Improving Low Resource Code-switched ASR using Augmented Code-switched TTS

Yash Sharma[1], Basil Abraham[2], Karan Taneja[1], Preethi Jyothi[1]

[1]Indian Institute of Technology Bombay, Mumbai, India
[2]Microsoft India Development Center, Hyderabad, India

{yashsharma,karantaneja,pjyothi}@iitb.ac.in, baabraha@microsoft.com

Abstract

Building Automatic Speech Recognition (ASR) systems for code-switched speech has recently gained renewed attention due to the widespread use of speech technologies in multilingual communities worldwide. End-to-end ASR systems are a natural modeling choice due to their ease of use and superior performance in monolingual settings. However, it is well-known that end-to-end systems require large amounts of labeled speech. In this work, we investigate improving code-switched ASR in low resource settings via data augmentation using code-switched text-to-speech (TTS) synthesis. We propose two targeted techniques to effectively leverage TTS speech samples: 1) Mixup, an existing technique to create new training samples via linear interpolation of existing samples, applied to TTS and real speech samples, and 2) a new loss function, used in conjunction with TTS samples, to encourage code-switched predictions. We report significant improvements in ASR performance achieving absolute word error rate (WER) reductions of up to 5%, and measurable improvement in code switching using our proposed techniques on a Hindi-English code-switched ASR task.

Index Terms: speech recognition, text to speech, mixup, augmentation, bilingual, code switching, code mixing

1. Introduction

Code mixing or code switching (CS) is a widespread linguistic phenomenon among speakers in multilingual communities, where they switch between two or more languages often within the confines of a sentence. Code switching appears naturally in conversational speech. With the prevalence of social media, code switching also appears frequently in textual form. As a result, building computational models for code switched speech and text has gained a lot of recent attention.

As is typical for state-of-the-art ASR systems, CS speech ASR would also require large quantities of labeled speech during training. While one could reasonably hope to gain access to large monolingual corpora in the component languages, CS speech data is a relatively scarce resource. Thus, it is of great interest to investigate ways in which one can augment CS speech data in order to effectively train CS ASR systems.

In this work, we investigate the effective use of CS text-to-speech (TTS) samples to augment ASR systems for CS speech. With CS text as the starting point, large quantities of TTS samples can be generated. TTS systems can be trained with far fewer hours of clean labeled speech compared to ASR systems. However, there is a distributional mismatch between TTS samples and real speech samples. Using TTS samples as-is could lead to over-fitting on artefacts in the TTS data. The main question of interest in this work is how to best leverage a fixed quantity of TTS samples to improve ASR for CS speech. Towards this, our contributions can be summarized as follows:

- We suggest mixup [1] as an appropriate strategy to be applied to TTS samples and real speech samples in order to help reduce the distributional mismatch.

- We propose a new *CS-bias* loss function that encourages the ASR system to code-switch and performs well in conjunction with TTS samples.

- We demonstrate the efficacy of our proposed techniques on a Hindi-English code-switched ASR task. We also measure error reductions at the switching points and observe clear improvements in performance.

2. Related Work

Prior work on speech recognition for CS speech has focused on aspects relevant to both the acoustic model and the language model. For the acoustic model, previous approaches have investigated efficient ways to do phone merging [2], tying mixture models of two languages within a GMM-HMM model [3] and using language-dependent inference [4]. There has also been a lot of prior work on improving LMs for CS text that includes the use of parallel text in the CS language pair [5], the use of generative models to synthesize CS text [6, 7, 8, 9]. In more recent work, end-to-end models have also been explored for code-switched ASR with improvements observed using language identification [10, 11, 12] in multitask (MTL) settings, jointly using CTC and attention based loss [13, 14] within an MTL model, multi-graph decoding and LM rescoring [15] for low-resource CS ASR, and using various units like wordpieces [10] and byte-pair encoding (BPE) units [16] to model text. Apart from techniques targeted at CS ASR, several other data augmentation techniques have been shnown to boost the performance of ASR systems such as speed perturbation [17], randomized spectrogram masking [18], vocal tract length perturbation and semi-supervised learning [19].

Due to the lack of sufficient CS labelled corpora for training, [20, 21] train ASR systems using synthetic datasets created by concatenating monolingual segments from the constituent languages of the CS speech. In contrast, we explore the use of samples generated from a TTS systems for training our ASR system. [22, 23] use TTS samples as an augmentation technique to improve ASR performance on the LibriSpeech corpus. For monolingual speech recognition in the low-resource scenario, [24] uses TTS samples to introduce speaker diversity. We propose techniques that utilize TTS speech for code-switched ASR with access to only 12 hours of real CS speech. [25] uses TTS and ASR set up as a chain to train a CS ASR system from non-parallel speech and text corpora. Though synthetic CS data can be generated using synthesis engines [26, 27], TTS samples significantly differ from real speech in their acoustic characteristics and hence need to be suitably handled in order to be more

effective as sources of data augmentation.

3. Our Proposed Techniques

The use of TTS samples for data augmentation and improving ASR performance has been investigated in recent work [22, 23]. In order to insulate the ASR model from artefacts specific to TTS data, prior work has observed that freezing the encoder parameters [28] and only updating the rest of the model using TTS samples is beneficial. We present this strategy as a baseline and propose two other techniques in this section that further improve performance.

3.1. Mixup Algorithm and Sampling

Mixup [1] was originally proposed as a simple regularization technique to train a classifier using convex combinations of pairs of training instances and their corresponding labels. Mixup has also been investigated within the paradigm of ASR [29, 30, 31, 32]. Given spectral features of two speech samples, X_a and X_b, the *mixed* training example X_{mix} is given by:

$$X_{\text{mix}} = \lambda_{\text{mix}} \cdot X_a + (1 - \lambda_{\text{mix}}) \cdot X_b \qquad (1)$$

Here, $\lambda_{\text{mix}} = \max(\lambda, 1 - \lambda)$ where $\lambda \sim \text{Beta}(\alpha, \beta)$, and α, β are the standard shape parameters of the beta distribution. We apply this mixup strategy to TTS samples and real speech samples such that X_a is a TTS sample and X_b is a randomly-chosen real speech sample. For our experiments, we set $\alpha = 0.4$ and $\beta = 0.4$. They were fine-tuned on **Dev** set. The resulting shape of the beta distribution ensures that a λ sampled from this distribution would either be very close to 0 or 1. Given this skew, we set the transcription of the mixed utterance X_{mix} to be the same as that of the TTS sample X_a.

With the above-mentioned mixup strategy in place, we intend for real speech samples to be mixed in with TTS samples to bring the latter a little closer to real speech in the input feature space. Given that the acoustic properties of TTS speech differ from natural speech in various dimensions, particularly in its intonational properties, we hope mixup helps mitigate these differences to a small extent and acts as a regularizer. This is indeed borne out by our experiments shown in Section 5.1.

3.2. Rewarding Code-switching with a *CS-bias* Loss

Before describing our new loss function, we briefly outline the end-to-end ASR architecture employed in this work. Our base ASR model is a standard hybrid CTC-attention end-to-end architecture that uses an encoder-decoder paradigm with attention [33]. This model is trained with a multitask objective that uses a weighted combination of a CTC-based loss and an attention-based loss via a shared encoder. The combined loss function is referred to as L_{MTL}. For CS speech, we devise a new loss term to be added to L_{MTL} that explicitly rewards code switching.

Many common English words have been assimilated into the Hindi language and appear in the Devanagari script within the Hindi vocabulary. They also appear as English words in their romanized forms. However, since Hindi is typically the dominant (matrix) language in Hindi-English CS speech, the model is hesitant to code-switch even when the acoustics have sufficient evidence of a potential switch point. Our proposed modification to the loss function rewards higher probabilities being assigned to English characters. This reward can be quan-

tified as: $\mathbf{R} = \sum_{t=1}^{N} \sum_{c \in \mathcal{S}_e} P_t(c)$, where $P_t(c)$ is the probability at output timestep t assigned to a character c that is drawn from the set of all English characters denoted by $\mathcal{S}_e$. (The output sequence is of total length N.)

The modified *CS-bias* loss function then becomes:

$$L_{\text{CS}} = L_{\text{MTL}} - \lambda' \cdot (\mathbf{R}_{\text{CTC}} + \mathbf{R}_{\text{ATT}}) \qquad (2)$$

where $\mathbf{R}_{\text{CTC}}$ and $\mathbf{R}_{\text{ATT}}$ are rewards specific to the CTC and attention outputs and λ' is the scaling coefficient associated with the new loss term. The reward term is meant to serve as an explicit boost to English characters whenever speech switches to English.

4. Experimental Setup

Datasets. We use a proprietary dataset from Microsoft comprising 50 hours of Hindi monolingual speech (**Hi-50**) and 50 hours of Indian accented English speech (**En-50**).We also make use of approximately 12 hours of real CS Hindi-English speech that amounts to 12K CS utterances (i.e. 12K different transcriptions). This dataset will henceforth be referred to as **CS-Real**. The speech corpus consists of mostly conversational type utterances collected in a multi-condition scenario. The utterances are transcribed in both Roman and Devanagari scripts corresponding to English and Hindi language respectively [21]. For the first set of experiments, until Section 3.2, we restrict ourselves to TTS samples generated using the text in **CS-Real** and assume no access to real CS speech. In Section 5.4, we show the effect of our techniques on a model that has been finetuned using speech in **CS-Real**.

Our models are evaluated on two evaluation datasets comprising real CS speech. The development set (**Dev**) consists of $2,000$ sentences with a total of $25,679$ words and the unseen test set (**Test**) contains $2,000$ sentences with $29,408$ words overall. **Dev** is used to tune α, β, and other hyperparameters that appear in our techniques.

TTS Generation. To generate TTS samples, we train a Tacotron-2 model [34] that employs a multi-speaker, multilingual acoustic model and maps phone sequences to Mel spectrograms. Our model is trained using roughly 30,000 utterances from six different Indian languages – English, Hindi, Tamil, Telugu, Gujarati and Marathi – spoken by a single speaker in each language. The acoustic model is created using a common phone set which is the union of phone sets of all six languages. Phones that are common across different languages get merged, while phones that are specific to a language remain intact.We use a parallel Wavenet vocoder [35] that is trained on large set of Microsoft proprietary TTS data.

We train the acoustic model for 250K iterations with a batch size of 32. We also make use of high-quality speech data from 70 Hindi speakers speaking in different accents from a larger Hindi dataset from which **Hi-50** was drawn [21]. We further adapt our acoustic model with 100-200 samples from specific speakers for an additional 10K iterations with the same training hyperparameters. This final adaptation step yields speaker-specific models for each speaker. Finally, we feed CS text from **CS-Real** as input and generate speech samples from these speaker-adapted models that amount to roughly 100 hours of speech overall.

Implementation Details. We use the ESPnet toolkit [33] for all our experiments as it provides a flexible interface for many end-to-end (E2E) architectures. As mentioned in Section 3.2, we train our baseline model using a multi-task (MTL) objective that combines a CTC loss and an attention loss via a shared encoder. All our models use graphemes as the output vocabulary.[1]

The encoder architecture consists of 4 layers of stacked bidirectional LSTMs, with 512 encoder units, 512-dimensional encoder projections and sub-sampling to reduce sequence lengths. We used a location-aware attention-based model with a 1-layer decoder for the CTC and attention modules each. The maximum input length is 800 frames and the maximum output length is 150 tokens. λ_{MTL} is the interpolation coefficient used in the combined loss L_{MTL}; this was set to 0.7 in all our experiments, inspired from results in [14]

Our output vocabulary includes both English and Devanagari characters, along with a "space" character, an unknown marker and an end-of-sentence marker. The model was trained using the Adadelta optimizer [37] with a base learning rate of 1.0. We trained the network for a total of 20 epochs with a patience value set to 10. All experiments used data shuffling as well as sortagrad [38] for all epochs, i.e. we first sorted the data by length, batched the samples, and then shuffled the order of batches. This turned out to perform better than doing no sortagrad and no shuffling. The effective batch size was set to 32 in all our experiments. For some experiments where we fine-tune a pre-trained model using TTS samples, we freeze the encoder as described in [28]. These experiments are labelled with **FE** (frozen encoder).

We used a word-based recurrent neural network language model (RNNLM) with a weight of 0.1. The RNNLM is a single-layer LSTM with 1000 units and an output vocabulary restricted to the 2000 most frequently occurring words in **CS-Real**. It is trained for 40 epochs before converging with a patience of 20. During decoding, we used beam decoding with a width of 20 and no insertion penalty was invoked.

5. Results

Our baseline hybrid CTC-based ASR model is trained on the two monolingual **Hi-50** and **En-50** datasets merged together.[2] The subsequent sections will be structured as follows. In Section 5.1, we show the benefits of mixup in conjunction with TTS samples. Section 5.2 will show the effect of using the *CS-bias* loss function when using TTS samples. Finally, in Section 5.4, we finetune our baseline ASR system using real CS speech in **CS-Real** and examine the effects of our proposed techniques on the resulting system.

5.1. Mixup with TTS samples

Table 1 shows how mixup is beneficial when used in conjunction with TTS samples. As mentioned before, the "Baseline" system does not make use of any CS speech. The "no-mixup" systems refer to the use of TTS samples with the real speech samples without any mixup. The numbers alongside each system denote the number of hours of TTS speech used in each experiment. That is, "no-mixup 50" refers to the use of 100 hours of real speech (from **Hi-50** and **En-50**) and 50 hours of TTS speech without any mixup in place. We observe that

---

[1]We also tried a byte-pair-encoding (BPE) [36] based model, but this did not perform as well as the purely grapheme-based models.

[2]By tuning on **Dev**, we observed that merging the Hindi and English monolingual data in unit ratio performed the best.

Table 1: *WERs (CERs) using TTS data, with and without mixup*

System	Dev	Test
Baseline	70.7 (54.3)	72.1 (54.5)
no-mixup 20	71.5 (49.1)	72.9 (49.2)
no-mixup 50	69.7 (48.6)	71.4 (48.5)
no-mixup 100	70.0 (48.7)	71.0 (48.2)
add-mixup 20	70.3 (48.5)	71.4 (48.2)
add-mixup 50	**68.8** (47.9)	70.3 (47.9)
add-mixup 100	69.0 (47.9)	**70.0** (**47.4**)

Table 2: *WERs (CERs) using TTS data, with and without freezing the encoder*

System	Dev	Test
no-mixup 100	70.0 (48.7)	71.0 (48.2)
no-mixup 100 (**FE**)	71.2 (46.7)	73.0 (47.6)
add-mixup 100	**69.0** (47.9)	**70.0** (47.4)
add-mixup 100 (**FE**)	69.5 (**45.2**)	71.3 (**46.3**)

adding code-switched TTS samples to the monolingual speech, even without any mixup, is beneficial. Adding more TTS data improves performance further, however there is a trend of diminishing returns (72.9 → 71.4 → 71.0). The "add-mixup" systems, in contrast to "no-mixup", uses mixup with each batch of TTS samples. Each batch of TTS speech samples is mixed with a different batch of real speech, and λ_{mix} is sampled anew for each such batch-wise mixing. We observe clear improvements in performance with using "add-mixup" in comparison to "no-mixup".

We examine the effect of freezing the encoder (**FE**) as suggested in [28] on both "no-mixup" and "add-mixup" in Table 2. We observe that freezing the encoder was not an effective strategy in our setting possibly due to the use of smaller amounts of TTS data (in comparison to the corpus used in prior work). However, the CERs do improve with freezing the encoder. This will need further investigation which we leave as future work.

When augmenting real speech samples with TTS samples in the "add-mixup" setting, there are two ways in which this could be implemented. The mixup samples could be used to train the ASR system from scratch or one could start with the baseline ASR system and further finetune it using the mixup samples. Table 3 shows results from both these training strategies. Training from scratch appears to be a more effective strategy yielding lower error rates on both **Dev** and **Test** compared to the systems using finetuning (denoted by **FT**).

Table 3: *WER(CER): Finetune vs from scratch experiments*

System	DEV	TEST
add-mixup 20 (scratch)	70.3 (48.5)	71.4 (48.2)
add-mixup 50 (scratch)	**68.8** (47.9)	70.3 (47.9)
add-mixup 100 (scratch)	69.0 (47.9)	**70.0** (47.4)
add-mixup 20 (**FT**)	69.8 (48.4)	71.6 (48.3)
add-mixup 50 (**FT**)	69.7 (47.6)	70.9 (47.7)
add-mixup 100 (**FT**)	68.9 (48.0)	70.2 (47.8)

Table 4: *WERs (CERs) highlighting the effect of CS-bias. A-100 refers to the "add-mixup 100" system.*

System	Dev	Test
Baseline	70.7 (54.3)	72.1 (54.5)
A-100	69.0 (47.9)	70.0 (47.4)
A-100 (**FT**)	68.9 (48.0)	70.2 (47.8)
A-100 + *CS-bias* (scratch)	**67.7** (47.4)	**69.0** (47.3)
A-100 (**FT**) + *CS-bias*	68.6 (47.3)	70.2 (47.3)

5.2. Influence of *CS-bias* loss function

We observe that the modified *CS-bias* loss function is effective in encouraging CS predictions and reduces the overall WER on our evaluation datasets, over and above the improvement gained by using mixup. Explicitly rewarding English predictions helps overcome the model's bias towards predicting back-transliterated Devaganari words instead of English words. In all our experiments involving *CS-bias*, λ' was set to 0.25.

Table 4 shows how *CS-bias* influences performance. Including this loss term during training of an "add-mixup 100" system from scratch is most beneficial. We see a clear reduction in WERs both on the **Dev** and **Test** datasets. We also tried including the loss term as part of the finetuning phase while training "add-mixup 100 **FT**". However, this did not help as much possibly because the model is already accustomed to predicting monolingual sentences and more resistant to switching.

5.3. Accuracy on CS Switch-points

The above experiments suggest that the *CS-bias* loss is useful in improving CS ASR performance. However, it would be illustrative to examine the errors at code switching points and the influence of *CS-bias* on these specific errors. [21] introduces one such metric that helps us compute errors at the switching points. If there are M words on both sides of the switch points across all reference transcriptions and N of these switch point words are predicted correctly, then the code-switched WER (CS-WER) is defined as, CS-WER = $1 - \frac{N}{M}$.

Table 5 lists the CS-WER values on both **Dev** and **Test** datasets for a few important systems identified in the previous experiments. As intended, we see consistent improvements on the CS-WER metric with using the *CS-bias* loss. Interestingly, the system with the best CS-WER score was one that used "add-mixup" in a finetuning phase with a frozen encoder along with the *CS-bias* loss. This could be because of the combination of *CS-bias* explicitly encouraging switching and the frozen encoder preventing the model from encoding artefacts from TTS speech.

5.4. Finetuning on small amounts of real CS speech

In all the experiments so far, we assume no access to real CS speech and exclusively used synthetic CS speech (in addition to monolingual speech). However, a natural question that may arise is how our proposed techniques fare when the model can also make use of small amounts of real CS speech. To demonstrate this, we use speech from **CS-Real** to finetune our baseline model. Table 6 shows that even with access to just 12 hours of real CS speech the WERs dramatically improve over the purely monolingual baseline system. Starting from the "add-mixup 100" system that was trained with *CS-bias* and further finetuning with **CS-Real** further improves WERs on both **Dev** and **Test**

Table 5: *Comparing CS-WER on different systems*

System	DEV	TEST
Baseline	82.4	83.3
no-mixup 100 (w/o *CS-bias*)	80.3	81.1
no-mixup 100 (with *CS-bias*)	79.7	81.2
add-mixup 100 (w/o *CS-bias*)	79.7	80.6
add-mixup 100 (with *CS-bias*)	79.3	79.8
add-mixup 100 (**FT**, w/o *CS-bias*)	79.1	80.4
add-mixup 100 (**FT**, with *CS-bias*)	78.7	79.8
add-mixup 100 (**FT**, with **FE**)	75.5	78.3
add-mixup 100 (**FT**, **FE**, *CS-bias*)	**73.1**	**76.1**

Table 6: *WERs / CS-WERs after finetuning with real CS speech*

System	DEV	TEST
Baseline (no CS speech)	70.7/82.4	72.1 /83.3
FT with **CS-Real** (B1)	58.6/59.8	59.9/60.2
B1 + *CS-bias*	58.4/58.8	59.4/59.6
A-100 (**FT** w/ **CS-Real**)	54.4/54.0	55.5/54.8
A-100 + *CS-bias* (**FT** w/ **CS-Real**)	**53.6/53.3**	**55.2/54.4**
A-100 (**FT** w/ **CS-Real**) + *CS-Bias*	54.8/54.7	55.7/55.5

datasets. Table 6 also shows the CS-WER values for all the models. Both our proposed techniques also help in achieving the best CS-WER.

6. Conclusions

In this work, we explore the usability of high quality text-to-speech (TTS) data as a resource for CS ASR in settings where natural code-switched speech is unavailable or is a scarce resource. We present two effective techniques to make use of TTS samples: mixup and a loss function that encourages code switching. We show performance improvements using both these techniques in two different experimental settings involving no real CS speech and small amounts of CS speech. This work opens up many interesting questions including whether there are other auxiliary loss functions that can help code switching better and how the proposed techniques will scale to very large quantities of TTS samples. We leave these explorations for future work.

7. Acknowledgements

8. References

[1] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," 2017.

[2] N. T. Vu, D.-C. Lyu, J. Weiner, D. Telaar, T. Schlippe, F. Blaicher, E.-S. Chng, T. Schultz, and H. Li, "A first speech recognition system for Mandarin-English code-switch conversational speech," in *Proceedings of ICASSP*, 2012, pp. 4889–4892.

[3] Y. Li, P. Fung, P. Xu, and Y. Liu, "Asymmetric acoustic modeling of mixed language speech," in *Proceedings of ICASSP*, 2011, pp. 5004–5007.

[4] D. Imseng, H. Bourlard, M. M. Doss, and J. Dines, "Language dependent universal phoneme posterior estimation for mixed language speech recognition," in *Proceedings of ICASSP*, 2011, pp. 5012–5015.

[5] G. I. Winata, A. Madotto, C. Wu, and P. Fung, "Learn to code-switch: Data augmentation using copy mechanism on language modeling," *CoRR*, vol. abs/1810.10254, 2018.

[6] E. Yılmaz, H. van den Heuvel, and D. Van Leeuwen, "Acoustic and textual data augmentation for improved ASR of code-switching speech," in *Proceedings of Interspeech*, 2018, pp. 1933–1937.

[7] H. Adel, N. T. Vu, F. Kraus, T. Schlippe, H. Li, and T. Schultz, "Recurrent neural network language modeling for code switching conversational speech," in *Proceedings of ICASSP*, 2013, pp. 8411–8415.

[8] S. Garg, T. Parekh, and P. Jyothi, "Dual language models for code switched speech recognition," in *Proceedings of Interspeech*, 2018, pp. 2598–2602.

[9] A. Pratapa, G. Bhat, M. Choudhury, S. Sitaram, S. Dandapat, and K. Bali, "Language modeling for code-mixing: The role of linguistic theory based synthetic data," in *Proceedings of ACL*, 2018, pp. 1543–1553.

[10] C. Shan, C. Weng, G. Wang, D. Su, M. Luo, D. Yu, and L. Xie, "Investigating end-to-end speech recognition for Mandarin-English code-switching," in *Proceedings of ICASSP*, 2019, pp. 6056–6060.

[11] K. Li, J. Li, G. Ye, R. Zhao, and Y. Gong, "Towards code-switching ASR for end-to-end CTC models," in *Proceedings of ICASSP*, 2019, pp. 6076–6080.

[12] S. Toshniwal, T. N. Sainath, R. J. Weiss, B. Li, P. J. Moreno, E. Weinstein, and K. Rao, "Multilingual speech recognition with a single end-to-end model," in *Proceedings of ICASSP*, 2018, pp. 4904–4908.

[13] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proceedings of ICASSP*, 2017, pp. 4835–4839.

[14] B. M. L. Srivastava, B. Abraham, S. Sitaram, R. Mehta, and P. Jyothi, "End-to-end ASR for code-switched Hindi-English speech," *CoRR*, vol. abs/1906.09426, 2019.

[15] X. Yue, G. Lee, E. Yilmaz, F. Deng, and H. Li, "End-to-end code-switching ASR for low-resourced language pairs," in *Proceedings of ASRU*, 2019, pp. 972–979.

[16] B. Li, Y. Zhang, T. N. Sainath, Y. Wu, and W. Chan, "Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes," *CoRR*, vol. abs/1811.09021, 2018.

[17] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proceedings of Interspeech*, 2015, pp. 3586–3589.

[18] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proceedings of Interspeech*, 2019, pp. 2613–2617.

[19] A. Ragni, K. Knill, S. P. Rath, and M. J. F. Gales, "Data augmentation for low resource languages," in *Proceedings of Interspeech*, 2014, pp. 810–814.

[20] H. Seki, S. Watanabe, T. Hori, J. L. Roux, and J. R. Hershey, "An end-to-end language-tracking speech recognizer for mixed-language speech," in *Proceedings of ICASSP*, 2018, pp. 4919–4923.

[21] K. Taneja, S. Guha, P. Jyothi, and B. Abraham, "Exploiting monolingual speech corpora for code-mixed speech recognition," in *Proceedings of Interspeech*, 2019, pp. 2150–2154.

[22] A. E. Rosenberg, Y. Zhang, B. Ramabhadran, Y. Jia, P. J. Moreno, Y. Wu, and Z. Wu, "Speech recognition with augmented synthesized speech," in *Proceedings of ASRU*, 2019, pp. 996–1002.

[23] G. Wang, A. Rosenberg, Z. Chen, Y. Zhang, B. Ramabhadran, Y. Wu, and P. Moreno, "Improving speech recognition using consistent predictions on synthesized speech," in *Proceedings of ICASSP*, 2020.

[24] C. Du and K. Yu, "Speaker augmentation for low resource speech recognition," in *Proceedings of ICASSP*, 2020, pp. 7719–7723.

[25] S. Nakayama, A. Tjandra, S. Sakti, and S. Nakamura, "Speech chain for semi-supervised learning of Japanese-English code-switching ASR and TTS," in *SLT Workshop*, 2018, pp. 182–189.

[26] S. Nakayama, T. Kano, Q. T. Do, S. Sakti, and S. Nakamura, "Japanese-english code-switching speech data construction," in *International Conference on Speech Database and Assessments*, 2018, pp. 67–71.

[27] T. Kano, S. Sakti, and S. Nakamura, "Structured-based curriculum learning for end-to-end english-japanese speech translation," in *Proceedings of Interspeech*, 2017, pp. 2630–2634.

[28] Y. Jia, M. Johnson, W. Macherey, R. J. Weiss, Y. Cao, C. Chiu, N. Ari, S. Laurenzo, and Y. Wu, "Leveraging weakly supervised data to improve end-to-end speech-to-text translation," in *Proceedings of ICASSP*, 2019, pp. 7180–7184.

[29] I. Medennikov, Y. Y. Khokhlov, A. Romanenko, D. Popov, N. A. Tomashenko, I. Sorokin, and A. Zatvornitsky, "An investigation of mixup training strategies for acoustic models in ASR," in *Proceedings of Interspeech*, 2018, pp. 2903–2907.

[30] G. Saon, Z. Tüske, K. Audhkhasi, and B. Kingsbury, "Sequence noise injected training for end-to-end speech recognition," in *Proceedings of ICASSP*, 2019, pp. 6261–6265.

[31] Y. Zhu, T. Ko, and B. Mak, "Mixup learning strategies for text-independent speaker verification," *Proceedings of Interspeech*, pp. 4345–4349, 2019.

[32] Y. Liang, L. Yang, X. Wang, Y. Li, C. Jia, and J. Wang, "The levoice far-field speech recognition system for voices from a distance challenge 2019," *Proceedings of Interspeech*, pp. 2483–2487, 2019.

[33] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proceedings of Interspeech*, 2018, pp. 2207–2211.

[34] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions," in *Proceedings of ICASSP*, 2018, pp. 4779–4783.

[35] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, "Parallel WaveNet: Fast high-fidelity speech synthesis," in *Proceedings of ICML*, 2019.

[36] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of ACL*, 2016, pp. 1715–1725.

[37] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.

[38] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, "Deep Speech 2: End-to-end speech recognition in English and Mandarin," in *Proceedings of ICML*, 2016, p. 173–182.