INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

UniSplice: Universal Cross-Lingual Data Splicing for Low-Resource ASR

Wei Wang, Yanmin Qian[†]

MoE Key Lab of Artificial Intelligence, AI Institute
X-LANCE Lab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China

wangwei.sjtu@sjtu.edu.cn, yanminqian@sjtu.edu.cn

Abstract

End-to-end (E2E) automatic speech recognition (ASR) has made remarkable progress thanks to the abundant annotated data for a few rich-resource languages. However, data scarcity remains a challenge for the majority of the world's languages. To address this issue, we propose UniSplice, a novel cross-lingual speech synthesis framework based on data splicing that leverages self-supervised learning (SSL) units from Hidden Unit BERT (HuBERT) as universal phonetic units. Our approach involves splicing speech fragments from rich-resource languages into complete speech that conforms acoustically to text from low-resource languages. UniSplice eliminates the need for computationally expensive neural text-to-speech (TTS) models, enabling the training of ASR models using on-the-fly synthesized speech. Experimental results on the COMMON-VOICE dataset show 20-30% relative improvement for four Indo-European languages and about 15% for Turkish with a 4-gram language model for rescoring, in a 10-hour low-resource setup.

Index Terms: low resource speech recognition, text-to-speech, data splicing, self-supervised learning

1. Introduction

End-to-end (E2E) automatic speech recognition (ASR) models have gained popularity in recent years due to their streamlined architecture and promising performance [1–4]. However, the effectiveness of E2E ASR models heavily relies on the availability of large amounts of transcribed audio data, which is often not the case in low-resource settings. This data scarcity issue poses a significant challenge to the applicability of E2E ASR models in low-resource languages [5]. To overcome this challenge, researchers have proposed various solutions, including multilingual transfer learning (MultiASR) [6–10] and self-supervised learning (SSL) techniques [11–17]. These approaches rely on paired or unpaired data from various languages to pretrain a seed model that can be fine-tuned for low-resource target languages. In contrast, this paper aims to tackle the data scarcity issue with an efficient text-to-speech (TTS) framework based on data splicing to generate more paired data in low-resource target languages, rather than optimizing the pretraining stage.

TTS architectures have been extensively employed to enhance ASR performance [18–25]. Specifically, [18, 19] utilized neural TTS models to adapt the RNN-T [26] model from the source domain to the target domain, while [21] explored the machine speech chain [20] framework to accomplish alternate adaptation for TTS and ASR models from the audiobook domain to the presentation domain. In addressing the degradation of recognition accuracy for out-of-vocabulary (OOV) words, [22, 23] trained ASR models on audio synthesized with text containing OOV words. [25] improved the performance of a children's ASR system by enhancing the quality of synthesized children's speech using multiple filtering algorithms. However, neural TTS models typically demand substantial quantities of high-quality data for training. While there has been a significant reduction in the amount of high-quality single-speaker paired data required to train these models [27–30], synthesizing coherent speech for multiple speakers with noisy ASR data in low-resource settings for training, remains a formidable challenge for neural TTS models. Moreover, neural TTS models suffer from substantial computational expenses during training and inference, posing a significant obstacle to training an ASR model using speech synthesized on-the-fly.

In [31], the authors proposed a TTS method that utilizes a world-level splicing data generation (SDG) approach, which was shown to outperform even neural TTS approaches for the text-only domain adaptation task, while requiring a negligible computational cost. [31] builds a mapping from words to speech fragments using a customized RNN-T, enabling text from the target domain to be mapped to speech fragments from the source domain and spliced into complete speech. However, the word SDG approach is limited to monolingual scenarios due to the absence of universal word tokens across languages.

To address this limitation, we proposed UniSplice, a novel SDG framework that can operate in cross-lingual scenarios. Our approach involves splicing speech fragments from rich-resource languages into complete speech by referencing text from a low-resource language. To achieve this, we adopt the denoised clustering unit (Hunit) from latent representations of Hidden Unit BERT (HuBERT) [12] as universal phonetic units across languages. We build a mapping (HuDict) from Hunit n-grams to speech fragments, and we develop a lightweight model to perform Grapheme-to-Hunit (G2H) conversion. During speech synthesis, a text sample from the low-resource language is mapped into Hunits by the G2H model, and then into speech fragments by the HuDict. These speech fragments are finally concatenated into complete speech that acoustically matches the input text. Details are elaborated in Fig 1.

Our contributions can be summarized as follows: (1) We introduce an efficient data splicing framework to train ASR models with on-the-fly synthesized speech. (2) Our framework leverages universal phonetic units derived from SSL units for data splicing, enabling it to be language-agnostic. (3) Our proposed framework can operate on unpaired speech from rich-resource languages and unpaired text from low-resource languages, making it readily applicable. (4) Experiment results on various low-resource languages demonstrate consistent word er-

[†] corresponding author

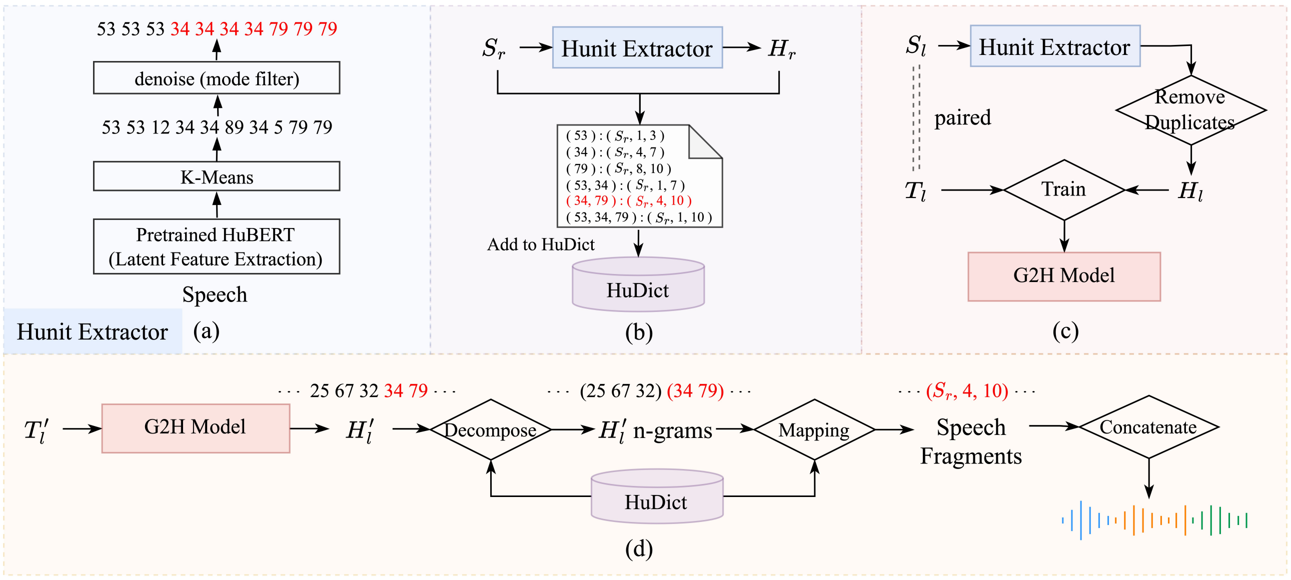

Figure 1: *Proposed UniSplice data splicing framework: (a) The Hunit extractor takes the raw waveform as input and extracts denoised clustering units (Hunits) from the latent representations. (b) Hunit extractor extracts H_r given S_r. The mappings from Hunit n-grams in H_r to the corresponding speech fragments in S_r are added to HuDict. (c) Limited paired data $\{S_l, T_l\}$ processed by Hunit extractor after duplicate removal is used to train a G2H model. (d) During synthesis, T_l' is converted to H_l' by the G2H model and mapped to speech fragments by querying HuDict, which are finally concatenated into complete speech.*

ror rate (WER) reduction by integrating cross-lingually spliced data into the training of low-resource ASR models.

2. Methodology

2.1. HuBERT

We first briefly recapitulate HuBERT. HuBERT is an SSL approach exploiting an offline clustering step to provide aligned target labels for a BERT-like prediction loss [32]. The backbone used in HuBERT is a convolutional encoder, followed by a BERT mask predictor comprised of many identical Transformer blocks [33]. Given an encoded speech embedding sequence $X = [x_1, \ldots, x_T]$, the mask predictor predicts a distribution over the target codewords at each timestep t with the masked version $\hat{X}$. Denote C as the number of clustered codewords, $\mathbf{e}_c$ as the embedding for codeword c and $\mathbf{A}_t$ as the output feature at step t. The distribution over codewords is formulated as:

$$p(c \mid \hat{X}, t) = \frac{\exp(\mathrm{sim}(\mathbf{A}_t \mathbf{W}, \mathbf{e}_c)/\tau)}{\sum_{c'=1}^{C} \exp(\mathrm{sim}(\mathbf{A}_t \mathbf{W}, \mathbf{e}_{c'})/\tau)} \tag{1}$$

where $\mathbf{W}$ is a projection matrix. $\mathrm{sim}(\cdot,\cdot)$ computes the cosine similarity and τ scales the logit.

Denote discrete target sequence for X as $Z = [z_1, z_2, \ldots, z_T]$, the prediction loss is formulated as:

$$\mathcal{L} = \sum_{t \in M} \log p\left(z_t \mid \hat{X}, t\right) \tag{2}$$

where $M \subset \{1 \ldots T\}$ are the masked timesteps for X.

The clustered codewords, which are iteratively refined and exhibit correlation with the underlying acoustic units, are used as universal phonetic units in our approach.

2.2. UniSplice

The proposed data splicing framework is comprised of three components, which require the following datasets: (i) unpaired speech from the rich-resource language, denoted as D_r and sampled as S_r, (ii) limited paired speech from the low-resource language, denoted as D_l and consisting of speech-text pairs $\{S_l, T_l\}$, and (iii) text-only data from the low-resource language, denoted as D_l' and consisting of text samples T_l'. These components are illustrated in Fig 1 (a) (b) (c), while Fig 1 (d) demonstrates the audio synthesis procedure.

2.2.1. *Hunit Extractor*

As shown in Fig 1 (a), the Hunit extractor comprises three modules: a pretrained HuBERT model, a K-Means cluster module, and a denoising module. The denoising module consists of a stack of mode filters, which are commonly used in image segmentation to replace pixels with the most common pixel value within a given window size. This process creates an impasto effect that preserves edges while smoothing out the noise. We apply mode filters to the clustered units of the HuBERT latent representations to produce reasonable articulatory boundaries.

2.2.2. *Hunit Dictionary (HuDict)*

The HuDict is a mapping from Hunit n-grams to speech fragments from D_r. For unpaired speech $S_r \in D_r$, we obtain the frame-level alignment between S_r and H_r with the Hunit extractor. As shown in Fig 1 (b), the mapping from Hunit n-grams to triplets *(utterance id, start frame, end frame)*, which indicates how the corresponding speech fragment could be fetched, are extracted from the frame-level alignment and added to HuDict. In practice, we only collect n-grams mappings with $4 \leq n \leq 8$. $n \geq 4$ is set to achieve a better quality of spliced speech, and $n \leq 8$ is set to reduce the memory overhead for loading HuDict.

2.2.3. *Grapheme-to-Hunit (G2H)*

We adopt the lightweight SoundChoice [34] Grapheme-to-Phoneme (G2P) model with Conformer-Transformer architecture [35] to train a sentence-level G2H model. As shown in Fig 1 (c), for paired speech $\{S_l, T_l\} \in D_l$, we obtain training data $\{T_l, H_l\}$ for G2H model by extracting Hunit sequence H_l from

S_l with the Hunit extractor. Note that H_l contains no consecutive duplicates, which differs from H_r in Fig 1 (b).

2.2.4. Audio Synthesis

As shown in Fig 1 (d), a text sample $T_l' \in D_l'$ is first converted to the Hunit sequence H_l' with the G2H model. Then H_l' is decomposed into H_l' n-grams with a divide-and-conquer algorithm shown in Algorithm 1. This algorithm is designed to maximize the averaged length of Hunit n-grams, as longer speech fragments are preferred for generating more fluent speech. The symbol $\times$ in line 10 denotes Cartesian Product. By applying mathematical induction, it can be proved that Algorithm 1 always returns Hunit n-gram sequences with maximized averaged n if such a sequence exists. An Hunit sequence is discarded if it cannot be decomposed into Hunit n-grams in HuDict with such a procedure. Finally, the resulting H_l' n-gram sequences are mapped to speech fragments by querying HuDict, and the speech fragments are concatenated into complete speech.

Algorithm 1 Decompose a Hunit sequence into Hunit n-grams

```
Input: $x$, the Hunit sequence
Output: $y$, list of Hunit n-gram sequences
Require: $\mathbb{S}$, the set of all Hunit n-grams in the HuDict
 1: function DECOMPOSE($x$)
 2:     $y \leftarrow []$
 3:     for $n \leftarrow 10$ to $3$ do
 4:         for $i \leftarrow 0$ to length($x$) $- n$ do
 5:             if $x[i : i + n] \in \mathbb{S}$ then
 6:                 $y_{pre} \leftarrow$ DECOMPOSE($x[0 : i]$)
 7:                 $y_{post} \leftarrow$ DECOMPOSE($x[i + n :]$)
 8:                 if $y_{pre}$ is $\varnothing$ or $y_{post}$ is $\varnothing$ then
 9:                     continue
10:                 Append ($y_{pre} \times [x[i : i + n]] \times y_{post}$) to y
11:         if length($y$) $\neq 0$ then
12:             return $y$
13:     return $\varnothing$
```

3. Experiments

3.1. Experiment Setup

3.1.1. Data

We conduct experiments by fine-tuning a HuBERT model pretrained on LIBRISPEECH with 10-hour paired data from different languages in COMMONVOICE dataset [36]. COMMONVOICE[1] is a multilingual speech corpus sourced primarily from Wikipedia articles. We select five languages in our experiments: Frisian, French, Dutch, German, and Turkish. We use the speech from the 960-hour LIBRISPEECH dataset [37] as D_r without utilizing the transcriptions. For each of the five languages in COMMONVOICE, we sample a 10-hour paired data split as D_l to create a low-resource setup, and D_l' is comprised of all available text transcriptions for that language. The number of words for each language in D_l' is listed in Table 1. We use the speech from the 960-hour LIBRISPEECH dataset [37] as D_r without utilizing the transcriptions. For each of the five languages in COMMONVOICE, we sample a 10-hour paired data split as D_l to create a low-resource setup, and D_l' is comprised of all available text transcriptions for that language. The number of words for each language in D_l' is listed in Table 1.

[1]https://commonvoice.mozilla.org/en/datasets

Table 1: *Number of words in D_l' for different languages*

	Frisian	French	Dutch	German	Turkish
# words	275 K	5.5 M	563 K	6.6 M	289 K

3.1.2. Hunit Extractor and G2H Model

We use the officially released `HuBERT-base-iter2` [2] as the backend for Hunit extractor. We fetch HuBERT latent representations from the ninth layer and extract clustered Hunits with a K-Means model trained on latent representations from D_r. The impact of number of K-Means clusters on model performance is investigated in Section 3.2.1. For the denoising module based on mode filters, we empirically adopt one mode filter with window size 3 followed by four mode filters with window size 5.

Following SoundChoice-G2P [34], the G2H model is a Conformer-Transformer encoder-decoder architecture comprised of 2 Conformer encoders and 2 Transformer decoders with hidden dimensions 512 and 8 heads. Instead of training on D_l from scratch, we use a released SoundChoice-G2P model[3] pretrained on D_r as initialization and fine-tune it on D_l with the same configurations as the sentence-level training stage of SoundChoice-G2P. Since the data in D_l is limited, it takes less than 1 hour to fine-tune the G2H model on a single GPU. Greedy decoding is adopted for faster inference.

3.1.3. HuBERT Model Fine-tuning

We carry out model fine-tuning with the FAIRSEQ [38] toolkit and adopt the same model used for Hunit extraction which is comprised of 12 transformer blocks with hidden dimensions 768 and 8 heads. We follow the `base_10h` fine-tuning setup in HuBERT but increase the steps from 10k to 100k since it takes significantly longer to converge on data from languages other than English as observed in our experiments. This is likely due to the language used for fine-tuning being different from the language used for pretraining. Fine-tuning takes two days on four GPUs for each language.

3.1.4. Decoding with Language Model

For language model rescoring, we train 4-gram language model trained on D_l' for each of the five languages. The hyperparameters for decoding are tuned on the dev set of each language. We use a beam size of 500. The LM weight and word insertion penalty for decoding are listed in Table 2.

Table 2: *Decoding hyperparameters for different languages*

	Frisian	French	Dutch	German	Turkish
LM weight	7.0	4.0	3.2	4.6	3.7
word insert.	-0.1	-0.6	-0.8	-0.8	-0.4

3.2. Experiment Results and analysis

We first conduct experiments on Frisian to investigate the impact of various hyperparameters.

3.2.1. Investigation on the number of K-Means clusters

Table 3 shows how the number of K-Means clusters used to extract Hunits affects the performance. The baseline model is trained by fine-tuning the pretrained HuBERT model on the Frisian 10-hour paired data. We conduct experiments

[2]https://dl.fbaipublicfiles.com/hubert/hubert_base_ls960.pt
[3]https://drive.google.com/drive/folders/1lbSjCKUit8H3FCzaDJmfBDJOkcDRH3XI?usp=sharing

on 100, 200, and 500 K-Means clusters. Intuitively, the speech fragments corresponding to the same Hunit n-gram are more acoustically similar if more clusters are applied for K-Means clustering. Results from `UniSplice-C100` and `UniSplice-C200` show that too few clusters lead to poor correlation between Hunits and the underlying acoustic units, resulting in low-quality synthetic data that degrades performance. With the number of clustering units increased to 500, `UniSplice-C500` achieves 18.3% and 18.0% relative improvement on Dev and Test sets, respectively, over baseline.

`Mixed Batch` refers to the inclusion of synthetic data in the same batch as real data. Comparing `UniSplice-C500` and `UniSplice-C500-S`, we observe slight degradation when `Mixed Batch` is not applied. This can be attributed to the greater variation in gradients between real and synthetic data batches, which may impede the model's convergence. Subsequent experiments employ the `Mixed Batch` strategy for model fine-tuning, along with 500 clusters for Hunit extraction.

Table 3: *WER(%) of different number of clusters*

System	Clusters	Mixed Batch	Dev / Test
Baseline	N/A	N/A	15.3 / 15.0
UniSplice-C100	100	✓	17.3 / 16.9
UniSplice-C200	200	✓	16.8 / 16.4
UniSplice-C500	500	✓	**12.5 / 12.3**
UniSplice-C500-S	500	✗	13.0 / 12.6

3.2.2. Investigation on the number of word tokens in D_l'

In table 4, we investigate the impact of the size of D_l'. The performance of the model keeps improving as the amount of text in D_l' increases from 50K words to 275K words. We also include results for data synthesis with a VITS [39] neural TTS system trained on D_l with ESPnet [40] following the multi-speaker recipe except that the sample rate is set to 16kHz. The results show that the VITS model trained with 10 hours of ASR data produces low-quality audio detrimental to ASR performance.

Table 4: *WER(%) comparison of different size of D_l'*

System	#words	Dev / Test
Baseline	N/A	15.3 / 15.0
VITS Neural TTS	275 K	17.1 / 16.7
UniSplice	275 K	**12.5 / 12.3**
	200 K	12.7 / 12.4
	100 K	13.4 / 12.7
	50 K	14.8 / 13.9

3.2.3. Validation on different langauges

Finally, we explore the best mixing ratio of real and synthetic data in Frisian and validate the effectiveness of the proposed methods in all languages. Since UniSplice introduces additional text for speech synthesis, we also report performance with a 4-gram LM rescoring. The results are shown in Table 5.

Adjusting the mixing ratio of real and synthetic data is implemented by repeating D_l or D_l' multiple times. E.g. The 1:2 mixing ratio means to repeat D_l' twice in each epoch before mixing real and synthetic data into batches. Results show the best mixing ratio to be 2:1 for Frisian, and we apply this ratio to other languages. Comparing the third and sixth rows, increasing the proportion of synthetic data is more harmful than increasing real data, which can be ascribed to the mismatch between the synthesized training data and the real test data.

The results in Table 5 indicate a consistent improvement across all languages with UniSplice. However, the relative improvement for Turkish (7%) is comparatively lower than other languages (10-20%) without LM rescoring. This is probably because English and the four non-Turkish languages belong to the Indo-European family and share similar acoustic characteristics. Although the unpaired text D_l' is significantly larger for French and German as shown in Table 1, the relative improvement is not substantial, which can be attributed to the size of HuDict being fixed. Notably, the application of LM rescoring results in larger relative improvements for all languages (15% for Turkish and 20-30% for other languages), highlighting the compatibility between UniSplice and LM rescoring.

Table 5: *WER(%) results on all languages*

Language	System	LM	Mixing Ratio	Dev / Test
Frisian	Baseline	✗	N/A	15.3 / 15.0
		✓		2.9 / 2.7
	UniSplice	✗	1 : 2	13.1 / 12.7
		✗	1 : 1	12.5 / 12.3
		✗	2 : 1	12.2 / 11.8
		✗	3 : 1	12.5 / 11.9
		✓	2 : 1	**2.0 / 2.0**
French	Baseline	✗	N/A	42.8 / 46.9
		✓		27.4 / 31.4
	UniSplice	✗	2:1	35.7 / 39.3
		✓		**21.1 / 24.1**
Dutch	Baseline	✗	N/A	20.9 / 22.1
		✓		13.4 / 11.9
	UniSplice	✗	2:1	17.9 / 18.7
		✓		**10.7 / 9.2**
German	Baseline	✗	N/A	36.5 / 39.1
		✓		19.0 / 20.7
	UniSplice	✗	2:1	33.0 / 35.1
		✓		**14.7 / 15.8**
Turkish	Baseline	✗	N/A	28.1 / 29.2
		✓		10.1 / 9.5
	UniSplice	✗	2:1	26.2 / 27.1
		✓		**8.6 / 7.9**

4. Conclusions

In this paper, we proposed UniSplice, a cross-lingual data splicing framework for low-resource speech recognition. UniSplice leverages denoised clustering units from HuBERT latent layers (Hunits) as universal phonetic units, which allows for cross-lingual data splicing. Our approach enables on-the-fly speech synthesis using text from low-resource languages and speech fragments from rich-resource languages, which can be used to improve the training of ASR models. Experiments on five languages from the COMMONVOICE dataset demonstrate the effectiveness of our proposed framework. By introducing cross-lingually spliced data during training, we achieved a 20-30% relative improvement in WER for four Indo-European languages and about 15% for Turkish with a 4-gram language model. These promising results suggest that UniSplice can be a valuable tool for building low-resource ASR systems.

5. Acknowledgements

This work was supported in part by China NSFC projects under Grants 62122050 and 62071288, and in part by Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102.

6. References

[1] A. Zeyer, K. Irie *et al.*, "Improved training of end-to-end attention models for speech recognition," in *Proc. Interspeech*, 2018, pp. 7–11.

[2] W. Chan, D. S. Park *et al.*, "SpeechStew: Simply mix all available speech recognition data to train one large neural network," in *Workshop on Machine Learning in Speech and Language Processing (MLSLP)*, 2021.

[3] A. Radford, J. W. Kim *et al.*, "Robust speech recognition via large-scale weak supervision," *ArXiv*, vol. abs/2212.04356, 2022.

[4] J. Li *et al.*, "Recent advances in end-to-end automatic speech recognition," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2022.

[5] J. Kunze, L. Kirsch *et al.*, "Transfer learning for speech recognition on a budget," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017, pp. 168–177.

[6] S. Toshniwal, T. N. Sainath *et al.*, "Multilingual speech recognition with a single end-to-end model," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 4904–4908.

[7] B. Li, R. Pang *et al.*, "Scaling end-to-end models for large-scale multilingual asr," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 1011–1018.

[8] S. Dalmia, R. Sanabria *et al.*, "Sequence-based multi-lingual low resource speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4909–4913.

[9] S. Tong, P. N. Garner, and H. Bourlard, "An Investigation of Deep Neural Networks for Multilingual Speech Recognition Training and Adaptation," in *Proc. Interspeech 2017*, 2017, pp. 714–718.

[10] N. T. Vu, D. Imseng *et al.*, "Multilingual deep neural network based acoustic modeling for rapid language adaptation," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 7639–7643.

[11] A. Baevski, Y. Zhou *et al.*, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12 449–12 460.

[12] W.-N. Hsu, B. Bolte *et al.*, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM TASLP*, vol. 29, pp. 3451–3460, 2021.

[13] S. Khorram, J. Kim *et al.*, "Contrastive siamese network for semi-supervised speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7207–7211.

[14] C. Yi, J. Wang *et al.*, "Applying wav2vec2.0 to speech recognition in various low-resource languages," *ArXiv*, vol. abs/2012.12121, 2020.

[15] A. Baevski, M. Auli, and A. rahman Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," *ArXiv*, vol. abs/1911.03912, 2019.

[16] J. Zhao and W.-Q. Zhang, "Improving automatic speech recognition performance for low-resource languages with self-supervised models," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1227–1241, 2022.

[17] Z. Ma, Z. Zheng *et al.*, "MT4SSL: Boosting Self-Supervised Speech Representation Learning by Integrating Multiple Targets," *Proc. of InterSpeech*, 2023.

[18] J. Li, R. Zhao *et al.*, "Developing RNN-T Models Surpassing High-Performance Hybrid Models with Customization Capability," in *Proc. Interspeech 2020*, 2020, pp. 3590–3594.

[19] Y. Deng, R. Zhao *et al.*, "Improving RNN-T for Domain Scaling Using Semi-Supervised Training with Neural TTS," in *Proc. Interspeech 2021*, 2021, pp. 751–755.

[20] A. Tjandra, S. Sakti, and S. Nakamura, "Listening while speaking: Speech chain by deep learning," in *Proc. IEEE ASRU*, 2017, pp. 301–308.

[21] F. Yue, Y. Deng *et al.*, "Exploring machine speech chain for domain adaptation and few-shot speaker adaptation," *ArXiv*, vol. abs/2104.03815, 2021.

[22] X. Zheng, Y. Liu *et al.*, "Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5674–5678.

[23] S. Murthy, D. Sitaram, and S. Sitaram, "Effect of TTS generated audio on OOV detection and word error rate in ASR for low-resource languages," in *Proc. Interspeech*, 2018, pp. 1026–1030.

[24] Z. Chen, Y. Zhang *et al.*, "Tts4pretrain 2.0: Advancing the use of text and speech in ASR pretraining with consistency and contrastive losses," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7677–7681.

[25] W. Wang, Z. Zhou *et al.*, "Towards data selection on TTS data for children's speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6888–6892.

[26] A. Graves, "Sequence transduction with recurrent neural networks," in *Proc. ICML*, 2012.

[27] J. Xu, X. Tan *et al.*, "LRSpeech: Extremely low-resource speech synthesis and recognition," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2802–2812.

[28] Y. Ren, X. Tan *et al.*, "Almost unsupervised text to speech and automatic speech recognition," in *Proc. ICML*, 2019, pp. 5410–5419.

[29] A. H. Liu, T. Tu *et al.*, "Towards unsupervised speech recognition and synthesis with quantized speech representation learning," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7259–7263, 2020.

[30] Y.-A. Chung, Y. Wang *et al.*, "Semi-supervised training for improving data efficiency in end-to-end speech synthesis," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6940–6944, 2019.

[31] R. Zhao, J. Xue *et al.*, "On addressing practical challenges for RNN-Transducer," in *Proc. IEEE ASRU*, 2021, pp. 526–533.

[32] J. Devlin, M.-W. Chang *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[33] A. Vaswani, N. Shazeer *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[34] A. Ploujnikov and M. Ravanelli, "SoundChoice: Grapheme-to-phoneme models with semantic disambiguation," in *Proc. Interspeech*, 2022, pp. 486–490.

[35] A. Gulati, J. Qin *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[36] R. Ardila, M. Branson *et al.*, "Common voice: A massively-multilingual speech corpus," in *International Conference on Language Resources and Evaluation*, 2020, pp. 4218–4222.

[37] V. Panayotov, G. Chen *et al.*, "Librispeech: an ASR corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.

[38] M. Ott, S. Edunov *et al.*, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proc. NAACL-HLT: Demonstrations*, 2019, pp. 48–53.

[39] J. Kim, J. Kong, and J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," in *Proc. ICML*, 2021, pp. 5530–5540.

[40] S. Watanabe, T. Hori *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.