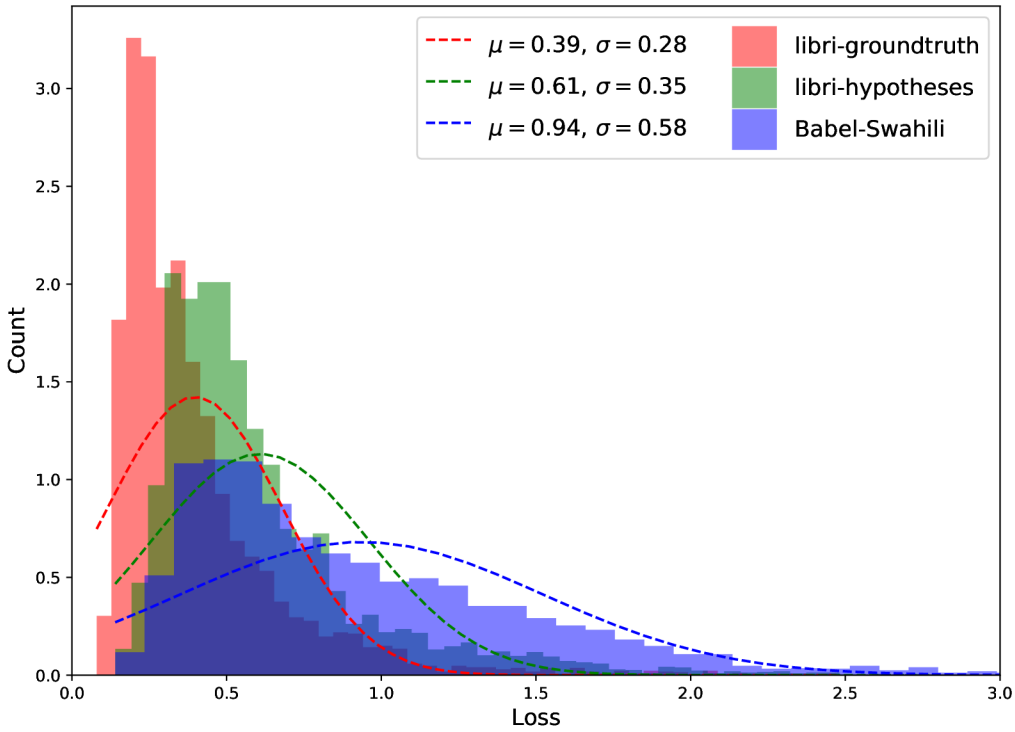


Figure 6.1: Distribution of TTS predictions across different domains of data

in TTS predictions. Figure 6.1 also shows the limitations of TTS in handling OOD data such as Swahili when it is trained with an English corpus. For instance, the distribution of predictions (red) when ground-truth (in-domain) text is provided has its mean centred around 0.39. The ASR hypotheses containing slight variations (with 16% word errors) from the ground-truth result in predictions (green) with their mean slightly shifted from the red distribution. However, the ASR hypotheses still lie under the in-domain category as the context does not vary from the in-domain text. In the case of the ground-truth Swahili (out-of-domain) text, the distribution (blue) diverges much farther from the green and red distributions as the MSE error increases due to increase in difference between the English and Swahili contexts. For example, the word '*hizo*' in Swahili does not occur in the LibriSpeech corpus. These factors make the Babel-Swahili PDF mean $\mu = 0.94$ widely deviating from the original $\mu = 0.39$. In addition to this, the Swahili has a higher variance of $\sigma = 0.58$ compared to both the LibriSpeech groundtruth and hypotheses.

6.1.2 Attention context scaler α

An adhoc way to handle the OOD problem with synthesized speech is to scale the output of TTS by a hyper-parameter and vary it based on the heuristic knowledge regarding how the target text varies from the text used to pre-train TTS. Instead of directly scaling the filterbank output from TTS, the output of the attention component in ASR is suppressed to reduce the effect of acoustic context. The idea behind this approach is to exploit the TO data to train the decoder component of the ASR for improving its implicit language model. Figure 6.2 shows the architecture of TTS→ASR with a hyper-parameter α to scale the attention context output of the ASR.

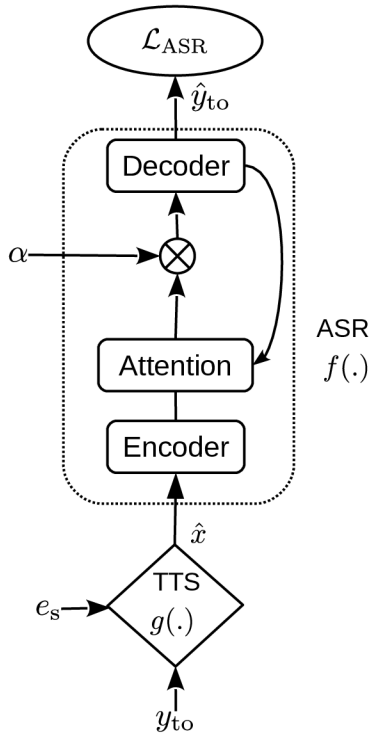


Figure 6.2: Block diagrammatic view of TTS→ASR with attention context scaler α

The TTS→ASR cycle-consistent TO training objective from section 3.4 exhibits a major weakness when training with out-of-domain data. TTS is less robust to out-of-domain data and generates poor log-mel filterbank (fbank) frames in this conditions. In this pipeline, features $\hat{\mathbf{X}}$ are predicted by TTS as

$$\hat{\mathbf{X}} = \arg\max_{\mathbf{X}} \{p_{\text{TTS}}(\mathbf{X} \mid \mathbf{Y})\}, \tag{6.1}$$

encoded in the ASR encoder as $H = \text{Encoder}(X)$ and sent to the attention component to obtain the attention context vector c_l as

$$c_l = \sum_t a_{lt} h_t, \tag{6.2}$$

where, t and l denote the timestep and token ID, respectively. The final loss is then given by

$$\mathcal{L}_{\text{TO}} = -\log p_{\text{ASR}}(\mathbf{Y}^* \mid \hat{\mathbf{X}}) = -\sum_l^L \log \text{Decoder}(c_l, y_{l-1}). \tag{6.3}$$

The primary reason behind this is that the ground truth is available in ASR→TTS to perform teacher-forcing. However in TTS→ASR, the ground truth is not available for TTS, and thus the reconstructed output deviates from the ground truth. Even the speech silence, segmentation may be wrongly predicted in the TTS→ASR pipeline. Therefore, we modify (6.2) to:

$$c_l = \alpha \sum_t a_{lt} h_t \tag{6.4}$$

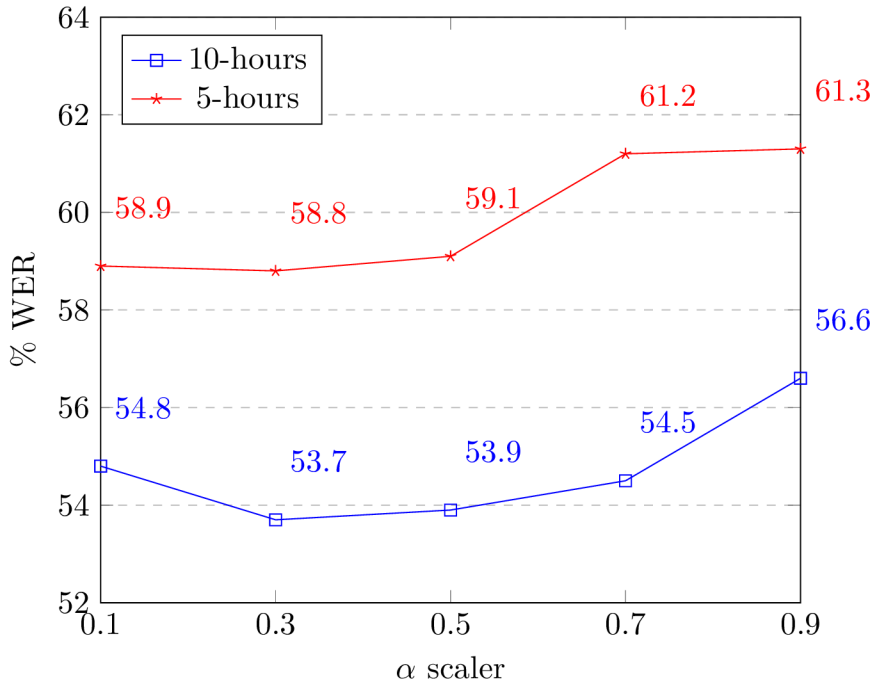


Figure 6.3: % WER of TTS→ASR with attention context scaler α on Swahili with 5 hours and 10 hours of supervision

If $\alpha = 0$, no encoder features are used and only the decoder component of the ASR model is trained to learn the language model information implicitly. This prevents the erroneous TTS generated features from providing a misleading signal, while still allowing gradient updates only to the ASR decoder. The value of α is chosen heuristically based on the difference in domains between data used to train TTS and the TO data. The final loss, used both for speech only and text only data (ST) is given by summing the loss functions $\mathcal{L}_{\text{ST}} = \mathcal{L}_{\text{SO}} + \mathcal{L}_{\text{TO}}$ of the above pipelines as introduced in section 3.5.

Figure 6.3 shows how α affects the recognition performance on Swahili when semi-supervised with 5 hours and 10 hours of target data. The plot justifies the idea that increasing α after 0.3 degrades the performance of Swahili for both 5 and 10 hours of semi-supervised training. 10-hours of semi-supervision proved to be more sensitive to an increase in α from 0.5 to 0.7 when compared to 5-hours model. The details of the experiments are given in the experimental section 6.4.3.

6.1.3 TTS→AFV: Need of regularizer for TTS→ASR

The significant improvement in performance obtained by scaling the attention context with α shows the negative impact of passing the OOD acoustic information predicted by the in-domain TTS model. To mitigate this issue conceptually, the TTS needs to be re-trained or fine-tuned with a large amount of paired data which is harder to obtain for languages such as Swahili. A straightforward way to solve this inconsistency of domains is to use models such as multilingual TTS or low-resource TTS, but these models are still in research stage and cannot be easily integrated with other models.

We propose simple yet effective approach to overcome this issue by regularizing the TTS model to provide reasonable context with an auxiliary model. The auxiliary model

acts as a penalizer to correct the context learned by the TTS during the course of TO based training. In this section, a feat2vec based penalizer is incorporated into TTS→ASR to guide the TTS in learning newly encountered context variations and provide reasonable reconstructed features to ASR as input.

Feat2vec Model

The feat2vec model provides acoustic context representation by learning from the input filter-bank features. The model is a variant of the wav2vec 2.0 model proposed in [Baevski et al., 2020]. With speech waveform as input (as in chapter 2) it directly predicts context representations. The input component of wav2vec 2.0 is modified to intake the filter-bank features instead of waveform to fit into the TTS→ASR model pipeline. The feat2vec model encodes the incoming filter-bank features via self-attention layers inside the transformer encoder to capture contextual information. The model is trained in a self-supervised fashion using contrastive loss objective to match the quantized targets with the transformer encoder outputs:

$$\mathcal{L}_{cont} = -\log \frac{\exp(sim(c_t, q_t)\ k)}{\sum_{\bar{q}\in\mathcal{Q}} \exp(sim(c_t, \bar{q})/k)}. \tag{6.5}$$

Here, c_t and q_t are the transformer encoder output and quantizer output obtained by by feeding filterbank features as input. More details are in section 2.6.1. Figure 6.4 shows a block diagram view of the feat2vec model.

Procedure

Feat2vec primarily focuses on capturing accurate context information at every time instant present in the feature input. The contrastive loss training objective provides an estimate of the accuracy of the context predicted by the feat2vec model. This characteristic of the feat2vec makes it an attractive auxiliary model to correct the inconsistent context predicted by the TTS model.

The TTS→ASR is incorporated with the feat2vec model as a *plug and play* module and is named the TTS→AFV (TTS→ASR + feat2vec) pipeline. Figure 6.5 shows the TTS→AFV model architecture with the attention context scaler inside ASR (see figure 6.3) and feat2vec integrated in parallel to the ASR model. Given a text input from the TO data y_{to}, the TTS generates filterbank frames $\hat{x}$ which are passed to both the ASR and feat2vec models. The final objective $\mathcal{L}$ of the TTS→AFV is determined as:

$$\mathcal{L} \leftarrow \mathcal{L}_s + \alpha \cdot \mathcal{L}_{\text{TTS}\rightarrow\text{AFV}} \tag{6.6}$$

where the semi-supervision objective $\mathcal{L}_s$ to the ASR is

$$\mathcal{L}_s = CE(\hat{y}_{to}, y_{to}). \tag{6.7}$$

The unsupervised training objective $\mathcal{L}_{\text{TTS}\rightarrow\text{AFV}}$ for TO training is determined as:

$$\mathcal{L}_{\text{TTS}\rightarrow\text{AFV}} = \mathcal{L}_{\text{ASR}} + \mathcal{L}_{cont}. \tag{6.8}$$

Here, the $\mathcal{L}_s$ and $\mathcal{L}_{\text{ASR}}$ are different as $\mathcal{L}_s$ is trained with supervised data containing parallel speech and text. While the $\mathcal{L}_{\text{ASR}}$ is determined using the speech generated by the TTS and the text from the TO data.

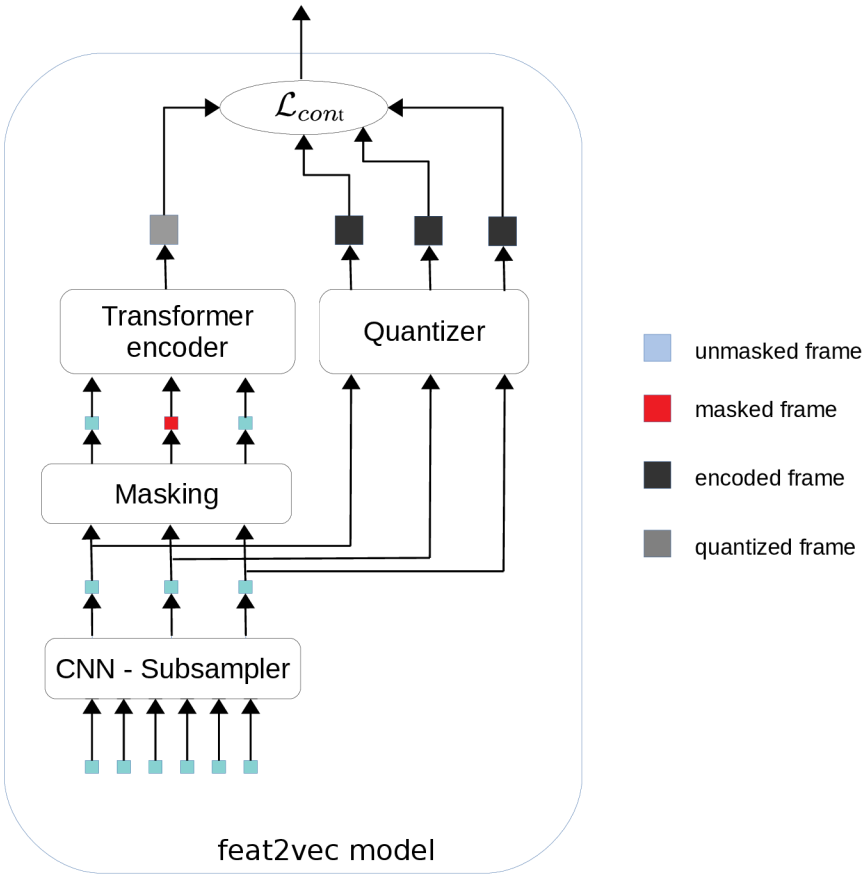


Figure 6.4: Feat2vec model architecture. $\mathcal{L}_{cont}$ is the feat2vec loss which is the constrastive objective to match the model predictions from transformer encoder with quantized targets from quantizer.

6.1.4 Training procedure

Algorithm 4 shows the complete procedure involved in training the TTS→AFV pipeline. The feat2vec θ_{FV} is initially pre-trained with a huge amount of unsupervised data and the parameters are not updated (kept frozen) during the complete training process. The TTS provides the feature vectors $\mathbf{X}$ and they are passed simultaneously to the feat2vec and the ASR models.

Feat2vec produces the quantized output and the encoder output capturing acoustic context. These are used to compute the contrastive loss as defined in (6.5). The ASR module provides the cross-entropy loss $\mathcal{L}_{\text{ASR}}$ and the contrastive loss $\mathcal{L}_{\text{cont}}$ is computed with feat2vec. The gradients of contrastive loss do not impact the parameters of ASR and only update the TTS related parameters. The gradients of the cross-entropy objective should impact both the TTS and ASR related parameters. Our assumption is that the gradients backpropagated from ASR will help the TTS to correct its language related errors, while the gradients propogated through feat2vec updates will aid the TTS to fix the imperfections in predicting acoustic contexts.

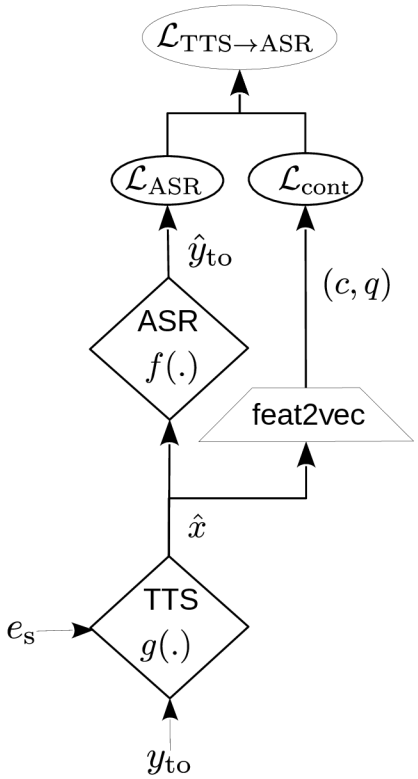


Figure 6.5: TTS→AFV model: TTS→ASR training pipeline integrated with feat2vec

6.2 Enhanced ASR↔TTS (EAT)

In this section, the upgraded models namely ASR→TLM from chapter 5 and TFV→ASR from the previous section are integrated to perform both SO and TO training as introduced in chapter 3. The updated model is termed as enhanced ASR↔TTS (EAT) and in addition, incorporates the following changes over our previous approach:

- Transformer architecture is incorporated into the encoder layers of seq2seq ASR and TTS of the ASR↔TTS model. The previous framework was constructed using recurrent architectures that limited the learning of longer context during training. The recently proposed transformer architecture [Vaswani et al., 2017] is capable of capturing longer context and helps in generalizing across temporal characteristics using the self-attention layers in both the ASR [Zeyer et al., 2019, Karita et al., 2019] and TTS [Li et al., 2019, Hayashi et al., 2020] models.
- Specaugment [Park et al., 2019] based augmentation strategy is introduced into our EAT model to empirically prove that consistency training is complementary to augmentation techniques and also leads to improved performance.
- Inconsistency between the amount of unpaired and paired data is addressed by proposing different annealing techniques during training.

The experiments are conducted on LibriSpeech and BABEL-Swahili and compared with the state-of-the-art (SoTA) baseline ASR systems. The analysis proves the importance of using transformer based architecture, specaugment and annealing strategy.

Algorithm 4 TTS→AFV training algorithm

Require: Text input $y_{to} \in \mathcal{D}_{\text{to}}$ from TO dataset $\mathcal{D}_{\text{to}}$, paired speech utterance with text $(x_s, y_s) \in \mathcal{D}_{\text{s}}$ from supervised dataset $\mathcal{D}_{\text{s}}$, pre-trained ASR θ_{ASR} and TTS θ_{TTS} models, pre-trained feat2vec θ_{FV} model, attention context scaling hyper-parameter α, learning rate γ

repeat

$y_{\text{to}} \in \mathcal{D}_{\text{to}}$. // Sample a text sequence

$\hat{x} \leftarrow \text{TTS}(y_{\text{to}})$

$\hat{y}_{\text{to}} = \text{Attention Context Scaled ASR}(\hat{x})$

$\mathcal{L}_{\text{ASR}} \leftarrow -\log\, p_(\hat{y}_{to} \mid \hat{x})$

$\mathcal{L}_{cont} = \text{feat2vec}(\hat{x})$

$\mathcal{L}_{\text{TTS}\rightarrow\text{AFV}} = \mathcal{L}_{\text{ASR}} + \mathcal{L}_{cont}$

$x_{\text{s}}, y_{\text{s}} \in \mathcal{D}_{\text{s}}$ // Sample speech with corresponding text

$\hat{y} \leftarrow \text{ASR}(x_{\text{s}})$

$\mathcal{L}_{\text{s}} \leftarrow -\log\ p(\hat{y} \mid x_s)$

$\mathcal{L} \leftarrow \mathcal{L}_s + \mathcal{L}_{\text{TTS}\rightarrow\text{AFV}}$ // Final objective

if update TTS **then**

$\hat{\theta}_{FV} \leftarrow \theta_{FV}$ // Feat2vec parameters are not updated

$\hat{\theta}_{TTS} \leftarrow \theta_{TTS} + \gamma \cdot \nabla_{\theta_{TTS}} \mathcal{L}$ // TTS update

end if

$\hat{\theta}_{ASR} \leftarrow \theta_{ASR} + \gamma \cdot \nabla_{\theta_{ASR}} \mathcal{L}$ // ASR update

until convergence

6.3 Experimental Setup

LibriSpeech [Panayotov et al., 2015] and BABEL-Swahili [Karafiát et al., 2016] datasets are used in our experiments. WSJ-sI284 is used to pre-train the ASR and TTS models. 5 and 10 hours of paired data are obtained from 39.74 hours of BABEL-Swahili data and the rest of the dataset is used as unpaired data. In BABEL-Swahili, the 40 hours of training data is partitioned to pairec (5h, 10h) and unpaired (30h, 35h) without any overlap between paired and unpaired utterances. The test set is based on the original split in the corpus [Karafiát et al., 2016]. Table 6.2 shows the number of utterances in Swahili training and test splits.

Table 6.2: BABEL Swahili corpus description with training (both supervised and unsupervised) and test splits

Type	# Hours	# Utterances
Paired Train split	5.0	5000
Paired Train split	9.9	10180
Paired Train split	40	40053
Unpaired Train split	350	30000
Unpaired Train split	35	35000
Test split	1.84	1991

83 dimensional filterbank features are extracted and used to train our ASR and TTS systems. RNNLM for LibriSpeech is built with 460 hours of clean data and 500 hours of other data. RNNLM for Swahili is built with external text containing 83k utterances and a 62k vocabulary size.

Our experiments are performed using ESPnet toolkit and the code is published on GitHub.All experiments are conducted with RNNLM during testing.

Evaluation with LibriSpeech is carried out on dev-clean, dev-other, test-clean and test-other as such variability can showcase the effectiveness of EAT.

6.3.1 Feat2vec pre-training

The CNN component in feat2vec contains two 2D convolutional layers: a) the first layer with 512 dimensional input and output channels, kernel size of 3 and stride factor of 2. b) the second layer contains both input and output channels with size of 512, kernel size of 3 and stride of 2. Outputs of both convolutional layers are subject to ReLU activation functions.

The transformer encoder module contains 12 encoder layers, each with 2048 dimensions. Each encoder layer contains a multi-head attention component with 8 heads and 512 attention dimensions. Two feed-forward layers, each with 2048 dimensions are applied after multi-head attention component to capture position-wise information. Layer normalization is applied after the above procedure before proceeding to the next layer of the encoder.

The masking and quantization hyper-parameters are kept similar to wav2vec2.0 [Schneider et al., 2019]. The masking is done over $N_M = 10$ timesteps with a probability of 0.065 over all time samples to choose the starting indices I. The quantization with Gumbel-softmax is performed with a temperature parameter varying from $\tau = 2$ to $\tau = 0.5$ with increases in training iterations. The number of groups $G = 2$ and the number of variables M is set as 320. The number of distractors for contrastive loss in equation (6.5) is chosen empirically as 50.

The feat2vec model training is performed with both contrastive loss and diversity loss, and the contrastive loss is only used after integrating with TTS→ASR. The feat2vec model is pre-trained with the following open-source corpora:

- 1000 hours of Libri-light data. The Libri-light data does not overlap with the 960 hours of LibriSpeech data used for TTS→AFV fine-tuning.
- 500 hours of TED talks [Rousseau et al., 2012].
- 2000 hours of Fisher corpus [Cieri et al., 2004].

The above datasets are used in a completely unsupervised fashion as only the speech data is required. The models are trained with a batch size of 15 million bins [Watanabe et al., 2018] for a maximum of 500 epochs.

6.3.2 EAT Architecture and Training its Improvements

Model Architecture

The ASR and TTS architecture in the EAT model is meticulously designed as it plays a major role in attaining improved performance. The motivation is to keep the model light-weight and simple to easily fit in GPU memory during training.

ASR: The ASR component in the EAT model is equipped with a location based multi-head attention component [Karita et al., 2019]. Instead of RNN layers in the encoder, self-attention layers are used to reduce model complexity. The decoder is built with RNN layers as before, since the transformer decoder is more difficult to implement with our training objective. In these experiments, 2 VGG [Simonyan and Zisserman, 2015] layers followed by 6 self-attention layers each with 800 dimensions are used. The encoder output is sent to attention component with 10 heads and 512 dimensions. 10 convolution channels with 100 filters are used in this attention to be location specific. Adadelta optimizer [Zeiler, 2012b] is used and the training proceeds with batch size 20. Our experiments shows that multi-head attention and self-attention layer based encoder provided performance gains.

TTS: Transformer-based TTS [Hayashi et al., 2020, Li et al., 2019] is used in this work, as we found that the transformer consumes less memory and is effective in out-of-domain conditions when compared to Tacotron architecture [Hayashi et al., 2020]. The TTS is multi-speaker based and handles each speaker input by providing an x-vector [Snyder et al., 2018] as speaker embedding. The transformer architecture contains 6 encoder and decoder layers, each with 1536 units, respectively. The attention component contains 4 attention heads, each with 384 attention dimensions. 2 pre-net layers with 256 units and 5 post-net layers with 256 channels are used. The output frames reduction factor is set to 1 as all frames are required during self-supervised training. Speaker embeddings are added to the encoder output before sending to the decoder. The pre-training of transformer TTS is done using 24 hours of LJSpeech (single speaker TTS dataset) [Ito and Johnson, 2017] and the training is carried out using its standard optimizer with 10000 warmup steps for 100 epochs.

Data Augmentation

Specaugment: In ASR↔TTS, simple Gaussian noise is used as augmentation to stabilise the training; it provides minor gains but is inconsistent across datasets. In this work, the EAT model is trained with the specaugment [Park et al., 2019] approach. The frequency mask and time mask are applied using a window width of 30 consecutive log-Mel frequency channels and 40 consecutive timesteps respectively. The recognition performance of the EAT model with specaugment is shown in table 6.4. Specaugment approach attains consistent gains using self-supervised training with SO and ST (SO + TO) data. No augmentation is done during training with TO data.

`in general investors are a conservative lot these days she says` (ground-truth)
`in general investors are a conservative lat these days she says` (baseline)
`in deneral investors are a conservative lot these days she say` (ASR↔TTS)

Figure 6.6: An example of text sequence predicted by baseline and ASR↔TTS compared with the ground-truth

Data Annealing

In ASR↔TTS model training, alternating between large amounts of unpaired data and less paired data is difficult. The paired training of certain labels can result in over-fitting, which hinders the effect of unsupervised training [Xie et al., 2019] as shown in figure 6.6.

Here, 'general' in reference text is correctly predicted in baseline training. During ASR-TTS training, the supervised samples from baseline and unsupervised samples are provided

alternately in a linear fashion. Although, in chapter 3 we repeated the supervised data to reduce the under-fitting, it still resulted in incorrect predictions due to overfitting from supervised data and also increased the training time due to redundancy. To mitigate this, the supervised samples are released only when the probability $p_{\text{ASR}}(\hat{y}_l \mid x)$ of a particular label is greater than a threshold l:

$$p_{\text{ASR}}(\hat{y}_l \mid x) > \gamma_t \tag{6.9}$$

Here the hyper-parameter, $\gamma_t = \eta_t \times (1 - \frac{1}{K}) + \frac{1}{K} \times \eta_t$, where K is the number of classes. For log schedule:

$$\eta_t = 1 - \exp(\frac{t}{T} \cdot 5) \tag{6.10}$$

and $\exp((\frac{t}{T} - 1) \times 5)$ for exponential schedule, where T is the number of training steps. Table 6.3 shows that linear and exponential schedules are better than log schedule, as release of supervised data is high initially and reduces at the end of training. The performance of EAT trained using exponential schedule outperforms linear schedule as the supervised data is mostly released at the final stage of training, paving a smoother way for training with unsupervised data.

6.3.3 EAT setup

The experiments are primarily conducted to evaluate the TTS→AFV model on OOD data. The ASR and TTS models are pre-trained with conventional English data from WSJ-si284 which contains clean speech. The model is tested under two separate conditions: 1) when the unpaired text data (text from TED talks) is taken from other datasets to evaluate English speech from LibriSpeech test sets (target), and 2) when the amount of paired data is limited for exotic language such as Swahili compared to language English corpora. This helps to show the importance of unsupervised training with very minimal supervision. In addition to this, the TTS systems used to train Swahili are still pre-trained with English speech due to lack of a sufficient amount of Swahili data. This analysis provides the effect of the model under the same language but OOD with respect to the vocabulary and the acoustic contexts. Two distinct experimental setups are suggested by varying the datasets from which the TO data is chosen:

- **Libri-TO**: 500 hours (268083 lines) of training text from the TED dataset is used to train the TTS→AFV model in an unsupervised fashion as described in chapter 3. The TED corpus is chosen as the vocabulary used in the TED talk dataset which is different from LibriSpeech test sets. This allows for analysing the effect of TTS→AFV under OOD condition with the English dataset.
- **Swahili-TO**: The TO data for Swahili is obtained from the open-source corpus named ALFFA, containing 9.9 hours (10180 lines) of training text and around 70 hours (30053 lines) of text from the BABEL-Swahili dataset.

During training, the Libri-TO is mixed with semi-supervised data taken from 100 hours of LibriSpeech corpus similar to previous experiments on chapters 3 and 5. For Swahili-TO, the semi-supervision is performed with either 10 or 20 hours of data selected from the BABEL-corpus. These two datasets are used to train two separate TTS→AFV models to evaluate the LibriSpeech (test-clean, test-other) and BABEL-Swahili test sets.

6.4 Results and Discussion

6.4.1 Ablation studies on EAT using WSJ and LibriSpeech

EAT was initially tested with different data annealing schedules to choose the best training schedule for the remainder of the experiments. A total of 360 hours of both speech only and text only (SO+TO) data is used during EAT training and the results for log, linear and exponential schedules are in table 6.3. The results show that exponential schedule is better and thus was used in rest of our experiments. Table 6.4 shows the effect of data

Table 6.3: %WER performance of log, linear and exp based annealing schedules data during self-supervised training using EAT

360 SO+TO	dev-clean	dev-other	test-clean	test-other
log	7.7	23.5	6.9	24.3
linear	7.1	22.7	6.9	23.6
exp	6.9	22.5	6.9	22.1

augmentation by specaugment. SO training with 360 hours of data attained consistent gains on all evaluation sets. The 360 SO+TO denotes that the 360 hours of speech only and text only data were used simultaneously for training EAT. The performance improvements obtained by SO and SO+T0 with augmentation shows that the EAT model training is complementary to the specaugment approach.

6.4.2 LibriSpeech

The attention context vector in ASR is scaled by $\alpha = 0.7$ for LibriSpeech and attains 15.9% WER on test-other when compared to 24.1% WER without α scaler. The EAT model is further improved with the proposed TTS→AFV resulting in 4.3% WER on test-clean and 14.7% WER on test-other. The oracle experiment is done by training an ASR with 460 hours of supervised data and it attains 3.5 %WER on test-clean and 12.6 %WER on test-other. The simple, yet effective method of integrating ASR→TLM and TTS→AFV has allowed EAT to further improve its performance proving that SO and TO pipeline are complementary to each other.

6.4.3 Swahili

The key results of the EAT are shown on BABEL-Swahili as it helps to focus on the impact of ASR→TLM and TFV→AR compared to our previous ASR↔TTS model. The reason behind the difficulty is that building a multi-speaker TTS model for Swahili is challenging and hence our previous work failed to provide reasonable TTS scores. The EAT model mitigates this problem by modifying its TTS architecture and reducing the importance of synthesized speech from TTS by feat2vec. Here, the pre-trained TTS is retrained during the TO training with $\alpha = 0.3$. Table 6.4 shows that with 5 hours of paired data, the effect of TO is higher compared to SO, but with 10 hours the TO obtains comparable gains as SO. With α in TO the model obtains 58.8 %WER and further reduced to 58.5 %WER with SO+TO training. Here, Oracle in the table denotes the performance of the ASR model trained with a complete (39.75 hours) training set in Swahili. With 10 hours of paired data

Table 6.4: Detailed comparison of recognition performance of EAT model trained using 360 hours of SO, TO and SO+TO data with 100 hours of supervision for English and 5h, 10h of supervision for Swahili. Oracle model for LibriSpeech is trained with 460 hours of supervised data and oracle Swahili model is trained with 40 hours of supervised data. Shallow fusion is not performed. Pretrained LM (PT LM) is used in ASR→TLM.

Model	Training Description	Libripeech 100 hrs paired 360 hrs unpaired				Swahili 5hrs paired 35 hrs unpaired	Swahili 10hrs paired 30 hrs unpaired
		dev	dev-other	test	test-other	test	test
Baseline	Supervised (100 hrs)	16.7	40.4	17.5	41.3	71.4	64.7
	+ trans. enc., specaug	14.3	36.4	14.4	36.9	70.1	63.5
ASR→TTS	SO pipeline	11.0	32.4	10.6	33.6	63.9	61.1
	+ specaug	9.1	24.2	8.9	25.0	61.4	55.0
ASR→TLM	+ PT LM	6.0	18.6	5.8	19.0	60.1	54.8
TTS→ASR	TO pipeline	8.9	23.0	8.6	24.1	62.2	60.9
	+ α scaler	4.5	15.8	4.7	15.9	61.7	54.0
TTS→AFV	+ PT feat2vec	4.3	**13.5**	4.5	**13.1**	58.8	53.7
ASR↔TTS	SO+TO pipeline	7.9	23.4	8.1	24.4	63.1	60.4
EAT	+ trans. enc., specaug	7.5	23.0	7.5	24.0	60.1	53.6
	+ annealing	6.9	22.5	6.9	23.6	59.7	53.0
	+ α scaler	5.2	19.5	5.3	20.4	58.5	52.5
	+ PT LM (ASR→TLM)	4.3	14.9	**4.3**	15.2	58.4	51.6
	+ feat2vec (TTS→AFV)	**4.2**	13.6	**4.3**	14.7	**57.0**	**50.1**
Oracle	Supervised	3.7	12.7	3.6	12.9	49.0	
	+ specaug	3.7	12.3	3.5	12.6	47.6	

and SO+TO training, the EAT model attained 50.1 %WER which is only absolute 2.4% less compared to Oracle's 47.6 %WER.

6.4.4 Related Works

Some of the recent works using ASR and TTS to handle unpaired speech and text data have raised the performance benchmark on LibriSpeech.

Local Prior Matching (LPM)

LPM [Hsu et al., 2020a] is a technique integrating LM $p_{\text{LM}}(\mathbf{Y})$ with ASR $p_{\text{ASR}}(\mathbf{Y} \mid \mathbf{X})$ to train using a SO dataset. Given an utterance X from SO, the ASR model first generates a possible set of hypotheses (model distribution). The resulting hypotheses Y are sent to the LM to produce a target distribution. LPM objective minimizes the cross entropy between the target distribution and the ASR model distribution:

$$\mathcal{L}_{\text{LPM}} = -\sum_{\mathbf{Y}\in B} \frac{p_{\text{LM}}(\mathbf{Y})}{\sum_{\mathbf{Y}'\in B} p_{\text{LM}}(\mathbf{Y}')} \cdot \log p_{\text{ASR}}(\mathbf{Y}|\mathbf{X}) \tag{6.11}$$

Here, B is the set of beam search hypotheses generated by the ASR model.

Global Style Tokens (GST)

A simple and naive approach to using the TO data is proposed in [Rossenbach et al., 2020]. Tacotron TTS trained with unsupervised speaker embeddings named global style tokens (GST) is used to synthesize speech to train the ASR model.

Generative Consistent Predictions (GCP)

In GCP [Wang et al., 2020b], real and synthesized speech are used to provide consistent predictions. A consistency loss term is applied to reduce the difference between the predictions. Synthetic speech is generated using the TO data. SO data is also used with pseudo labels generated by another ASR model. The final GCP training objective is a combination of consistency loss and cross entropy loss using pseudo labels.

Comparison of Results

Table 6.5 compares the recognition performance of the related literature with our proposed EAT approach. The existing works in literature can be classified as models trained with SO and TO data respectively.

Training with SO data: The self-training approaches [Synnaeve et al., 2019, Xu et al., 2020b], uses a pre-trained ASR to predict the hypotheses from the SO data. The predicted pseudo-labels are used to retrain the ASR in supervised way. Self-training approach attains reasonably better performance on all dev and test sets over the baseline model. The errors in pseudo-labels were further corrected with a language model using LPM objective [Hsu et al., 2020a] and led to improvements in dev-other and test-other while the performance slightly degrades in dev-clean and test-clean. Our EAT model outperforms the self-training model on all evaluation sets as noted in table 6.5.

Table 6.5: Comparison of SotA results in literature with EAT model

Method	Type	dev		test	
		clean	other	clean	other
Self-training	Pseudo [Xu et al., 2020b]	5.41	20.31	5.79	21.63
	LPM [Hsu et al., 2020b]	5.69	20.22	5.99	20.93
Synthesis	GST [Rossenbach et al., 2020]	7.4	25.7	7.9	26.7
	GCP [Wang et al., 2020a]	4.1	-	4.1	-
Cycle	ASR↔TTS	11.0	32.4	10.6	33.6
	EAT	4.2	13.6	4.3	14.7

Training with TO: The GST method synthesizes speech from TO data to train the ASR model and improves the synthesis quality using GST based speaker embeddings. This model attains 7.4%WER and 7.9%WER on dev-clean and test-clean, respectively, these results are not spectacular due to the small language model used. In the case of GCP, the authors synthesize speech and use consistency loss together to attain 4.1% WER on dev-clean and 4.1% WER on test-clean. This is better than our EAT model because the GCP uses 460 hours of paired data while we use only 100 hours of paired data. The effect of penalizer, attention context scalar and other training strategies make our EAT model attain 4.2%WER on dev-clean and 4.3% on test-clean. Our model also attains the best performance in more difficult conditions such as dev-other and test-other.

Chapter 7

Conclusion

Summary

The aim of this thesis is presented in chapter 1 with emphasis on using unpaired speech and text to improve ASR. Introduction to ASR and a brief survey of related semi-supervised learning techniques are covered in chapter 2.

Objectives of this thesis

- End-to-end differentiable training pipeline by integrating seq2seq ASR and TTS respectively. The architecture is simple to construct and allows usage of existing ASR and TTS models.
- Synergise both unpaired speech and text data to reduce ASR errors by generalizing across: 1) acoustic variations 2) language variations.
- Understand the cycle-consistency objective for ASR and propose improvements to handle different data domains by analysing performance on harder test sets (eg: BABEL-Swahili).

Proposed Approaches and their Implications

ASR→TTS inherits REINFORCE based score function estimator to obtain improved performance by learning from multiple ASR hypotheses while providing end-to-end differentiability. Extensive analysis based on gender, CTC or Attention, amount of paired and unpaired data is done. The results show that ASR→TTS improves the alignment capability, reduces language related errors and leads to consistent gains on test conditions closer to paired and unpaired data. The last two chapters 5 and 6 focus on improving in SO training pipeline (ASR→TTS) and TO training pipeline (TTS→ASR):

ASR Training with LM: The TTS synthesizes speech based on the ASR predictions in ASR→ TTS model. While the acoustic variations are handled by the help of TTS scores, language related errors persists when unpaired data unseen by ASR is used during training. Integration of LM to correct the ASR hypotheses using ASR→TLM architecture solves this issue and further improves recognition performance on standard test sets. The model performance is complementary to shallow fusion by obtaining 16.9% and 17.4% relative improvement with and without shallow fusion respectively.

Training with Feat2vec: TTS→ASR struggles when fed with unseen text data sequence leading to incorrect scores. The proposed TTS→AFV model integrates a pretrained feat2vec model to finetune TTS under OOD conditions. This training does not require additional supervised data for finetuning TTS. Evaluation is done using test set (BABEL-Swahili). Swahili is a good candidate for OOD evaluation as it is written in ASCII characters but the context in text sequence is different from English words. Further improvements to architecture and training strategy reduces the gap between paired and unpaired data training in Swahili from 10.4% to 2.5%.

Future Considerations

Due to both the time and space limitations, a trainable TTS with speaker-embedding extractor and applying EAT for pseudo-label generation and voice conversion have not been explored.

In future, the EAT should be updated to perform synthesis with better and more TTS training data. EAT model's backbone is based on the pretrained ASR and TTS models. Pretraining has provided significant benefits for EAT in enabling greater accuracy, faster training, increased flexibility, and reduced data requirements. Incorporating latest and robust pretrained models is an important possible improvement to EAT in near future.

The performance gains in EAT have shown the importance of jointly training an ASR with multiple modalities such as speech and text. Two possible extensions to this will be to 1) have a shared encoder to jointly train with speech and text. 2) upsample the text sequence to resemble speech sequence or downsample the speech sequence to resemble text sequence. In this work, the primary focus was on improving ASR performance. Few works on machine translation and speech translation [Khadivi and Ney, 2008, Karafiát, 2021, Li and Specia, 2019] have shown the importance of using ASR for either correcting hypotheses or as data augmentation. Dialogue task [Dhole et al., 2021, Jianfeng et al., 2019] is another downstream case which can benefit from better ASR. This can be useful for a variety of applications, including chatbots, voice assistants, and customer service systems.

Bibliography

[Amodei et al., 2016] Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML*, pages 173–182.

[Baevski et al., 2020] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

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

[Bahdanau et al., 2016] Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. In *ICASSP*, pages 4945–4949.

[Baskar et al., 2018] Baskar, M. K., Burget, L., Watanabe, S., Karafiát, M., Hori, T., and Černockỳ, J. H. (2018). Promising accurate prefix boosting for sequence-to-sequence asr. *arXiv preprint arXiv:1811.02770*.

[Baskar et al., 2019] Baskar, M. K., Watanabe, S., Astudillo, R., Hori, T., Burget, L., and Černocký, J. (2019). Semi-Supervised Sequence-to-Sequence ASR Using Unpaired Speech and Text. In *Proc. Interspeech 2019*, pages 3790–3794.

[Bengio et al., 2013] Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013). Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907.

[Buciluundefined et al., 2006] Buciluundefined, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA. Association for Computing Machinery.

[Burget et al., 2010] Burget, L., Schwarz, P., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Povey, D., et al. (2010). Multilingual acoustic modeling for speech recognition based on subspace gaussian mixture models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4334–4337. IEEE.

[Chan et al., 2016] Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*, pages 4960–4964.

[Cho et al., 2018] Cho, J., Baskar, M. K., Li, R., Wiesner, M., Mallidi, S. H., Yalta, N., Karafiat, M., Watanabe, S., and Hori, T. (2018). Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling. *arXiv preprint arXiv:1810.03459*.

[Chorowski et al., 2014] Chorowski, J., Bahdanau, D., Cho, K., and Bengio, Y. (2014). End-to-end continuous speech recognition using attention-based recurrent NN: first results. *arXiv preprint arXiv:1412.1602*.

[Cieri et al., 2004] Cieri, C., Miller, D., and Walker, K. (2004). The fisher corpus: a resource for the next generations of speech-to-text. In *LREC*, volume 4, pages 69–71.

[Dhole et al., 2021] Dhole, K. D., Gangal, V., and et al. (2021). Nl-augmenter: A framework for task-sensitive natural language augmentation. *CoRR*, abs/2112.02721.

[Doddipatla et al., 2017] Doddipatla, R., Braunschweiler, N., and Maia, R. (2017). Speaker adaptation in dnn-based speech synthesis using d-vectors. In *INTERSPEECH*, pages 3404–3408.

[Drexler and Glass, 2018] Drexler, J. and Glass, J. (2018). Combining end-to-end and adversarial training for low-resource speech recognition. In *SLT*, pages 361–368.

[Gales et al., 2008] Gales, M., Young, S., et al. (2008). The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304.

[Glass, 2012] Glass, J. (2012). Towards unsupervised speech processing. In *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, pages 1–4. IEEE.

[Graves et al., 2006] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). 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.

[Graves and Jaitly, 2014] Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pages 1764–1772.

[Griffin and Lim, 1984] Griffin, D. and Lim, J. (1984). Signal estimation from modified short-time fourier transform. *IEEE Transactions on acoustics, speech, and signal processing*, 32(2):236–243.

[Hayashi et al., 2018] Hayashi, T., Watanabe, S., Zhang, Y., Toda, T., Hori, T., Astudillo, R., and Takeda, K. (2018). Back-translation-style data augmentation for end-to-end asr. In *SLT*, pages 426–433.

[Hayashi et al., 2020] Hayashi, T., Yamamoto, R., Inoue, K., Yoshimura, T., Watanabe, S., Toda, T., Takeda, K., Zhang, Y., and Tan, X. (2020). Espnet-tts: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7654–7658. IEEE.

[Hayashi et al., 2021] Hayashi, T., Yamamoto, R., Yoshimura, T., Wu, P., Shi, J., Saeki, T., Ju, Y., Yasuda, Y., Takamichi, S., and Watanabe, S. (2021). Espnet2-tts: Extending the edge of tts research. *arXiv preprint arXiv:2110.07840*.

[He et al., 2016] He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T.-Y., and Ma, W.-Y. (2016). Dual learning for machine translation. In *NIPS*, pages 820–828.

[Hinton et al., 2012] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

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

[Hori et al., 2019] Hori, T., Astudillo, R., Hayashi, T., Zhang, Y., Watanabe, S., and Roux, J. L. (2019). Cycle-consistency training for end-to-end speech recognition. In *ICASSP*.

[Hori et al., 2018] Hori, T., Cho, J., and Watanabe, S. (2018). End-to-end speech recognition with word-based RNN language models. *arXiv preprint arXiv:1808.02608*.

[Hori et al., 2017] Hori, T., Watanabe, S., Zhang, Y., and Chan, W. (2017). Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. In *INTERSPEECH*.

[Hou et al., 2017] Hou, X., Shen, L., Sun, K., and Qiu, G. (2017). Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141. IEEE.

[Hsu et al., 2020a] Hsu, W.-N., Lee, A., Synnaeve, G., and Hannun, A. (2020a). Semi-supervised speech recognition via local prior matching. *arXiv preprint arXiv:2002.10336*.

[Hsu et al., 2020b] Hsu, W.-N., Lee, A., Synnaeve, G., and Hannun, A. Y. (2020b). Semi-supervised speech recognition via local prior matching. *ArXiv*, abs/2002.10336.

[Hsu et al., 2017] Hsu, W.-N., Zhang, Y., and Glass, J. (2017). Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. *arXiv preprint arXiv:1707.06265*.

[Ito and Johnson, 2017] Ito, K. and Johnson, L. (2017). The lj speech dataset. `https://keithito.com/LJ-Speech-Dataset/`.

[Jha et al., 2018] Jha, A. H., Anand, S., Singh, M., and Veeravasarapu, V. (2018). Disentangling factors of variation with cycle-consistent variational auto-encoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 805–820.

[Jianfeng et al., 2019] Jianfeng, G., Michel, G., Lihong, L., et al. (2019). Neural approaches to conversational ai. *Foundations and Trends, in Information Retrieval*, 13:127–298.

[Karafiát, 2021] Karafiát, M. (2021). The iwslt 2021 but speech translation systems. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*.

[Karafiát et al., 2016] Karafiát, M., Baskar, M. K., Matějka, P., Veselỳ, K., Grézl, F., and Černocky, J. (2016). Multilingual blstm and speaker-specific vector adaptation in 2016 but babel system. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 637–643. IEEE.

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

[Karita et al., 2018a] Karita, S., Ogawa, A., Delcroix, M., and Nakatani, T. (2018a). Sequence training of encoder-decoder model using policy gradient for end-to-end speech recognition. In *ICASSP*, pages 5839–5843.

[Karita et al., 2018b] Karita, S., Watanabe, S., Iwata, T., Ogawa, A., and Delcroix, M. (2018b). Semi-supervised end-to-end speech recognition. In *Interspeech*, pages 2–6.

[Khadivi and Ney, 2008] Khadivi, S. and Ney, H. (2008). Integration of speech recognition and machine translation in computer-assisted translation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1551–1564.

[Khurana et al., 2021] Khurana, S., Moritz, N., Hori, T., and Roux, J. L. (2021). Unsupervised domain adaptation for speech recognition via uncertainty driven self-training. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6553–6557.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Li et al., 2014] Li, J., Zhao, R., Huang, J.-T., and Gong, Y. (2014). Learning small-size dnn with output-distribution-based criteria. In *Fifteenth Annual Conference of the International Speech Communication Association*.

[Li et al., 2019] Li, N., Liu, S., Liu, Y., Zhao, S., and Liu, M. (2019). Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6706–6713.

[Li and Specia, 2019] Li, Z. and Specia, L. (2019). Improving neural machine translation robustness via data augmentation: Beyond back-translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 328–336.

[Liu et al., 2019] Liu, A., Lee, H.-y., and Lee, L.-s. (2019). Adversarial training of end-to-end speech recognition using a criticizing language model. In *ICASSP*.

[Microsoft,] Microsoft. Scaling speech, language and vision models with mixture of experts technique. (Date last accessed 29-April-2022).

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[Oord et al., 2018] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

[Panayotov et al., 2015] Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: an asr corpus based on public domain audio books. In *ICASSP*, pages 5206–5210.

[Park et al., 2019] Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

[Paul and Baker, 1992] Paul, D. B. and Baker, J. M. (1992). The design for the Wall Street Journal-based CSR corpus. In *Proc. of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics.

[Pesme and Flammarion, 2020] Pesme, S. and Flammarion, N. (2020). Online robust regression via sgd on the l1 loss. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2540–2552. Curran Associates, Inc.

[Povey et al., 2011a] Povey, D., Burget, L., Agarwal, M., Akyazi, P., Kai, F., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Rastrow, A., et al. (2011a). The subspace gaussian mixture model-a structured model for speech recognition. *Computer Speech and Language*, 25(2):404–439.

[Povey et al., 2011b] Povey, D., Burget, L., Agarwal, M., Akyazi, P., Kai, F., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Rastrow, A., Rose, R. C., Schwarz, P., and Thomas, S. (2011b). The subspace Gaussian mixture model-a structured model for speech recognition. *Computer Speech & Language*, 25(2):404–439.

[Povey et al., 2011c] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011c). The kaldi speech recognition toolkit. In *Automatic Speech Recognition and Understanding, 2011 IEEE Workshop on*, pages 1–4. IEEE.

[Prabhavalkar et al., 2017] Prabhavalkar, R., Rao, K., Sainath, T. N., Li, B., Johnson, L., and Jaitly, N. (2017). A comparison of sequence-to-sequence models for speech recognition. In *Interspeech*, pages 939–943.

[Rabiner and Juang, 1986] Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *IEEE ICASSP magazine*, 3(1):4–16.

[Ranzato et al., 2015] Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

[Renduchintala et al., 2018] Renduchintala, A., Ding, S., Wiesner, M., and Watanabe, S. (2018). Multi-modal data augmentation for end-to-end asr. In *Interspeech*, pages 2394–2398.

[Rossenbach et al., 2020] Rossenbach, N., Zeyer, A., Schlüter, R., and Ney, H. (2020). Generating synthetic audio data for attention-based speech recognition systems. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7069–7073. IEEE.

[Rousseau et al., 2012] Rousseau, A., Deléglise, P., and Estève, Y. (2012). TED-LIUM: an automatic speech recognition dedicated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 125–129, Istanbul, Turkey. European Language Resources Association (ELRA).

[Schneider et al., 2019] Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*.

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

[Seide et al., 2011] Seide, F., Li, G., and Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *INTERSPEECH*, pages 437–440.

[Sennrich et al., 2015] Sennrich, R., Haddow, B., and Birch, A. (2015). Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

[Shen et al., 2018] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. (2018). Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *ICASSP*, pages 4779–4783.

[Sim and Li, 2008] Sim, K. C. and Li, H. (2008). Robust phone set mapping using decision tree clustering for cross-lingual phone recognition. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4309–4312. IEEE.

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

[Snyder et al., 2018] Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., and Khudanpur, S. (2018). X-vectors: Robust dnn embeddings for speaker recognition. In *ICASSP*, pages 5329–5333.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

[Swietojanski et al., 2014] Swietojanski, P., ., and Renals, S. (2014). Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 171–176. IEEE.

[Synnaeve et al., 2019] Synnaeve, G., Xu, Q., Kahn, J., Likhomanenko, T., Grave, E., Pratap, V., Sriram, A., Liptchinsky, V., and Collobert, R. (2019). End-to-end asr: from supervised to semi-supervised learning with modern architectures. *arXiv preprint arXiv:1911.08460*.

[Tarvainen et al., 1780] Tarvainen, A., ., and Valpola, H. (1780). Weight-averaged, consistency targets improve semi-supervised deep learning results. *CoRR,, vol. abs/1703*, 2017.

[Thrun, 1995] Thrun, S. (1995). Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, 8.

[Tjandra et al., 2017] Tjandra, A., Sakti, S., and Nakamura, S. (2017). Listening while speaking: Speech chain by deep learning. In *ASRU*, pages 301–308.

[Tjandra et al., 2018] Tjandra, A., Sakti, S., and Nakamura, S. (2018). End-to-end feedback loss in speech chain framework via straight-through estimator. *arXiv preprint arXiv:1810.13107*.

[Tjandra et al., 2020] Tjandra, A., Sakti, S., and Nakamura, S. (2020). Machine speech chain. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:976–989.

[Van Den Oord et al., 2016] Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

[Verma et al., 2019] Verma, V., Kawaguchi, K., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. (2019). Interpolation consistency training for semi-supervised learning. *arXiv preprint arXiv:1903.03825*.

[Wang et al., 2020a] Wang, G., Rosenberg, A., Chen, Z., Zhang, Y., Ramabhadran, B., Wu, Y., and Moreno, P. (2020a). Improving speech recognition using consistent predictions on synthesized speech. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7029–7033. IEEE.

[Wang et al., 2020b] Wang, G., Rosenberg, A., Chen, Z., Zhang, Y., Ramabhadran, B., Wu, Y., and Moreno, P. (2020b). Improving speech recognition using consistent predictions on synthesized speech. pages 7029–7033.

[Wang et al., 2007] Wang, L., Gales, M. J., and Woodland, P. C. (2007). Unsupervised training for mandarin broadcast news and conversation transcription. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–353. IEEE.

[Wang et al., 2017] Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q. V., Agiomyrgiannakis, Y.,

Clark, R., and Saurous, R. A. (2017). Tacotron: Towards end-to-end speech synthesis. In *INTERSPEECH*, pages 4006–4010. ISCA.

[Watanabe et al., 2018] Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Enrique Yalta Soplin, N., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., and Ochiai, T. (2018). Espnet: End-to-end speech processing toolkit. In *Interspeech*, pages 2207–2211.

[Watanabe et al., 2017a] Watanabe, S., Hori, T., Kim, S., Hershey, J. R., and Hayashi, T. (2017a). Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

[Watanabe et al., 2017b] Watanabe, S., Hori, T., Kim, S., Hershey, J. R., and Hayashi, T. (2017b). Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

[Williams and Zipser, 1989] Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

[Xie et al., 2019] Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2019). Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*.

[Xu et al., 2020a] Xu, Q., Likhomanenko, T., Kahn, J., Hannun, A., Synnaeve, G., and Collobert, R. (2020a). Iterative pseudo-labeling for speech recognition. *arXiv preprint arXiv:2005.09267*.

[Xu et al., 2020b] Xu, Q., Likhomanenko, T., Kahn, J., Hannun, A., Synnaeve, G., and Collobert, R. (2020b). Iterative Pseudo-Labeling for Speech Recognition. In *Proc. Interspeech 2020*, pages 1006–1010.

[Zeiler, 2012a] Zeiler, M. D. (2012a). ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

[Zeiler, 2012b] Zeiler, M. D. (2012b). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

[Zeyer et al., 2019] Zeyer, A., Bahar, P., Irie, K., Schlüter, R., and Ney, H. (2019). A comparison of transformer and lstm encoder decoder models for asr. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15. IEEE.

[Zhang et al., 2022] Zhang, Y., Park, D. S., Han, W., Qin, J., Gulati, A., Shor, J., Jansen, A., Xu, Y., Huang, Y., Wang, S., et al. (2022). Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition. *IEEE Journal of Selected Topics in Signal Processing*.

[Zhang et al., 2019] Zhang, Y., Weiss, R. J., Zen, H., Wu, Y., Chen, Z., Skerry-Ryan, R., Jia, Y., Rosenberg, A., and Ramabhadran, B. (2019). Learning to Speak Fluently in a Foreign Language: Multilingual Speech Synthesis and Cross-Language Voice Cloning. In *Proc. Interspeech 2019*, pages 2080–2084.

[Zheng et al., 2014] Zheng, X., Wu, Z., Meng, H., and Cai, L. (2014). Contrastive auto-encoder for phoneme recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2529–2533. IEEE.

[Zhu et al., 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.