

SPEECHAIN: A SPEECH TOOLKIT FOR LARGE-SCALE MACHINE SPEECH CHAIN

Heli Qi¹, Sashi Novitasari^{1*}, Andros Tjandra^{1†}, Sakriani Sakti², Satoshi Nakamura¹

¹Nara Institute of Science and Technology, Japan

²Japan Advanced Institute of Science and Technology, Japan

ABSTRACT

This paper introduces SpeeChain, an open-source Pytorch-based toolkit designed to develop the machine speech chain for large-scale use. This first release focuses on the TTS→ASR chain, a core component of the machine speech chain, that refers to the TTS data augmentation by unspoken text for ASR. To build an efficient pipeline for the large-scale TTS→ASR chain, we implement easy-to-use multi-GPU batch-level model inference, multi-dataloader batch generation, and on-the-fly data selection techniques. In this paper, we first explain the overall procedure of the TTS→ASR chain and the difficulties of each step. Then, we present a detailed ablation study on different types of unlabeled data, data filtering thresholds, batch composition, and real-synthetic data ratios. Our experimental results on *train_clean_460* of LibriSpeech demonstrate that our TTS→ASR chain can significantly improve WER in a semi-supervised setting.[‡]

Index Terms— Open-source speech processing toolkit, machine speech chain, ASR, TTS, semi-supervised learning

1. INTRODUCTION

End-to-end (E2E) automatic speech recognition (ASR) [1–4] and text-to-speech synthesis (TTS) [5–7] have achieved great success due to recent advances in deep learning. ASR and TTS are considered symmetric to each other, as they share the same type of training data with flipped inputs and outputs. Despite the close relationship, research on both areas progressed more or less independently until [8, 9] proposed the machine speech chain. Research on E2E ASR-TTS has received more attention ever since [10, 11].

Machine speech chain is a closed-loop architecture based on deep learning that connects E2E ASR and TTS models. One core application of the machine speech chain is the TTS→ASR chain, where E2E ASR models are trained on synthetic speech produced by E2E TTS models using unspoken text. The initial experiments of the TTS→ASR chain on single-speaker [8] and multi-speaker datasets [12] were successful, which has led to an interest in training E2E ASR

models with synthetic speech produced by multi-speaker TTS models in the community [13–17]. For multi-speaker TTS, a speaker embedding model is introduced to help TTS systems capture various speaker identities and generate synthetic speech with different speaking styles. Moreover, [16] proposed two additional embedding encoders to extract local and global utterance embeddings for better TTS fidelity. Apart from speaker diversity, [16–19] provided TTS systems with unlabeled out-of-domain text to help ASR models overcome linguistic mismatch.

In the speech processing community, many excellent open-source toolkits [20–25] are developed to support ASR and TTS models, and they have achieved comparable performance to the state-of-the-art. However, most of these all-in-one toolkits develop the ASR and TTS components independently with little linkage. To bridge the gap and establish the connection between ASR and TTS, we present SpeeChain, a toolkit that offers researchers and developers an efficient pipeline of the TTS→ASR chain. SpeeChain supports various modular functions, such as fast model inference by multiple GPUs, uniform storage structure for real and synthetic data, and on-the-fly data filtering and loading. Furthermore, we provide intensively-tuned hyper-parameters for our ASR and TTS models to facilitate research on joint ASR-TTS.

The rest of the paper is organized as follows: Section 2 describes the procedure of the TTS→ASR chain. Section 3 explains the technical solutions of our toolkit to the TTS→ASR chain. Section 4 presents our experimental results. In Section 5, we conclude this work and introduce our future work.

2. TTS DATA AUGMENTATION FOR ASR (TTS→ASR CHAIN)

The procedure of the TTS→ASR chain is presented in Fig. 1.

2.1. Base ASR&TTS training

In the beginning, base ASR and TTS models are trained on the same labeled dataset. The performance of the base models largely determines the quality of synthetic speech and its WER evaluation in the next stage. However, due to the incomplete distribution of the limited labeled data, it's hard to train

*Sashi Novitasari is currently with IBM Tokyo Research Lab.

†Andros Tjandra is currently with Meta AI.

‡We will release our toolkit upon publication.

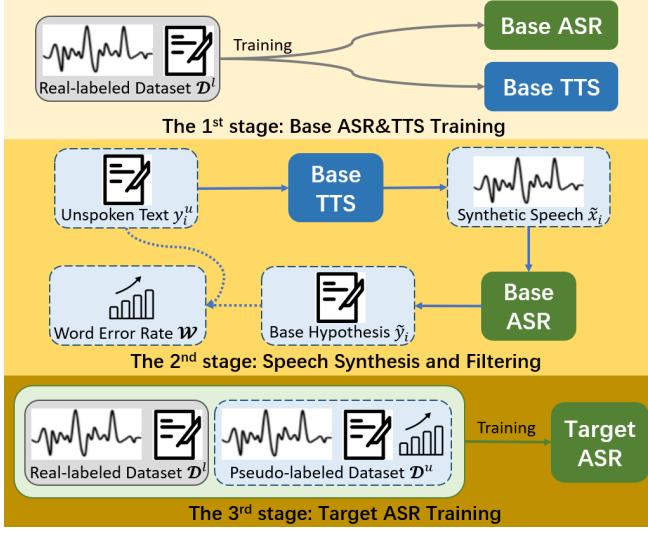


Fig. 1. The overview of the TTS→ASR chain.

a base model that performs reasonably well, especially for multi-speaker TTS. Commonly-used strategies include elongating the warm-up stage, lowering the learning rate, raising dropout rates [26], shrinking the model parameters, etc.

2.2. Speech synthesis and filtering

In this stage, given an unlabeled dataset $\mathcal{D}^u = \{y_1^u, \dots, y_{|\mathcal{D}^u|}^u\}$, the base TTS model θ_{TTS}^{base} generates synthetic speech \tilde{x}_i for unspoken text y_i^u ($i \in \{1, \dots, |\mathcal{D}^u|\}$) as

$$\tilde{x}_i = \arg \max_z \prod_{t=1}^T p(z_t | \tilde{x}_{i,0:t-1}, y_i^u; \theta_{TTS}^{base}). \quad (1)$$

The synthetic speech \tilde{x}_i is combined with unspoken text y_i^u to form the pseudo-labeled dataset $\mathcal{D}^u = \{(\tilde{x}_1, y_1^u), \dots, (\tilde{x}_{|\mathcal{D}^u|}, y_{|\mathcal{D}^u|}^u)\}$. However, despite the considerable effort in training the base TTS model, there may still be some errors in the synthetic speech (e.g., mispronunciations, silences, repeating phrases) because of the limitation on the amount of available labeled data. Data filtering is usually done to reduce the errors in the synthetic speech before the final stage. The TTS→ASR chain enables us to filter the synthetic speech via the base word error rate (WER) \mathcal{W}^{base} as

$$\tilde{y}_i = \arg \max_z \prod_{t=1}^T p(z_t | \tilde{y}_{i,0:t-1}, \tilde{x}_i; \theta_{ASR}^{base}), \quad (2)$$

$$\mathcal{W}_i^{base} = \frac{\text{edit_distance}(\tilde{y}_i, y_i^u)}{|y_i^u|} \quad (3)$$

where \tilde{y}_i is the hypothesis made by the base ASR model for \tilde{x}_i , $\text{edit_distance}()$ is the function that measures the minimum edit distance between the word sequences of \tilde{y}_i and y_i^u , and $|y_i^u|$ is the number of words in y_i^u .

2.3. Target ASR training

In the final stage, the synthetic speech whose \mathcal{W}^{base} is larger than the given threshold will be first filtered out from the pseudo-labeled dataset \mathcal{D}^u . Then, the union of the real-labeled dataset $\mathcal{D}^l = \{(x_1^l, y_1^l), \dots, (x_{|\mathcal{D}^l|}^l, y_{|\mathcal{D}^l|}^l)\}$ and \mathcal{D}^u gives us an enlarged one $\mathcal{D} = \mathcal{D}^l \cup \mathcal{D}^u$ where we could train our target ASR model θ_{ASR}^{target} . During training, a single batch $\mathcal{B} = \mathcal{B}^l \cup \mathcal{B}^u$ is composed of real-labeled data $\mathcal{B}^l = \{(x_1^l, y_1^l), \dots, (x_{|\mathcal{B}^l|}^l, y_{|\mathcal{B}^l|}^l)\}$ and pseudo-labeled data $\mathcal{B}^u = \{(\tilde{x}_1, y_1^u), \dots, (\tilde{x}_{|\mathcal{B}^u|}, y_{|\mathcal{B}^u|}^u)\}$ proportionally fetched from \mathcal{D}^l and \mathcal{D}^u . Two training losses \mathcal{L}^l and \mathcal{L}^u are calculated on \mathcal{B}^l and \mathcal{B}^u respectively as

$$\mathcal{L}^l = -\frac{1}{|\mathcal{B}^l|} \sum_{i=1}^{|\mathcal{B}^l|} \sum_{t=1}^T \log p(y_{i,t}^l | y_{i,0:t-1}^l, x_i^l; \theta_{ASR}^{target}), \quad (4)$$

$$\mathcal{L}^u = -\frac{1}{|\mathcal{B}^u|} \sum_{i=1}^{|\mathcal{B}^u|} \sum_{t=1}^T \log p(y_{i,t}^u | y_{i,0:t-1}^u, \tilde{x}_i; \theta_{ASR}^{target}), \quad (5)$$

where both speech x_i and prefix tokens $y_{0:t-1}$ are used as input based on the teacher-forcing technique. Finally, the target ASR model is optimized by the overall loss $\mathcal{L} = (1 - \lambda)\mathcal{L}^l + \lambda\mathcal{L}^u$ controlled by a manually-set weight λ .

3. TOOLKIT CHARACTERISTICS

While the base model training follows the ordinary supervised training scheme as done in many toolkits, the last two stages need more specific designs to improve pipeline efficiency. Our toolkit implements the following unique functionalities to simplify the pipeline of the TTS→ASR chain.

3.1. Fast model inference with uniform data structure

Due to the auto-regressive nature of E2E ASR&TTS models, model inference consumes plenty of time for large amounts of unlabeled data in the second stage. Our toolkit implements batch-level model inference by `torch.nn.parallel.DistributedDataParallel` that simultaneously utilizes multiple GPUs to speed up model inference. During inference, we partition unlabeled data into multiple non-overlapping segments. Each GPU holds an exclusive process that decodes a batch of unlabeled data in its responsible part at one inference step. After all the GPUs finish their assignments, we combine all the inference results to form the final pseudo-labeled dataset. Fig 2 shows an illustration of TTS decoding by multiple GPUs.

During model inference, we decouple the model calculation from the data storage so that the model hypotheses and inference metadata (e.g., WER, ASR confidence) could have the same structure as the real datasets. The uniform data structure for real and pseudo datasets dramatically reduces the effort required for us to conduct the experiments.

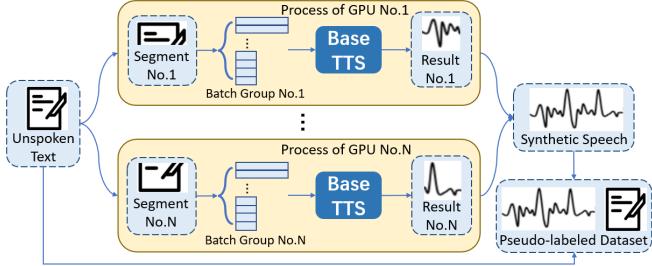


Fig. 2. The flowchart of the batch-level TTS inference by multiple GPUs. We group unlabeled text by the summation of their lengths to ensure the same number of synthetic frames in each batch. ASR beam searching shares a similar mechanism.

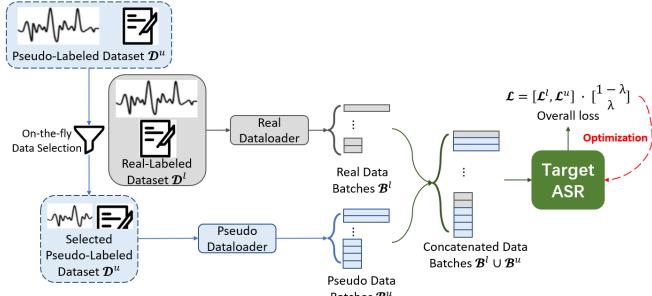


Fig. 3. The flowchart of data loading by two dataloaders with on-the-fly data selection. We concatenate real and pseudo batches with similar utterance lengths to reduce the number of model forwarding operations.

3.2. Batch generation by multiple dataloaders

When training the target ASR model in the final stage, it's not a good idea to mix the real-labeled dataset with the pseudo-labeled one and dynamically fetch data from the mixed one. In most cases, unlabeled data is way more than labeled data, which results in plenty of data batches dominated by pseudo-labeled data. Therefore, it's hard to control the gradient direction of each data batch, and training becomes unstable.

Our toolkit utilizes multiple `torch.utils.data.DataLoader` to separately fetch speech-text pairs from different datasets, which generates the batches with a static real-pseudo data ratio, as shown in Fig. 3. The static data ratio not only regularizes the gradient direction calculated by each batch but also makes an evident real-pseudo composition of each batch. The evident composition allows us to design more advanced algorithms for better usage of pseudo-labeled data during training.

3.3. On-the-fly data selection for pseudo-labeled data

Our toolkit decouples the batch generation from the dataset access (i.e., `torch.utils.data.Dataset`) so that we can easily change the accessible speech-text pairs of each dataloader by

Work (Model)	dev_clean	test_clean	test_other
Baseline on LibriSpeech-train_clean_100			
[16] (LAS [2])	-	12.46%	34.00%
[27] (TDS [4])	14.00%	14.85%	39.95%
[28] (Transformer [3])	12.20%	12.90%	31.10%
Ours (Transformer)	11.90%	12.20%	30.19%
Topline on LibriSpeech-train_clean_100 & LibriSpeech-train_clean_360			
[16] (LAS)	-	6.30%	22.41%
[18] (LAS)	-	6.50%	-
Ours (Transformer)	5.08%	5.65%	15.91%
Topline on LibriSpeech-train_clean_100 & LibriTTS-train_clean_360			
Ours (Transformer)	6.14%	6.57%	19.50%

Table 1. Word error rate (WER) of ASR baseline and topline without the language model compared to the literature. For the baseline, we use the vocabulary with 1k BPE tokens; for the topline, we use the vocabulary with 5k BPE tokens. We trained two topline models on LibriSpeech and LibriTTS to make a fair comparison with our TTS→ASR chain.

the given metadata during training. We also provide metadata distribution histograms to help users choose the proper thresholds to filter out the low-quality pseudo-labeled data.

4. EXPERIMENTS

4.1. ASR baseline and topline of supervised learning

Our ASR models are based on Speech-Transformer [3], where the encoder and decoder are composed of 12 and 6 Transformer layers, respectively. We make two model configurations for ASR baseline ($d_{model}=256$, $n_{head}=4$) and topline ($d_{model}=512$, $n_{head}=8$). The two configurations share the same 2048-dimensional feed-forward layers. We uniformly do ASR decoding by 16 beams without a CTC branch for all experiments. The results of our baseline and topline models are shown in Table 1.

4.2. Base TTS model training

As mentioned in [15], LibriSpeech [29] is unsuitable for training TTS models, so we downsampled LibriTTS [30] to 16kHz for both base TTS training (*train_clean_100*) and speech synthesis (*train_clean_360*). Our TTS model is based on MultiSpeech [7], where the encoder and decoder are composed of the same 6 Transformer layers ($d_{model}=512$, $n_{head}=8$, $d_{ff}=2048$). The encoder directly consumes characters as the input.

We observed that TTS is data-hungry than ASR. Therefore, we turn all the lowercase letters into capital versions and remove punctuation marks other than commas, periods, and single quotes to help TTS convergence. We make our base

Filtering Thresholds	dev_clean	test_clean	test_other
TTS→ASR Chain by LibriTTS- <i>train_clean_360</i>			
Base WER<10%	9.27%	10.22%	27.12%
Base WER<20%	8.48%	9.30%	25.90%
Base WER<30%	8.11%	8.68%	24.82%
Base WER<50%	7.83%	8.76%	25.65%
Base WER<100%	7.81%	8.45%	24.50%
No Base WER Filtering	7.32%	8.07%	24.17%
ASR Self-Training by LibriSpeech- <i>train_clean_360</i>			
Top20% Confidence	10.23%	10.80%	26.28%
Top40% Confidence	9.04%	9.75%	24.26%
Top60% Confidence	8.86%	9.47%	23.44%
Top80% Confidence	8.51%	9.36%	22.76%
No Confidence Filtering	8.67%	9.33%	22.66%

Table 2. WER of target ASR models with different filtering thresholds.

TTS model generate four consecutive frames at each decoding step to speed up the speech synthesis. We don't adopt the neural vocoder and use the same 80-dimensional log-Mel spectral features extracted by 50ms window length and 10ms window shift for both ASR and TTS training. Also, we found it helpful for base TTS training if we scale up the encoder and decoder embedding vectors by $\sqrt{d_{model}}$ before adding them to the positional encodings. During TTS speech synthesis, we randomly sample speakers from LibriTTS-*train_clean_100* as reference speakers for our base TTS model.

4.3. WER filtering by the base ASR model

Our target ASR models have the same configuration as our topline model. We vary the base WER filtering thresholds as shown in Table 2, where λ is uniformly set to 0.5. We compare our TTS→ASR chain with ASR self-training [27], where the base ASR model generates pseudo text for the untranscribed speech to enlarge the training set on its own. In the experiments of ASR self-training, we used LibriSpeech-*train_clean_360* as the unlabeled data and the ASR decoding confidence as the filtering criterion.

The results show that the base WER filtering instead reduces the ASR improvement. We found that the WERs given by the base ASR failed to evaluate the quality of synthetic speech because of the mismatch between real speech and synthetic speech. However, the target ASR still benefits from training on synthetic speech even though there are some errors in them. We hypothesize that some errors in the synthetic speech may act as data augmentations, e.g., silence could force ASR to predict the missing parts.

We also observed that the improvement of the TTS→ASR chain on *test_other* is smaller than ASR self-training. We hypothesize that there is an acoustic mismatch of speech data between *clean* and *other* datasets. Since our TTS model is trained only on *clean* data, the synthetic speech fails to provide the target ASR model with much acoustic details as the

Batch Composition	dev_clean	test_clean	test_other
Topline on LibriSpeech- <i>train_clean_100</i> & LibriTTS- <i>train_clean_360</i>			
Static R2S Ratio	6.37%	6.82%	19.75%
Dynamic R2S Ratio	6.14%	6.57%	19.50%
10% Base WER Filtering (2:1)			
Static R2S Ratio	9.27%	10.22%	27.12%
Dynamic R2S Ratio	10.59%	11.39%	28.97%
No Base WER Filtering (1:2)			
Static R2S Ratio	7.32%	8.07%	24.17%
Dynamic R2S Ratio	7.75%	8.54%	24.93%

Table 3. WER of target ASR models with different batch compositions. R2S Ratio stands for the real-to-synthetic ratio of each training batch. The numbers in the brackets indicate the R2S ratio of the entire training set.

real untranscribed speech does.

4.4. The effect of real-synthetic data composition

To study the influence of batch compositions on ASR training mentioned in Sec 3.2, we conducted a contrast experiment in different batch generations shown in Table 3. We considered three scenarios where the training set contains 100%, 66%, and 33% of real speech. In each scenario, the batch compositions are either static by two dataloaders or dynamic by one dataloader. Our results indicate that as long as synthetic speech exists in the training set, it's necessary to include a certain amount of real speech in each batch to regularize the gradient directions. But for topline training on two real datasets, it's better to mix them so that the model training could enjoy more randomness on the batch generation.

5. CONCLUSION AND FUTURE WORK

This paper introduces SpeeChain, a novel toolkit designed for the large-scale machine speech chain. This first version provides an easy-to-use pipeline of TTS data augmentation for semi-supervised ASR. Our extensive experiments demonstrate that semi-supervised ASR models carefully developed via the TTS→ASR chain could achieve performance close to the supervised one trained on real-labeled data. In our future work, we will implement various advanced ASR and TTS models and develop additional applications of the machine speech chain, such as the ASR→TTS chain for semi-supervised TTS and online speech chain learning for joint ASR-TTS optimization.

6. ACKNOWLEDGEMENTS

Part of this work is supported by JSPS KAKENHI (grant number JP21H05054 and JP21H03467) and Google AI grant.

7. REFERENCES

- [1] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, “Attention-based models for speech recognition,” *Advances in neural information processing systems*, vol. 28, 2015.
- [2] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.
- [3] Linhao Dong, Shuang Xu, and Bo Xu, “Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5884–5888.
- [4] Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert, “Sequence-to-sequence speech recognition with time-depth separable convolutions,” *arXiv preprint arXiv:1904.02619*, 2019.
- [5] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al., “Natural tts synthesis by conditioning wavenet on mel spectrogram predictions,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.
- [6] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu, “Neural speech synthesis with transformer network,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33-01, pp. 6706–6713.
- [7] Mingjian Chen, Xu Tan, Yi Ren, Jin Xu, Hao Sun, Sheng Zhao, Tao Qin, and Tie-Yan Liu, “Multispeech: Multi-speaker text to speech with transformer,” *arXiv preprint arXiv:2006.04664*, 2020.
- [8] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, “Listening while speaking: Speech chain by deep learning,” in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 301–308.
- [9] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, “Machine speech chain,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 976–989, 2020.
- [10] Yi Ren, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu, “Almost unsupervised text to speech and automatic speech recognition,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 5410–5419.
- [11] Gary Wang, Andrew Rosenberg, Zhehuai Chen, Yu Zhang, Bhuvana Ramabhadran, Yonghui Wu, and Pedro Moreno, “Improving speech recognition using consistent predictions on synthesized speech,” in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7029–7033.
- [12] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, “Machine speech chain with one-shot speaker adaptation,” *arXiv preprint arXiv:1803.10525*, 2018.
- [13] Nick Rossenbach, Mohammad Zeineldeen, Benedikt Hilmes, Ralf Schlüter, and Hermann Ney, “Comparing the benefit of synthetic training data for various automatic speech recognition architectures,” in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 788–795.
- [14] Jason Li, Ravi Gadde, Boris Ginsburg, and Vitaly Lavrukhin, “Training neural speech recognition systems with synthetic speech augmentation,” *arXiv preprint arXiv:1811.00707*, 2018.
- [15] Nick Rossenbach, Albert Zeyer, Ralf Schlüter, and Hermann Ney, “Generating synthetic audio data for attention-based speech recognition systems,” in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7069–7073.
- [16] Andrew Rosenberg, Yu Zhang, Bhuvana Ramabhadran, Ye Jia, Pedro Moreno, Yonghui Wu, and Zelin Wu, “Speech recognition with augmented synthesized speech,” in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 996–1002.
- [17] Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara, “Multi-speaker sequence-to-sequence speech synthesis for data augmentation in acoustic-to-word speech recognition,” in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6161–6165.
- [18] Fengpeng Yue, Yan Deng, Lei He, and Tom Ko, “Exploring machine speech chain for domain adaptation and few-shot speaker adaptation,” *arXiv preprint arXiv:2104.03815*, 2021.
- [19] Murali Karthick Baskar, Lukáš Burget, Shinji Watanabe, Ramon Fernandez Astudillo, et al., “Eat: Enhanced asr-tts for self-supervised speech recognition,” in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6753–6757.
- [20] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al., “Espnet: End-to-end speech processing toolkit,” *arXiv preprint arXiv:1804.00015*, 2018.
- [21] Tomoki Hayashi, Ryuichi Yamamoto, Katsuki Inoue, Takenori Yoshimura, Shinji Watanabe, Tomoki Toda, Kazuya Takeda, Yu Zhang, and Xu Tan, “Espnet-tts: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit,” in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7654–7658.
- [22] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli, “fairseq: A fast, extensible toolkit for sequence modeling,” *arXiv preprint arXiv:1904.01038*, 2019.
- [23] Changhan Wang, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Ann Lee, Peng-Jen Chen, Jiatao Gu, and Juan Pino, “fairseq s^c: 2: A scalable and integrable speech synthesis toolkit,” *arXiv preprint arXiv:2109.06912*, 2021.
- [24] Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuany Zhong, et al., “Speechbrain: A general-purpose speech toolkit,” *arXiv preprint arXiv:2106.04624*, 2021.
- [25] Hui Zhang, Tian Yuan, Junkun Chen, Xintong Li, Renjie Zheng, Yuxin Huang, Xiaojie Chen, Enlei Gong, Zeyu Chen, Xiaoguang Hu, et al., “Paddlespeech: An easy-to-use all-in-one speech toolkit,” *arXiv preprint arXiv:2205.12007*, 2022.
- [26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [27] Jacob Kahn, Ann Lee, and Awni Hannun, “Self-training for end-to-end speech recognition,” in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7084–7088.
- [28] Yosuke Higuchi, Niko Moritz, Jonathan Le Roux, and Takaaki Hori, “Momentum pseudo-labeling: Semi-supervised asr with continuously improving pseudo-labels,” *IEEE Journal of Selected Topics in Signal Processing*, 2022.
- [29] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, “Librispeech: an asr corpus based on public domain audio books,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.
- [30] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu, “Libritts: A corpus derived from librispeech for text-to-speech,” *arXiv preprint arXiv:1904.02882*, 2019.