10-2021

Data augmentation for automatic speech recognition for low resource languages

Ronit Damania
rjd2551@rit.edu

Data augmentation for automatic speech recognition for low resource languages

by

Ronit Damania

A Thesis Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science

Supervised by

Dr. Christopher Homan

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

October 2021

The thesis "Data augmentation for automatic speech recognition for low resource languages" by Ronit Damania has been examined and approved by the following Examination Committee:

Dedication

This thesis is dedicated to my parents.

Acknowledgments

I would like to thank Dr. Christopher Homan for his continued support and guidance, with invaluable suggestions. I am thankful to Dr. Emily Prud'hommeaux for her advice and general expertise. I want to thank Dr. Raymond Ptucha for being part of my thesis committee and getting me interested in deep learning. Thank Dr. Cory E. Merkel, Mr. Spencer Montan, Mr. Byron D. Behm, and Mr. Daniel W. Adams from NTID's Center on Access Technology (CAT) team for their support.

Abstract

Data augmentation for automatic speech recognition for low resource languages

Ronit Damania

Supervising Professor: Dr. Christopher Homan

In this thesis, we explore several novel data augmentation methods for improving the performance of automatic speech recognition (ASR) on low-resource languages. Using a 100-hour subset of English LibriSpeech to simulate a low-resource setting, we compare the well-known SpecAugment augmentation approach to these new methods, along with several other competitive baselines. We then apply the most promising combinations of models and augmentation methods to three genuinely under-resourced languages using the 40-hour Gujarati, Tamil, Telugu datasets from the 2021 Interspeech Low Resource Automatic Speech Recognition Challenge for Indian Languages. Our data augmentation approaches, coupled with state-of-the-art acoustic model architectures and language models, yield reductions in word error rate over SpecAugment and other competitive baselines for the LibriSpeech-100 dataset, showing a particular advantage over prior models for the "other", more challenging, dev and test sets. Extending this work to the low-resource Indian languages, we see large improvements over the baseline models and results comparable to large multilingual models.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Automatic speech recognition (ASR) has been a fundamental problem in artificial intelligence for decades. Recently, the performance of ASR on high-resource languages has benefited enormously from neural models [17, 2, 6, 15], enabling the integration of ASR for into software and devices used every day by the people who speak these languages. However, the vast majority of the world's languages, even those spoken by tens of millions of people, do not have the quantity of transcribed speech necessary to build ASR models of this caliber, making speech-based applications out of reach for billions of people.

A common approach for learning in under-resourced settings is *data augmentation*. Data augmentation of the acoustic training data via distortion of the speech signal has been found to be particularly useful for ASR. Early work often focused on changes in speaking rate or pitch in order to accommodate variations in speaking style and voice quality and features [24, 12, 42]. More recently, SpecAugment [32] has been shown to be effective at augmenting acoustic training data by warping and zeroing out random regions of the speech spectrum, making models more robust to spectral variation and variability in recording quality. In this paper we explore variants to SpecAugment that alter, rather than eliminate, random regions in the spectrum. We also explore a form of augmentation via concatenation. We demonstrate the utility of these methods in a simulated low-resource setting using a 100-hour subset of English LibriSpeech [31]. We find that some of our augmentation methods outperform SpecAugment, particularly on the more challenging dev and test sets (i.e., those labeled "other" as opposed to "clean"). Combined with a language model, this approach yields WER on both dev and test sets lower than several state-of-the-art architectures for this dataset. We then test the most promising of these methods on three genuinely low-resource datasets – 40 hours each of Gujarati, Tamil, Telugu – using a conformer acoustic model, yielding reductions in WER for all three languages over the Interspeech challenge baselines by 5 to 12 percentage points (a reduction of 17-33%).

1.1 Contributions

The contributions of this thesis are

- Multiplying the region to augment with a random value (MWR).
- Replacing the region to augment with a random value (RWR).
- Input concatenation (IC).
- In addition to conceiving new augmentation methods, we test them on four datasets, three of which are under resourced.

Chapter 2

Related Work

The acoustic model (AM) represents the relation between audio or feature extracted audio with the lexical unit. For large vocabulary continuous speech recognition (LVCSR), the first AMs were statistical in nature, e.g., hidden Markov models with Gaussian mixture models (HMM-GMMs). GMMs were later replaced by deep neural networks (DNNs), which were first used in deep belief networks (DBNs) [19]. Sequence modeling is one of the most popular techniques in ASR, with models like Listen, Attend and Spell (LAS) [6] and Deep Speech [17, 2]. Convolution techniques, like time delay neural networks (TDNNs) [33], are also used. Language model (LM) are trained on text data. It predicts the next unit given the previous unit(s). LM follows a similar trend, starting with statistical models like n-grams [9] and moving to sequence modeling where recurrent neural network (RNNs) [30, 29, 39] are used. And convolution-based methods like gated convolutional networks [10] have also shown promising results.

2.1 ASR pipeline

The speech signal is a $1D$ vector that is sampled typically at 16kHz for ASR. Each observation is typically 16-bit for ASR and represents the amplitude of the signal. The most commonly used features, such as MFCCs, fbanks, or spectrograms, are typically extracted with a 25ms window, and a stride of 10ms. Let O be a sequence of acoustic features, O = $O_1, O_2, ..., O_T$, where $O_i \in \mathbb{R}^d$, where d is the dimension of the features and T is the length of the sequence. Let W be a word sequence. ASR solves the following problem.

$$\mathrm{W}^* = \arg\max_{\mathrm{W}} \mathrm{P}(\mathrm{W}|\mathrm{O})$$

where $\mathrm{P}(\mathrm{W}|\mathrm{O})$ is obtained from the AM and $\mathrm{P}(\mathrm{W})$ is obtained from the LM.

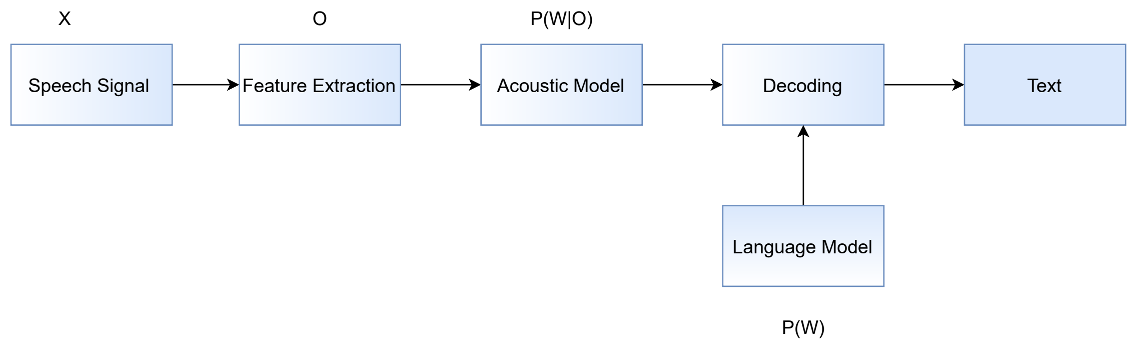

Figure 2.1: **ASR Pipeline.**

2.2 Mel Spectrogram

A spectrogram is a visual way of representing the signal strength, of a speech signal over time at various frequencies present in a particular waveform[1]. To get the spectrogram of the audio signal, it needs to be converted from the time domain to the frequency domain. Windowing functions like Hamming and Hanning [5] are popular techniques for smoothing values. For our experiments, we used frames of size 25ms with a stride of 10ms and a Hanning window. Let $w[n]$ be the window function, M be the window width, $\alpha = 0.46164$ for Hamming, and $\alpha = 0.5$ for Hanning. Then

$$w[n] = (1-\alpha) - \alpha \cos\left(\frac{2\pi n}{M-1}\right) \qquad 0 \leq M \leq 1.$$

The fast Fourier transform (FFT) method does the Fourier transform of the signal. The log of the magnitude of the frequency is taken to get the spectrogram. It has been reported [38, 37] that humans do not perceive sound in a linear way. For instance, it is easier to

[1]`https://pnsn.org/spectrograms/what-is-a-spectrogram`

differentiate between 700Hz and 1200Hz than between 10000Hz and 10500Hz even though the difference between the frequencies in each pair is the same (500Hz). For this, a new scale is created called the mel scale, and the spectrogram is converted to mel scale.

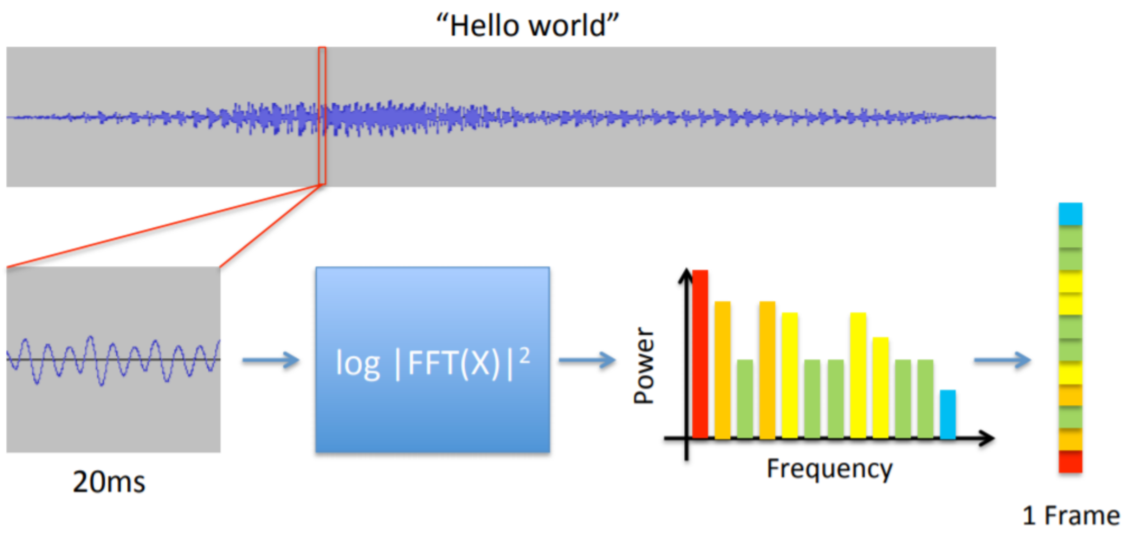

Figure 2.2: Taking a small window of, for example, 20ms and computing the magnitude of the FFT (for mel spectrogram converting into mel scale) to get the frequency information of the local window [7].

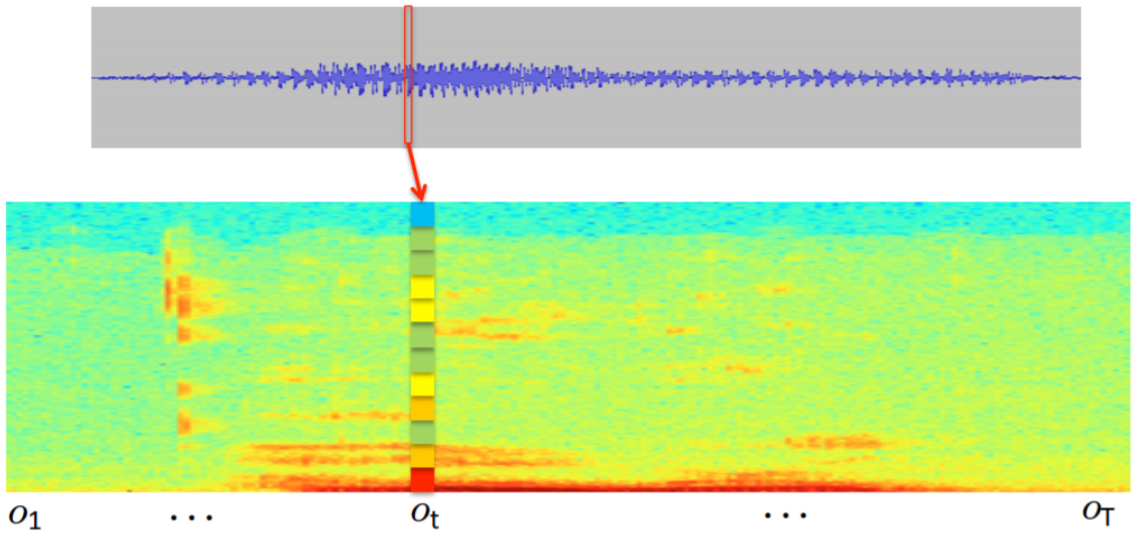

Figure 2.3: Concatenation of adjacent windows to form spectrogram [7].

2.3 Connectionist Temporal Classification (CTC)

Most AM models will provide the likelihood of units for each O_i. However, the length of O does not always equal the length of W (usually length of O is greater than length of W). Methods such as using a sequence to sequence model [4], a sequence to sequence model with attention [6], or a transformer model [40] have all been used to map multiple frames to multiple units. For the models that predict one unit per frame the most common approach has been to use Connectionist Temporal Classification (CTC) [13]. The CTC algorithm is alignment free. It considers all the possible alignments for the given O. Let's consider a naive approach of predicting a unit at each timestamp and remove the repeated unit as shown in the figure below.

O_1	O_2	O_3	O_4	O_5	O_6	feature observation
c	c	a	a	a	t	alignment
c		a			t	output

Table 2.1: Consider six timestamps and each of them is making a prediction of unit and in the end removing the adjacent repeating unit to predict the output 'cat' [16].

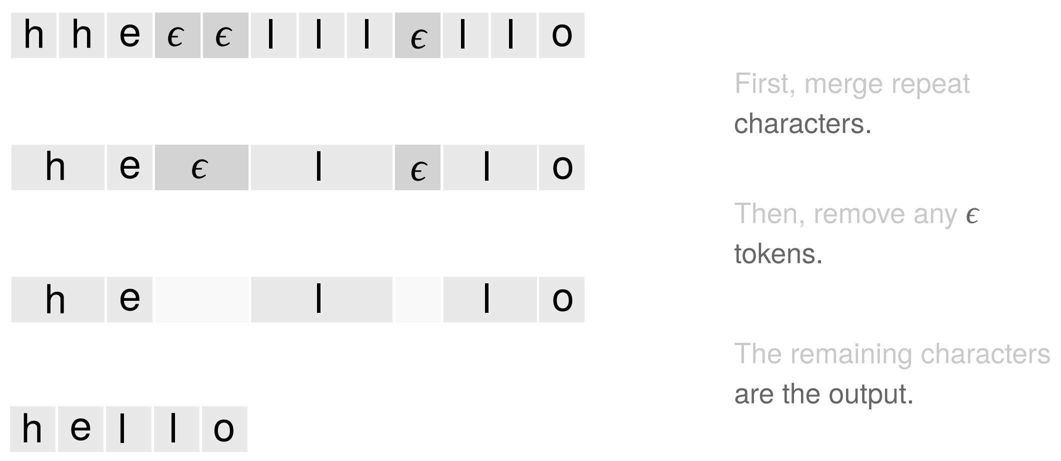
h h e ϵ ϵ l l l ϵ l l o
h e ϵ l ϵ l o
h e l l o
h e l l o
First, merge repeat characters.
Then, remove any ϵ tokens.
The remaining characters are the output.

Figure 2.4: If the output has repeated units then it must have an ϵ between them. This allows CTC to predict 'hello' instead of 'helo' [16].

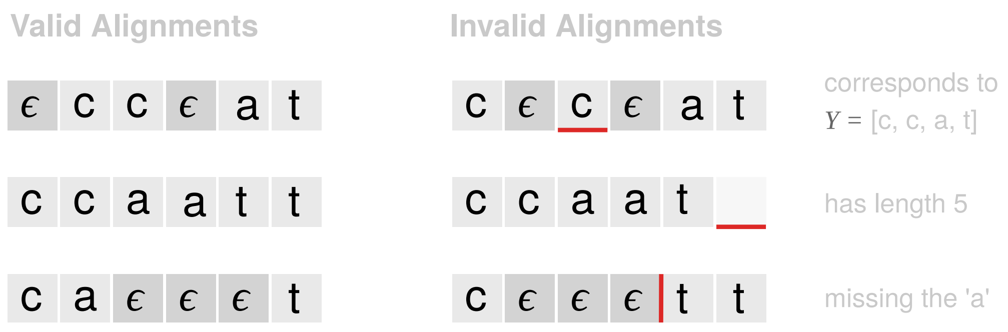

Figure 2.5: For a given output ‘cat’, examples of valid and invalid alignment [16].

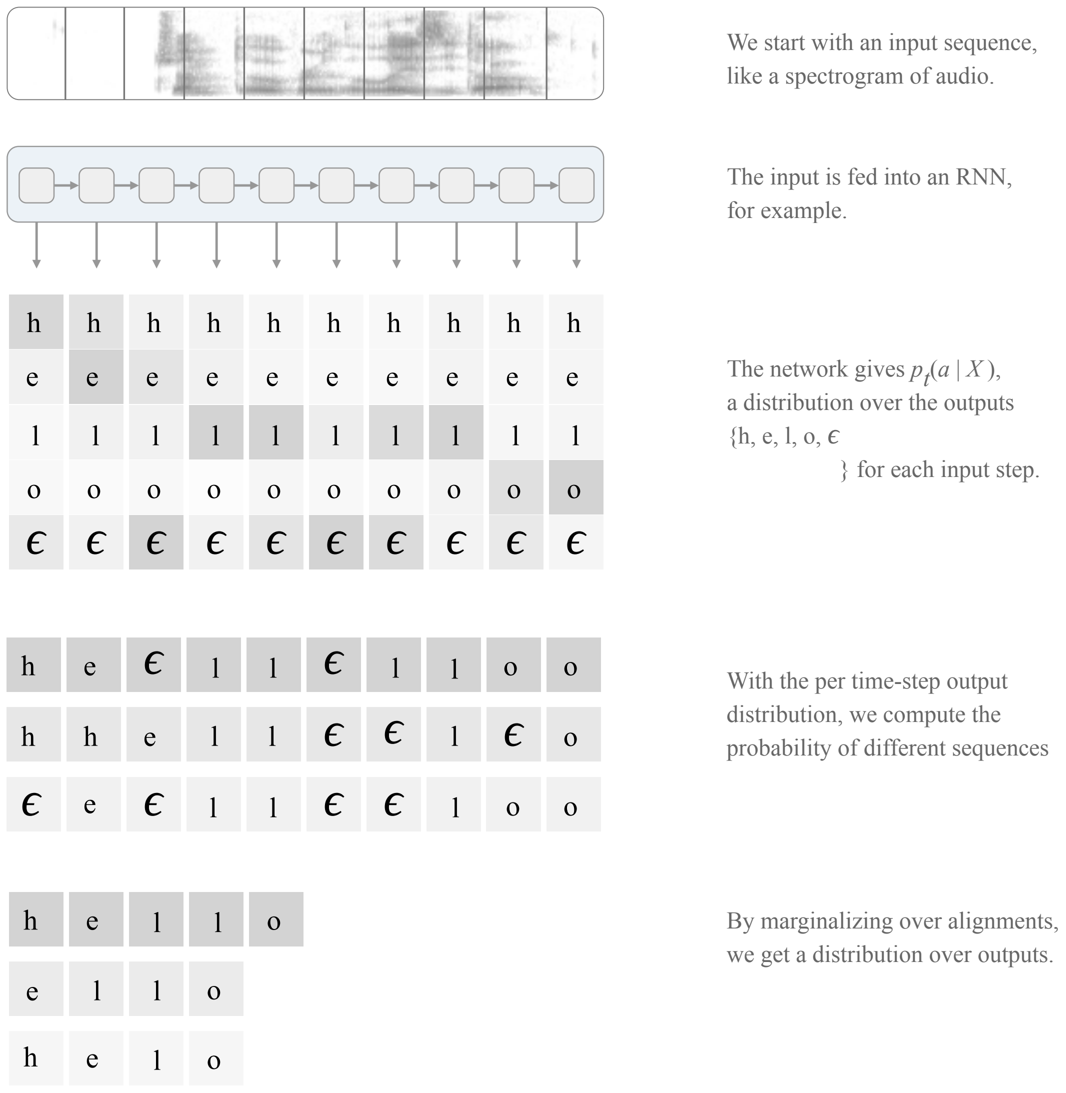
We start with an input sequence, like a spectrogram of audio.
The input is fed into an RNN, for example.
The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.
With the per time-step output distribution, we compute the probability of different sequences
By marginalizing over alignments, we get a distribution over outputs.
h e ϵ l l ϵ l l o o
h h e l l ϵ ϵ l ϵ o
ϵ e ϵ l l ϵ ϵ l o o
h e l l o
e l l o
h e l o

Figure 2.6: **CTC Pipeline:** More grey represents more probability [16].

There are essentially two problems with this approach. The first is that the speech signal can have timestamps where nothing is spoken. Second, it is impossible to identify the word with the repeated adjacent unit since repeated units are getting merged. For example, the alignment [b, b, a, a, t, t, t, l, e] will collapse to ‘batle’ instead of ‘battle.’ To fix this, CTC introduces a new unit to the set of allowed outputs. And is referred to as ϵ. The ϵ unit doesn’t correspond to anything and is simply removed from the output [16].

$$\mathrm{CTC(O, W) = P(W|O)} = \Sigma_{\mathrm{A \in A_{O,W}}} \prod_{\mathrm{t=1}}^{\mathrm{T}} \mathrm{P_t(a_t|O)}$$

Where $\Sigma_{\mathrm{A \in A_{O,W}}}$ marginalizes over the set of valid alignments and $\prod_{\mathrm{t=1}}^{\mathrm{T}} \mathrm{P_t(a_t|O)}$ computes the probability for a single alignment. CTC assumes conditional independence of $\mathrm{a_t}$ with the output at other timestamps given the input x.

2.4 Decoding

The AM will produce a $\mathrm{T} \times \mathrm{C}$ matrix with the probabilities of units, i.e., C at each frame T_i.

$$\mathrm{W^*} = \underset{\mathrm{W}}{\arg\max}\, \mathrm{P(W|O)}$$

The straightforward approach is to use a greedy search algorithm and pick the unit which has the maximum probability for each frame and remove the repeated units and ϵ unit to get the transcript.

$$\mathrm{A^*} = \underset{\mathrm{A}}{\arg\max} \prod_{t=1}^{\mathrm{T}} P_t(A_t|\mathrm{O})$$

This method is not optimal for ASR since it doesn’t consider those multiple alignments that can lead to the same output.

Consider the alignment [b, b], which has a higher probability than [a, a], [ϵ, a], or [a, ϵ] has individually. But the sum of their probabilities is greater than that of [b, b]. In this case, the greedy approach will produce the output ‘b’ while the actual output should have been ’a.’ To fix this the algorithm needs to take into account that [a, a], [ϵ, a], [a, ϵ] correspond to the same output. For this, a modified version of beam search is used. Beam search creates a new set of hypotheses by considering all combinations of units from the previous set of saved hypotheses, and then saves the k most likely outputs. Here, 'k' is the *beamwidth hyperparameter*.

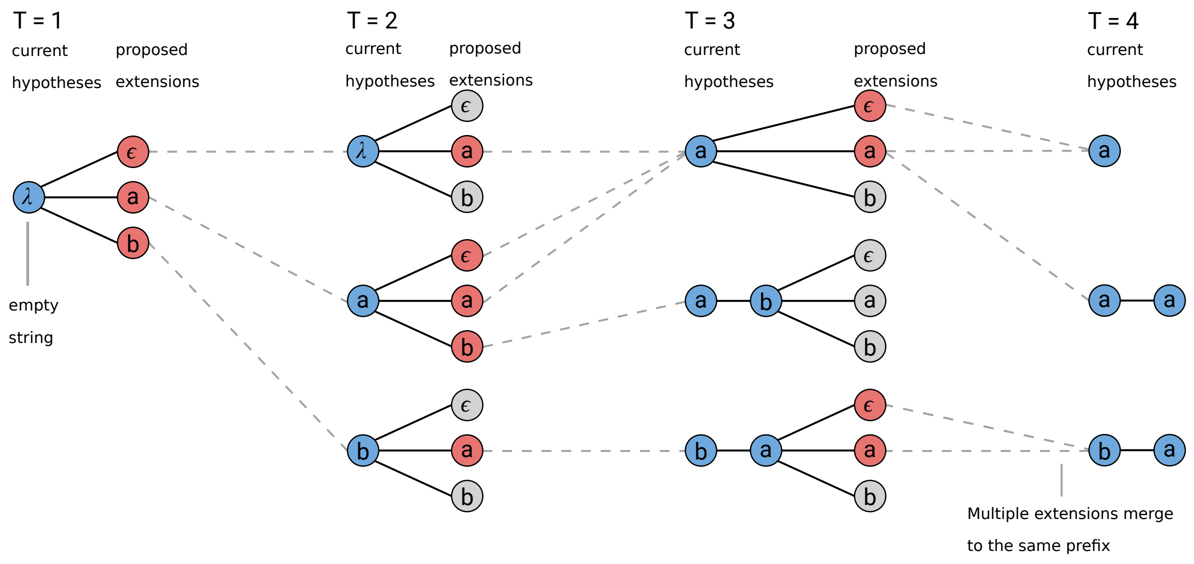

Figure 2.7: The CTC beam search algorithm with an output alphabet {ϵ, a, b} and a beam size of three. Figure from [16].

In modified beam search, instead of storing the alignments, the output prefix is stored after removing the repeated units and ϵ. At each step of the search, it accumulates scores for a given prefix based on all the alignments which map to it. For example, in the figure above at $T = 3$, all the alignments of 'a' , i.e., [a, a], [a, ϵ] and [ϵ, a], are taken. Now the LM has been known to improve the accuracy of the prediction to include it following equation is used. This decoding method was used in this paper [28].

$$\mathrm{W}^{*} = \arg\max_{\mathrm{W}} \mathrm{P}(\mathrm{W}|\mathrm{O}) * \mathrm{P}(\mathrm{W})^{\alpha} * L(\mathrm{W})^{\beta}$$

Where $\mathrm{P}(\mathrm{W})$ is the LM probability and $L(\mathrm{W})$ is the word insertion bonus. It is the length of W in the language model. It could be the number of characters or words in W based on what kind of the LM is used. The LM use only when a new character or word is added to the prefix. This favors the shorter prefixes, since there are fewer updates from LM, to counter this $L(\mathrm{W})$ is used, The α and β are hyperparameters that can be set by cross-validation.

2.5 SpecAugment

SpecAugment [32] is an augmentation technique that is applied directly to input features of the neural network. It uses time warping and time and frequency masking. Given a log mel spectrogram with τ time steps, it views it as an image where the frequency axis is vertical and the time axis is horizontal. In time warping, a random point along the horizontal line passing from the center of the image from the time steps (W, τ - W) is to be warped either left or right by a distance w chosen uniform distribution from 0 to the *time warping parameter* W along that line. In frequency masking, f consecutive mel frequency channels $[f_0, f_0 + f)$ are masked, where first f is selected uniformly randomly from 0 to *frequency masking parameter* F, and f_0 is chosen from [0, v - f), where v is the number of mel frequency channels. In time masking, t consecutive time steps $[t_0, t_0 + t)$ are masked, where first t is selected uniformly randomly from 0 to *time masking parameter* T, and t_0 is selected from [0, τ - t) [32]. The log mel spectrograms are normalized to have zero mean value, and thus setting the mask value as zero is equivalent to setting it as mean value.

Table 2.2: Augmentation parameters for policies. m_F and m_T represents number of frequency and time masks applied [32].

Policy	W	F	m_F	T	m_T
None	0	0	-	0	-
LB	80	27	1	100	1
LD	80	27	2	100	2

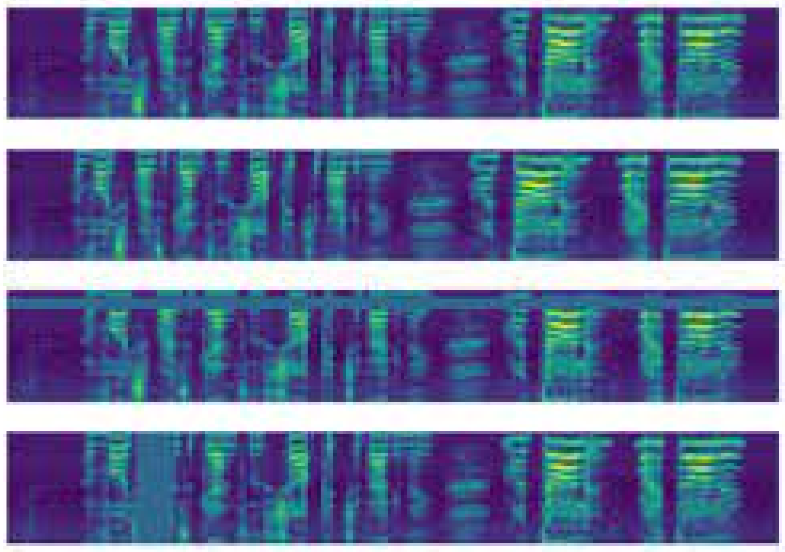

Figure 2.8: The images above show a base input, which is a log mel spectrogram with various types of augmentation. From top to bottom, no augmentation, time warp, frequency masking, and time masking were applied to base input, respectively [32].

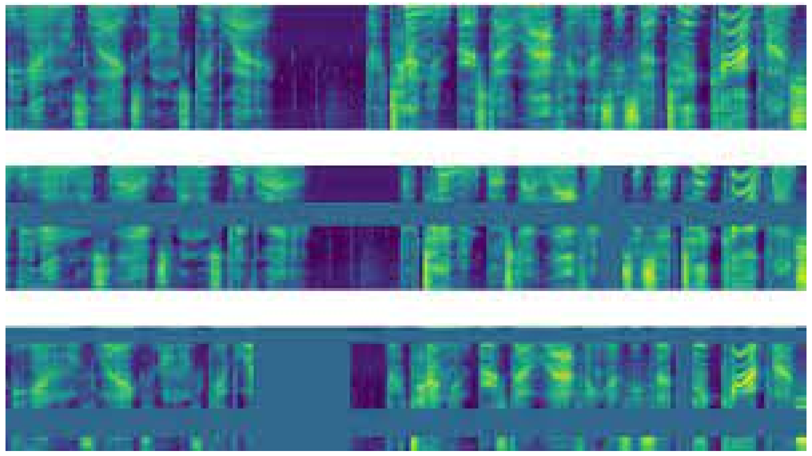

Figure 2.9: The images above show a base input, which is a log mel spectrogram with a couple of augmentation policies applied to it. From top to bottom, no augmentation, LB, and LD as described in Table 2.2 [32].

2.6 Transformer

The transformer [40] model follows an encoder-decoder structure. The encoder maps an input sequence $(x_1, x_2, ..., x_n)$ to a representation z=$(z_1, z_2, ..., z_n)$ and the decoder generates sequence of symbols $(y_1, y_2, ..., y_m)$ one at a time given the z from the encoder and the previous generated symbols as an input to the decoder to generate next.

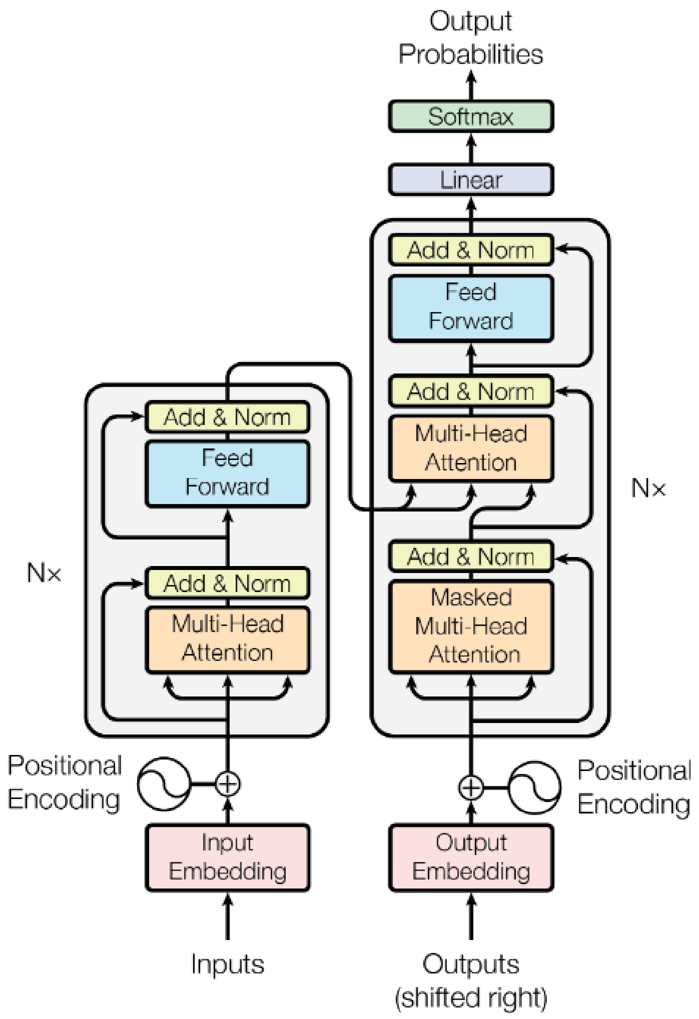

Figure 2.10: The Transformer - model architecture [40].

The encoder consists of a stack of layers. In each layer, there are two sub-layers. The first is a multi-head attention and the second is a feed forward layer. Between sub-layers there is a residual connection [18] followed by layer normalization [3]. All of the sub-layers and embedding layers produces an output of dimension d_{model} to facilitate residual connections. The decoder layer, in addition to the sub-layers in encoder stack contains an additional sub-layer to perform multi-head attention over the output of encoder stack.

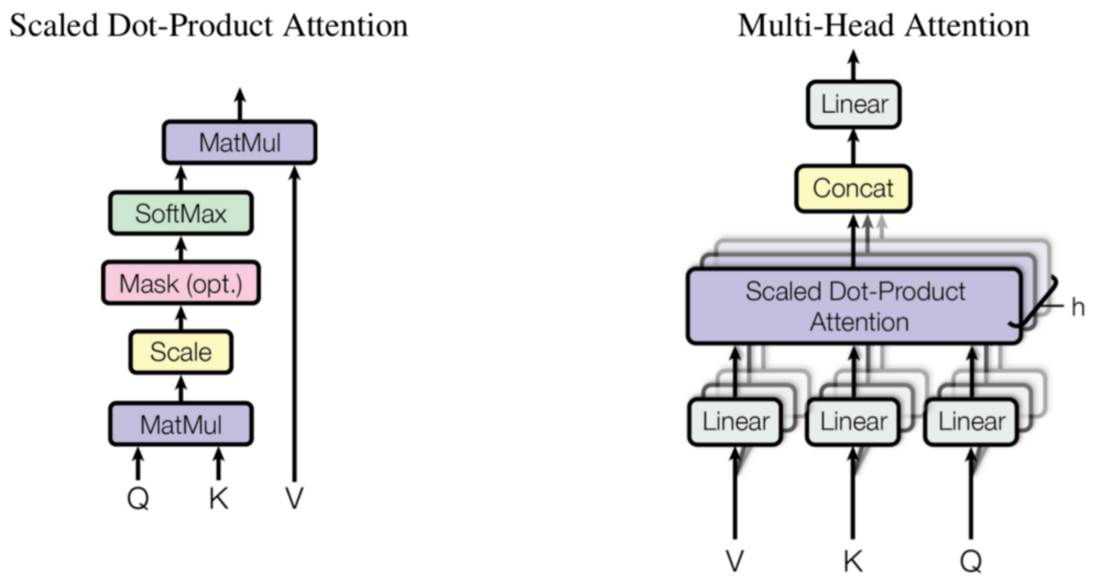

Figure 2.11: (left) Scaled Dot-Product Attention, (Right) Multi-Head Attention [40].

Attention function is described as a mapping query and a set of key-value pairs to an output, where query, keys, values, and output are all vectors. For “Scaled Dot-Product,” the input consists of queries and keys of dimension d_k and values of dimension d_v. The computation is done by taking the dot product of the query with all the keys, dividing it by $\sqrt{d_k}$, taking softmax of it, and finally taking the dot product with the values. The attention is calculated on a set of queries simultaneously, mapped in the matrix Q; similarly, keys and values are mapped to matrix K and V respectively.

$$\mathrm{Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V}$$

The queries, keys, and values are of d_{model} dimensional in a single attention mechanism. In multi-head attention, queries, keys and values are linearly projected h times in the dimensional d_k, d_k, d_v respectively. Apply attention function to each of the linear projections of the queries, keys, and values in parallel to get d_v dimensional output. As shown in the figure above, apply linear projection to concatenated output.

$$\mathrm{MutiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O}$$

where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ [40].

2.7 Conformer

CNNs are good at capturing local features, and transformers are good at capturing global interaction. The convolution-augmented transformer, i.e., conformer, combines both CNNs and transformers to model both global and local dependencies of the audio data for speech recognition [15].

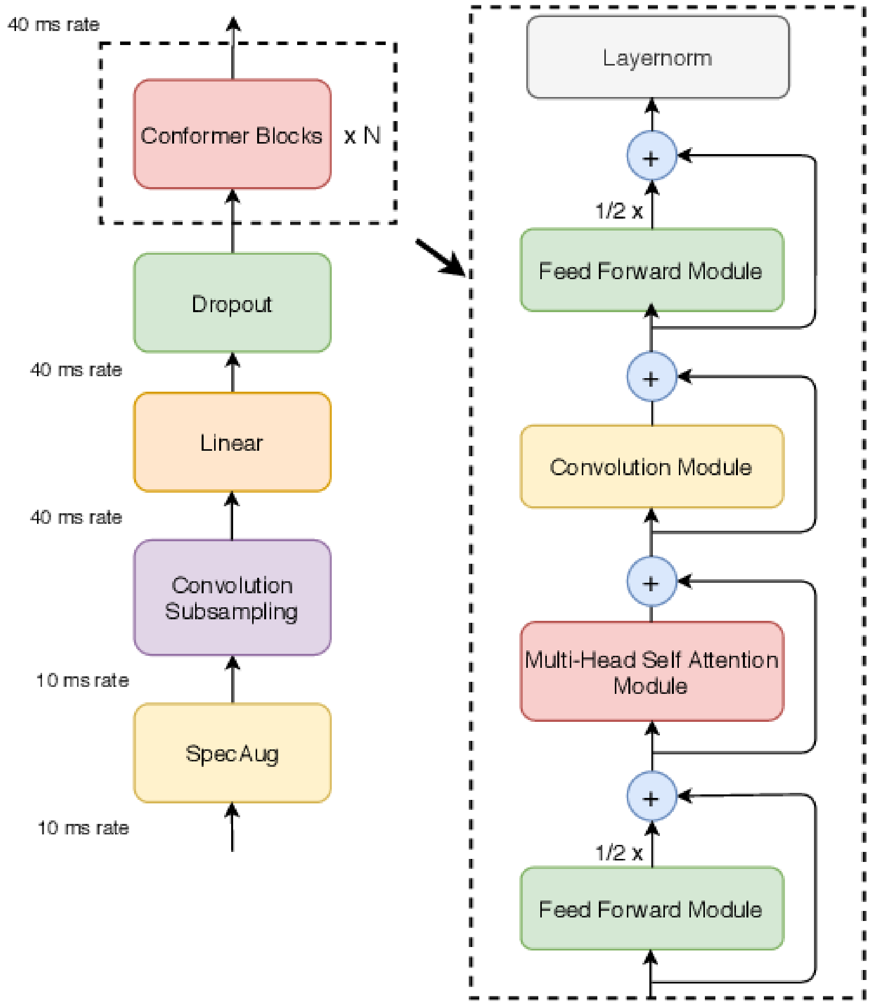

Figure 2.12: **Conformer encoder model architecture.** Conformer comprises of two macaron-like feed-forward layers with half-step residual connections sandwiching the multi-headed self-attention and convolution modules. This is followed by a post layernorm [15].

The conformer encoder does convolution subsampling followed by a number of conformer blocks. A conformer block consists of four modules that are stack together. It starts with feed-forward layer followed by multi-head self attention, a convolution module and a second feed forward network.

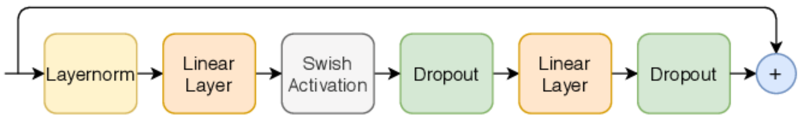

Figure 2.13: **Feed forward module.** It starts with layernorm, the first linear layer uses an expansion factor of 4, second linear layer projects it back to the original dimensions [15].

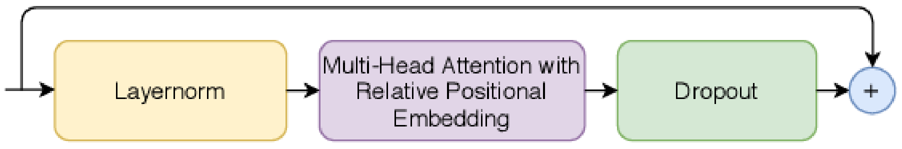

Figure 2.14: **Multi-Headed self-attention module.** It uses multi-headed self-attention with relative positional embedding in a pre-norm residual unit [15].

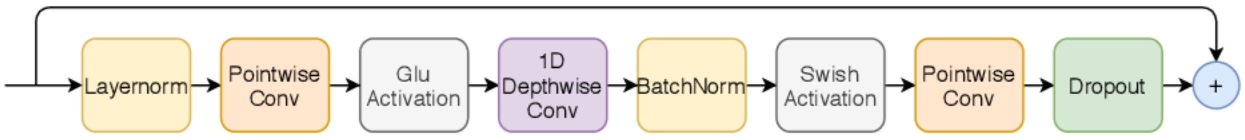

Figure 2.15: **Convolution module.** The convolution module contains a pointwise convolution with an expansion factor of 2 projecting the number of channels with a GLU activation layer, followed by a 1-D depthwise convolution. The 1-D depthwise conv is followed by batchnorm and then a swish activation layer [15].

For decoder it uses single-LSTM-layer.

2.8 Joint CTC-Attention based end-to-end speech recognition using multi-task learning

For end-to-end speech recognition, primarily two approaches are popularly used. One is CTC, and the other is the attention-based approach. CTC predicts output units for each audio frame assuming conditional independence. The attention-based model directly learns the unit sequence from the input. At each output time step, the model predicts the unit based on the input and the history of the target character. This method does not do well for noisy input since alignment is difficult. Also, the model is hard to learn from scratch because of the the potential for misalignment for long input sequences.

This method uses both CTC as well attention mechanism which improves the alignment and the performance along with speeding up the learning because of fast convergence.

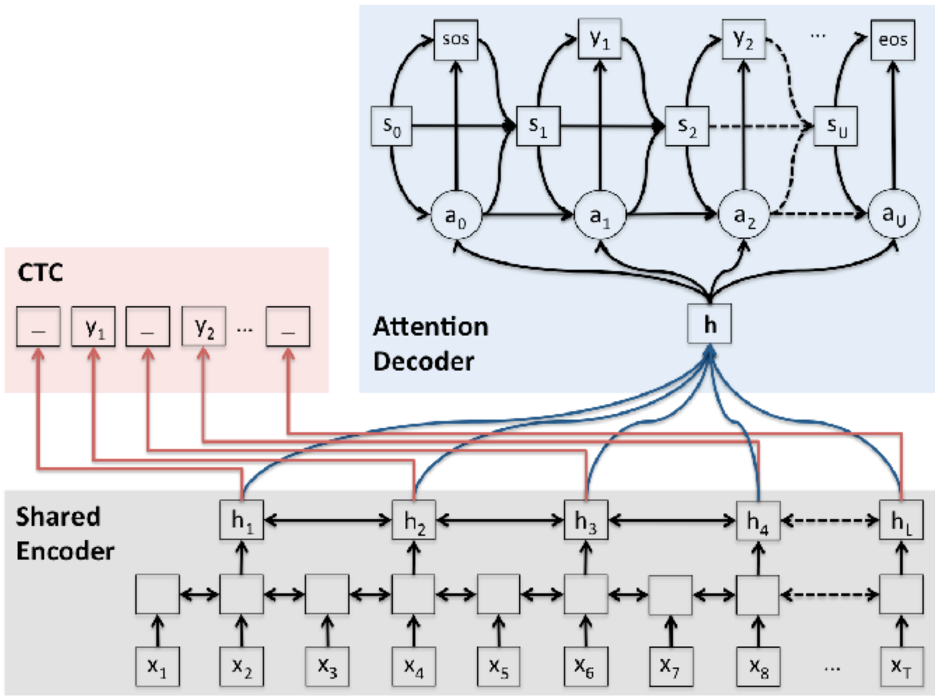

Figure 2.16: The joint CTC-attention have a shared encoder which transforms input x to higher dimension representation h which is pass to CTC and attention decoder to predict the output labels y [23].

The model uses multitask learning (MTL) with a CTC objective as the auxiliary task to train the attention based encoder-decoder model.The objective is represented as follows by using both CTC and attention objective.

$$\mathcal{L}_{MTL} = \lambda \mathcal{L}_{CTC} + (1 - \lambda)\mathcal{L}_{Attention}$$

Where $\mathcal{L}$ is loss and λ is a tunable parameter $\lambda : 0 \leq \lambda \leq 1$.

Chapter 3

Data

3.1 LibriSpeech

In order to simulate a low-resource setting while still having a significant body of prior work from which to gather competitive baselines, we used a 100-hour subset of the English LibriSpeech corpus [31]. LibriSpeech was collected from a corpus of audiobooks that are part of the libriVox project[1]. The creators of LibriSpeech separated the dev and test data into two categories, *clean* and *other*, where the audio in the latter category is drawn from speakers whose recordings yielded higher WER in the original baseline system, suggesting that these recordings are more challenging. The 100-hour subset of LibriSpeech provides a reasonable surrogate for truly under-resources languages.

Table 3.1: Data subsets in LibriSpeech [31].

subset	hours	per-spk minutes	female spkrs	male spkrs	total spkrs
dev-clean	5.4	8	20	20	40
test-clean	5.4	8	20	20	40
dev-other	5.3	10	16	17	33
test-other	5.1	10	17	16	33
train-clean-100	100.6	25	125	126	251
train-clean-360	363.6	25	439	482	921
train-other-500	496.7	30	564	602	1166

The LibriSpeech train-clean-100 corpus is widely used in the research community, enabling us to compare our system to four competitive baselines reported in prior work. These

[1]`https://librivox.org/`

include: (1) Kaldi's [34] best-performing hybrid DNN/HMM model [41] with a 4-gram language model (LM); (2) RWTH Aachen's [27] DNN/HMM hybrid model that uses a bi-directional LSTM [20, 14] of 6 layers with 1000 units for backward and forward directions each; (3) a direct-to-word CTC sequence model [8], which uses a transformer-based acoustic model (AM) with a CTC objective and 4-gram LM; additionally for augmentation, it uses SpecAugment [32]; (4) an end-to-end model [26] that uses a transformer-based acoustic model and a recurrent neural network with four LSTM layers with 2048 units each as a LM; the augmentation used in this particular model involves speed perturbation with perturbation factors of 0.9 and 1.1, along with SpecAugment [32].

3.2 Low resource Indian languages

We consider three small, monolingual datasets for Gujarati, Tamil, Telugu from the Multilingual and Code-Switching (MUCS) ASR Challenges at Interspeech 2021 [11]. The audio is a combination of conversational speech and read speech. There are three subsets in the data splits for each language: *train*, *test*, and *blind*, which we use for training, validation, and testing, respectively. Each training set contains 40 hours of training data. The *train* and *test* sets are sampled at 16kHz, while the *blind* set is sampled at 8kHz. In addition, 34.1%, 23.8% and 29.0% of the blind data chosen at random from Gujarati, Tamil and Telugu, respectively, is modified with speed perturbation and/or noise. Table 3.2 provides more information about the data sets for these three languages.

Table 3.2: Description of data provided (number of unique sentences (Uniq sent) and number of speakers(Spkrs)). The audio files in all the languages consist of single-channel and are encoded in 16-bit [11].

	Gujarati			Tamil			Telugu		
	Train	Test	Blind	Train	Test	Blind	Train	Test	Blind
Size(hrs)	40	5	5.26	40	5	4.41	40	5	4.39
Uniq sent	20257	3069	3419	30329	3060	2584	34176	2997	2506
Spkrs	94	15	18	448	118	118	464	129	129

The baseline results appears in Table 3.2 [11]. The ASR model uses a sequence-trained time-delay neural network (TDNN) architecture optimized using the lattice-free MMI objective function [35]. The architecture consists of an AM with 6 TDNN blocks, each of dimension 512 and a 3-gram LM.

Table 3.3: Baseline WER(%) [11].

Gujarati	Tamil	Telugu
25.98	35.82	29.35

Chapter 4

Methodology

4.1 Model

In the experiments a sequence to sequence AM [21] is used. In the encoder, there are 12 conformer blocks with 8 attention heads and 512 encoder dimensions. The decoder consists of 6 transformer blocks with 2048 linear units, 8 attention head, and a dropout of 0.1. The model does multitask learning [23] with the CTC weight of 0.3 and attention weight of 0.7. An Adam optimizer [36] is used with initial learning rate 0.0025, β_1 as 0.9, β_2 as 0.99 and warmup steps of 40000. Each model is trained for 150 epochs. The text data is tokenized by SentencePiece byte-pair-encoding [25] with 5000 vocabulary size for librispeech data and 200 for Gujarati, Tamil, and Telugu each. Pre-trained transformer LM [21] which is trained on text data from entire librispeech train data and external data[1] is used for librispeech. For the language model of Indian languages, we use RNNLM with 4 LSTM layers and 2048 nodes on each layer for each language. All the experiments are conducted on the Rochester Institute of Technology Research Computing cluster [1]. The inference is done in two ways: one is with the best performing model (BEST) on the validation set, the other is with an AVGn [40, 22] model, whose parameters are formed by taking the average of the parameter of top-n performing models on the validation set, where n is the hyperparameter. For example, we train a model with n as 3 for ten epochs, and during the training, top 3 performance on the validation set are at epoch 3, 7, and 8. At the end of the training AVG3 model is formed by taking the parameter average of the model at epoch 3, 7, and 8. In the

[1] `http://www.openslr.org/resources/11/librispeech-lm-norm.txt.gz`

experiments in which we use speed perturbation (SP) [24], the perturbation factors are 0.9 and 1.1 to increase the training data threefold.

4.2 Augmentation

We introduce three novel data augmentation methods:

- Multiplying the region to augment with a random value (MWR).
- Replacing the region to augment with a random value (RWR).
- Input concatenation

Section 4.2.1, 4.2.2, and 4.2.3 explore those in more detail. We select the region to augment (RTA) for both time and frequency axis of the log mel spectrogram (input) for MWR and RWR in the same way as SpecAugment [32] does for time and frequency masking. SpecAugment mask the region to augment with zero, we explore different options with MWR and RWR. In our experiments, we use the parameters F as 30, T as 40, m_T and m_F as 2 each, for the region to augment. We do not use time warping since it will be difficult to compare the results because of its non-deterministic nature.

4.2.1 Multiplying the region to augment with a random value

After masking, RTA doesn't correlate to the RTA before masking. In MWR, we multiply the RTA of the input with a value m, which is picked uniformly randomly from the range (a, b) for each utterances, where a and b are MWR parameters. So RTA after MWR is a factor of RTA before it as shown in the Figure 4.1.

4.2.2 Replacing the region to augment with a random value

SpecAugment [32] uses a constant masking value zero for the RTA of the input. To help the model regularize better, for masking, we chose a masking value r uniformly randomly from the range **(min(input features of the batch), max(input features of the batch))** for each axis as shown in Figure 4.1. There are two ways in which we can do this. The first is to have the same r for all the utterances in the batch, which is RWR per batch (RWRB), and the second way is to select r for each utterance, which is RWR per utterance (RWRU).

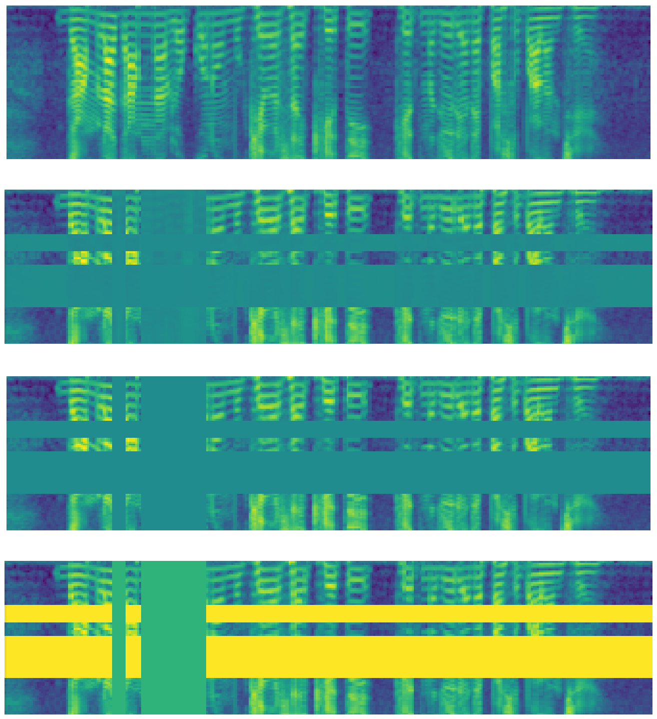

Figure 4.1: The images above show a base input, a log mel spectrogram with different augmentation applied to it. From top to bottom, no augmentation, MWR where the value to multiply was selected uniformly randomly from $(-0.5, 0.5)$ and came -0.2048 for frequency axis and -0.3085 for time axis, specAugment [32] where masked regions value is 0, RWRU/RWRB, where the value to replace is selected uniformly randomly from (-4.886, 6.209) which is the minimum and maximum value of the audio, for frequency axis the value is 5.2457. For the time axis, it is 1.6975.

4.2.3 Input concatenation

In input concatenation, we concatenate two inputs and their corresponding transcripts. For a given batch, we create an array of random integers ($randomInt$) whose length equals the length(batch), and we pick elements of the array from the range [0, length(batch) -1] with replacement. Given a batch array ($batch$), for index i, we concatenate $batch[i]$ with $batch[randomInt[i]]$. Input concatenation can help the AM generalize because AM has to possibly adapt to a couple of speakers in the input, who could have variability in accent, age, gender, etc. We experiment with applying input concatenation to a certain percentage of the element in the batch. We call it input concatenation percentage (ICP). We explore ICP with percentage 0.25, 0.5, 0.75, and 1. The results are in Section 5.1.3.

Chapter 5

Results and Discussion

5.1 LibriSpeech

All the experiments in this section uses train-clean-100, which is a 100 hours subset of the librispeech dataset described in section 3.1. Inference is done with beam size 60, CTC weight 0.4, and in the experiments with LM, the LM weight is 0.6. Table 5.1 shows that

Table 5.1: Baseline results WER(%).

Model	dev-clean	dev-other	test-clean	test-other
BEST	13.4	34.9	13.8	35.9
AVG10	9.5	28.7	10.0	29.4

the AVG10 model is performs significantly better than the BEST model.

5.1.1 Multiplying the region to augment with a random value

MWR is performing than baseline. The general trend is, as the range increases, the performance decreases.

Table 5.2: **Results of MWR:** WER(%) with the BEST model, the first column represents the range (a, b) from which we chose the random value m to multiply the RTA.

Range	dev-clean	dev-other	test-clean	test-other
(-0.1, 0.1)	11.7	29.5	12.0	30.3
(-0.25, 0.25)	12.1	30.6	12.3	31.2
(-0.5, 0.5)	12.3	31.0	12.6	32.5
(-1, 1)	12.4	31.6	12.5	32.8

Table 5.3: **Results of MWR:** WER(%) with the AVG10 model, the first column represents the range (a, b) from which we chose the random value m to multiply the RTA.

Range	dev-clean	dev-other	test-clean	test-other
(-0.1, 0.1)	8.6	23.5	8.8	24.2
(-0.25, 0.25)	8.8	23.9	9.1	24.8
(-0.5, 0.5)	8.5	24.6	9.1	25.2
(-1, 1)	8.7	25.2	9.2	26.1

5.1.2 SpecAugment and replace with random

Comparing the SpecAugment method with RWRB and RWRU. Tables 5.4 and 5.5 show both RWRB and RWRU perform better than SpecAugment. For the AVG10 model, the performance of all of the methods is close, with RWRB performing the best.

Table 5.4: Results: WER(%) of different augmentation methods with the BEST model.

Methods	dev-clean	dev-other	test-clean	test-other
SpecAugment	10.2	25.1	10.7	25.3
RWRB	10.5	24.5	10.6	25.1
RWRU	10.0	24.1	10.2	25.0

Table 5.5: Results: WER(%) of different augmentation methods with the AVG10 model.

Methods	dev-clean	dev-other	test-clean	test-other
SpecAugment	7.4	20.0	7.9	20.5
RWRB	7.5	19.7	**7.8**	**20.1**
RWRU	7.5	20.0	7.8	20.3

Since the RWRB method gives the best results, we perform further experiments with different n in AVGn. AVG1 means taking the average of the top 1 model, i.e., AVG1 is the same as the BEST model. Table 5.10 shows that increasing n improves the performance.

5.1.3 Input concatenation

Among the ICPs explored, an ICP of 0.50 gives the best result.

Table 5.6: **Results of ICPs:** WER(%) with the BEST model, the value in the first column is the percentage of input concatenation in a batch.

	dev-clean	dev-other	test-clean	test-other
ICP 0.25	13.3	35.4	13.6	37.1
ICP 0.50	13.1	34.9	13.6	36.0
ICP 0.75	13.4	35.2	13.8	36.0
ICP 1	13.7	34.8	14.0	35.4

Table 5.7: **Results of ICPs:** WER(%) with the AVG10 model, the value in the first column represent the percentage of input concatenation in a batch.

	dev-clean	dev-other	test-clean	test-other
ICP 0.25	9.5	28.8	10.0	29.7
ICP 0.50	9.6	28.7	10.1	29.0
ICP 0.75	9.3	28.7	10.1	29.6
ICP 1	9.7	28.7	10.0	29.3

5.1.4 RWRU with input concatenation

We conducted the experiments of RWRU with ICPs. It follows a similar trend as observed in Tables 5.6 and 5.7; RWRU with ICP 0.50 performs the best for the BEST model and the same as RWRU for the AVG10 model as shown in Table 5.8 and 5.9. No input concatenation means ICP 0.

Table 5.8: **Results of RWRU + ICPs:** WER(%) on the BEST model, the value in the first column represents the percentage of input concatenation in a batch.

Method	dev-clean	dev-other	test-clean	test-other
RWRU + ICP 0	10.0	24.1	10.2	25.0
RWRU + ICP 0.25	9.9	24.6	10.1	25.4
RWRU + ICP 0.50	**9.8**	**24.6**	**9.8**	**25.1**
RWRU + ICP 0.75	10.1	24.6	10.2	25.8
RWRU + ICP 1	11.2	25.3	11.4	26.0

Table 5.9: **Results of RWRU + ICPs:** WER(%) on the AVG10 model, the value in the first column represents the percentage of input concatenation in a batch.

Method	dev-clean	dev-other	test-clean	test-other
RWRU + ICP 0	7.5	20.0	7.8	20.3
RWRU + ICP 0.25	7.4	19.9	7.8	20.7
RWRU + ICP 0.50	7.6	20.0	7.8	20.3
RWRU + ICP 0.75	7.5	19.7	7.8	20.4
RWRU + ICP 1	7.6	19.9	7.9	20.4

5.1.5 Replace with random per batch

Since RWRB gives the best result on the AVG10 model, we conduct more experiments, including the different n for AVGn, using LM and speed perturbations.

Table 5.10: **Results:** WER(%) of RWRB method on the AVGn model.

Model	dev-clean	dev-other	test-clean	test-other
BEST/AVG1	10.5	24.5	10.6	25.1
AVG5	7.7	20.0	8.1	20.6
AVG10	7.5	19.7	7.8	20.1
AVG20	7.3	19.5	7.5	20.0
AVG20 + LM	5.1	14.0	5.2	14.4
AVG20 + LM +SP	**4.6**	**13.2**	**5.1**	**13.1**

5.1.6 Comparison with other methods

We compare our results with four competitive baselines reported in prior work. These include: (1) Kaldi's [34] best-performing hybrid DNN/HMM model [41] with a 4-gram language model (LM); (2) RWTH Aachen's [27] DNN/HMM hybrid model that uses a bi-directional LSTM [20, 14] of 6 layers with 1000 units for backward and forward directions each; (3) a direct-to-word CTC sequence model [8], which uses a transformer-based acoustic model (AM) with a CTC objective and 4-gram LM; additionally for augmentation, it uses SpecAugment [32]; (4) an end-to-end model [26] that uses a transformer-based acoustic model and a recurrent neural network with four LSTM layers with 2048 units each as a LM; the augmentation used in this particular model involves speed perturbation with perturbation factors of 0.9 and 1.1, along with SpecAugment [32].

Table 5.11: Results: WER(%) of different methods where training data used for acoustic model is train-clean-100.

Model	dev-clean	dev-other	test-clean	test-other
Kaldi[a]	5.9	20.4	6.6	22.5
word-level CTC [8]	6.3	19.1	6.8	19.4
RWTH [27]	5.0	19.5	5.8	18.6
End to End [26]	5.8	16.6	7.0	17.0
AVG20 +RWRB + LM +SP	**4.6**	**13.2**	**5.1**	**13.1**

[a]`https://github.com/kaldi-asr/kaldi/blob/master/egs/librispeech/s5/RESULTS`

5.2 Low resource Indian languages

The table 5.12 shows the performance of Indian languages Gujarati, Tamil, and Telugu with different augmentation methods and models. We use SP for all of our experiments in this section. While using BEST and AVG10 models, RWRB gives better performance. While using AVG10 with LM, RWRU + ICP 0.5 provides a better result, which is the best result overall.

Table 5.12: Results: WER(%) of Gujarati, Tamil and Telugu languages with different methods, inference model and LM. Baseline result in first row from [11].

Model	Method	Gujarati	Tamil	Telugu	Average
Baseline[11]		25.98	35.82	29.35	30.38
BEST	SpecAugment	30.3	29.0	32.0	30.4
	RWRB	29.1	29.0	31.6	29.9
	RWRU	29.9	28.9	32.4	30.4
	RWRU + ICP 0.5	30.7	28.9	31.2	30.3
AVG10	SpecAugment	25.3	24.9	26.5	25.6
	RWRB	25.0	24.7	26.9	25.5
	RWRU	25.1	24.9	26.7	25.6
	RWRU+ICP 0.5	26.3	25.9	27.7	26.6
AVG10 + LM	SpecAugment	21.9	23.9	24.3	23.4
	RWRB	22.3	23.9	**24.2**	23.5
	RWRU	22.0	24.2	**24.2**	23.5
	RWRU+ICP 0.5	**20.9**	**23.7**	24.3	**23.0**

Chapter 6

Conclusions

This thesis aimed to study the impact of data augmentation techniques in ASR, specifically for low resource datasets. We explored the performance of start of the art AM which contains conformer and transformer blocks, LMs like transformer LM and RNNLM, and data augmentation like specAugment and speed perturbation. We studied the new data augmentation techniques like MWR, RWRB, RWRU, and ICP and it's performance on datasets. We looked into different ranges for MWR. We explored RWRB and RWRU methods and studied the performance in comparison to SpecAugment. We looked into different percentage values for input concatenation. We also explored RWRU in conjunction with best performing ICP, i.e., 0.5. Additionally, we looked into different inference models like BEST and AVGn and the impact of increasing the value of n. On the libriSpeech train-clean-100 subset, we observe that both RWRB and RWRU perform better than SpecAugment on the more difficult test-other subset. On low resource Indian languages, SpecAugment, RWRB, and RWRU are close to each other for the AVG10 model. The inclusion of ICP 0.5 along with LM on RWRU gives us the best results.

6.1 Future Work

- We experimented with applying masking techniques (SpecAugment, RWRB, and RWRU) and MWR on RTA. Additionally, we can replace each value in RTA with a uniformly random value selected from the range **[min(input features of the batch), max(input features of the batch)]**, which can act as random noise in the input.

- Since the AVGn model has given us promising results, we can do additional analysis using weighted average or applying softmax, where weights are the model's accuracy on the validation set.

- One can explore the impact of these augmentation techniques on more datasets, on mono and multi-lingual models.

- Further investigation into different shapes of masking like the circle, triangle, and rectangle (current) along with MTL of shape prediction, given that it uses only one shape per utterance.

Bibliography

[1] Rochester institute of technology. `https://doi.org/10.34788/0S3G-QD15`. 2021.

[2] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595, 2015.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[5] R. B. Blackman and J. W. Tukey. The measurement of power spectra from the point of view of communications engineering — part i. *The Bell System Technical Journal*, 37(1):185–282, 1958.

[6] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell, 2015.

[7] Adam Coates and Vinay Rao. Speech recognition and deep learning. `https://cs.stanford.edu/~acoates/ba_dls_speech2016.pdf`.

[8] Ronan Collobert, Awni Hannun, and Gabriel Synnaeve. Word-level speech recognition with a letter to word encoder, 2020.

[9] James H. Martin Dan Jurafsky. Speech and language processing.

[10] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083, 2016.

[11] Anuj Diwan, Rakesh Vaideeswaran, Sanket Shah, Ankita Singh, Srinivasa Raghavan K. M., Shreya Khare, Vinit Unni, Saurabh Vyas, Akash Rajpuria, Chiranjeevi Yarra, Ashish R. Mittal, Prasanta Kumar Ghosh, Preethi Jyothi, Kalika Bali, Vivek Seshadri, Sunayana Sitaram, Samarth Bharadwaj, Jai Nanavati, Raoul Nanavati, Karthik Sankaranarayanan, Tejaswi Seeram, and Basil Abraham. Multilingual and code-switching ASR challenges for low resource indian languages. *CoRR*, abs/2104.00235, 2021.

[12] Mengzhe Geng, Xurong Xie, Shansong Liu, Jianwei Yu, Shoukang Hu, Xunying Liu, and Helen Meng. Investigation of data augmentation techniques for disordered speech recognition. 10 2020.

[13] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA, 2006. Association for Computing Machinery.

[14] Alex Graves, Navdeep Jaitly, and Abdel rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, 2013.

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

[16] Awni Hannun. Sequence modeling with ctc. *Distill*, 2017. https://distill.pub/2017/ctc.

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

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

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

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

[21] kamo naoyuki. ESPnet2 pretrained model, kamo-naoyuki/librispeech _asr_train_asr_conformer6_n_fft512_hop_length256_r aw_en_bpe5000_scheduler_confwarmup_steps40000_opti m_conflr0.0025_sp_valid.acc.ave, fs=16k, lang=en, March 2021.

[22] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang. A comparative study on transformer vs RNN in speech applications. *CoRR*, abs/1909.06317, 2019.

[23] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint ctc-attention based end-to-end speech recognition using multi-task learning. *CoRR*, abs/1609.06773, 2016.

[24] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *INTERSPEECH*, 2015.

[25] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226, 2018.

[26] Aleksandr Laptev, Roman Korostik, Aleksey Svischev, Andrei Andrusenko, Ivan Medennikov, and Sergey Rybin. You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation. *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, Oct 2020.

[27] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Rwth asr systems for librispeech: Hybrid vs attention. *Interspeech 2019*, Sep 2019.

[28] Andrew L. Maas, Awni Y. Hannun, Daniel Jurafsky, and Andrew Y. Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *CoRR*, abs/1408.2873, 2014.

[29] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531, 2011.

[30] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *INTERSPEECH*, pages 1045–1048. ISCA, 2010.

[31] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[32] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*, Sep 2019.

[33] Vijayaditya Peddinti, D. Povey, and S. Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH*, 2015.

[34] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[35] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, X. Na, Yiming Wang, and S. Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *INTERSPEECH*, 2016.

[36] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[37] S. S. Stevens and J. Volkmann. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3):329–353, 1940.

[38] S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

[39] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 1017–1024, Madison, WI, USA, 2011. Omnipress.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[41] Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. Improving deep neural network acoustic models using generalized maxout networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 215–219, 2014.

[42] Yingbo Zhou, Caiming Xiong, and Richard Socher. Improved regularization techniques for end-to-end speech recognition. *CoRR*, abs/1712.07108, 2017.