SpeeChain: A Scalable Framework for Machine Speech Chains

Heli Qi[1,a]*, Bagus Tris Atmaja*[1]*, Sashi Novitasari*[1,b]*, Andros Tjandra*[1,c]*, Sakriani Sakti*[1]*, Satoshi Nakamura*[1,d]

[1] Nara Institute of Science and Technology, Japan

qi.heli.qi9@is.naist.jp, bagus.tris@naist.ac.jp, sashi.novitasari.si3@is.naist.ac.jp, tjandra.ai6@is.naist.ac.jp, ssakti@is.naist.jp, s-nakamura@is.naist.jp

Abstract

This paper introduces SpeeChain, an open-source PyTorch-based framework for developing and benchmarking a machine speech chain. The TTS→ASR chain augments ASR training by generating synthetic speech from unspoken text using a TTS model. The ASR→TTS chain leverages pseudo-labels from ASR data to train TTS models. To build an efficient pipeline for large-scale speech processing, we implement easy-to-use multi-GPU batch-level model inference, multi-dataloader batch generation, and on-the-fly data selection. We describe the overall procedure for the TTS→ASR and ASR→TTS chains and compare training with and without the chains. Our experimental results demonstrate that the TTS→ASR chain can substantially improve WER in a semi-supervised setting. Meanwhile, the ASR→TTS chain has comparable objective spectral metrics to the supervised baseline while improving the duration ratio, demonstrating the potential of pseudo-labeled ASR data for TTS augmentation.

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

1. Introduction

The machine speech chain is a closed-loop deep-learning architecture that connects end-to-end (E2E) ASR and TTS models [1]. One core application of the machine speech chain is the TTS→ASR chain, in which E2E ASR models are trained on synthetic speech generated by E2E TTS models from unspoken text. The initial experiments of the TTS→ASR chain on single-speaker [2] and multi-speaker datasets [3] 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 [4, 5, 6, 7, 8, 9]. For multi-speaker TTS, a speaker embedding model is introduced to enable TTS systems to capture diverse speaker identities and generate synthetic speech with different speaking styles.

In the speech processing community, many excellent open-source toolkits [10, 11, 12, 13, 14, 15] have been 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 between them. To bridge the gap between ASR and TTS, we present SpeeChain, an extended and scalable framework that provides researchers and developers with an efficient pipeline for the TTS→ASR chain. SpeeChain supports various modular functions, including fast model inference across multiple GPUs, a uniform storage structure for real and synthetic data, and on-the-fly data filtering and loading. Earlier machine speech chain studies did not release their code or toolkit; SpeeChain provides an open-source PyTorch implementation of the machine speech chain for the community, available at `https://github.com/human-ai-lab/speechain`. Furthermore, we provide fixed, documented recipe configurations for our ASR and TTS models, facilitating reproducible research on joint ASR and TTS systems.

In the complementary direction, the ASR→TTS chain uses an ASR model to generate pseudo-labeled transcriptions from untranscribed speech, thereby providing additional text-speech pairs to augment TTS training. While TTS→ASR has been extensively studied [5, 8, 16], the ASR→TTS chain for semi-supervised TTS remains less explored. To bridge this gap, we also extend SpeeChain to support the ASR→TTS chain.

Our contributions in this paper are threefold:

- We present a unified and scalable framework that enables controlled, bidirectional evaluation of TTS→ASR and ASR→TTS speech chains under a consistent protocol.
- We show that, in our setting, filtering pseudo-labels by base ASR WER does not improve TTS→ASR results and the best result is obtained without filtering; because filtering also removes data, this may reflect the benefit of synthetic data quantity and diversity.
- We demonstrate that ASR→TTS augmentation improves duration modeling without degrading spectral metrics, suggesting that pseudo-labeled speech can mainly help duration prediction.

2. The Machine Speech Chain

2.1. TTS→ASR Chain

The TTS→ASR chain is shown in Fig. 1. In contrast, the ASR→TTS chain can be viewed as the inverse process of this system. For each chain, there are three main stages: (i) base-model training, (ii) speech synthesis or pseudo-label generation and filtering, and (iii) target ASR or TTS training. We detail these stages for the TTS→ASR chain below.

Initially, base ASR and TTS models are trained on labeled data of the same size (LS-100 for ASR and LT-100 for TTS in our experiments). 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 limited distribution of labeled data, it is hard to train a base model that performs reasonably well, especially for multi-speaker TTS. Commonly used strategies include elongating the warm-up stage, lowering

[a] Heli Qi is currently with Waseda University.
[b] Sashi Novitasari is currently with IBM Tokyo Research Lab.
[c] Andros Tjandra is currently with Meta AI.
[d] Satoshi Nakamura is currently with CUHK Shenzhen.

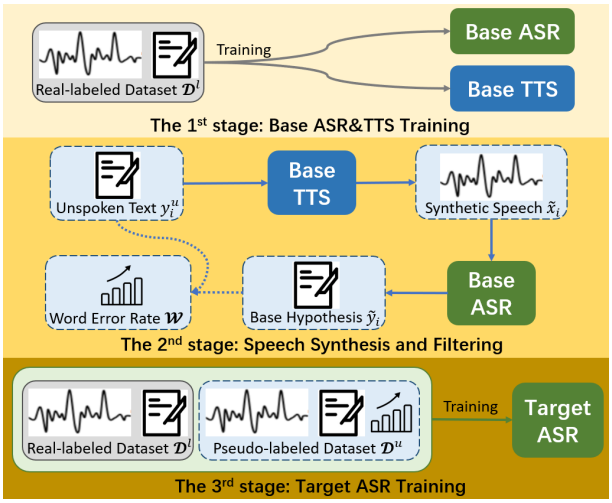


Figure 1: *Overview of the TTS→ASR chain.*

the learning rate, increasing dropout rates [17], and shrinking the model's parameter count.

In this stage, given an unlabeled dataset $\mathcal{D}^u = \{y_1^u, \ldots, y_{|\mathcal{D}^u|}^u\}$, the base TTS model θ_{TTS}^{base} generates synthetic speech $\tilde{x}_i$ for unspoken text y_i^u ($i \in \{1, \ldots, |\mathcal{D}^u|\}$) 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 $\tilde{\mathcal{D}}^u = \{(\tilde{x}_1, y_1^u), \ldots, (\tilde{x}_{|\mathcal{D}^u|}, y_{|\mathcal{D}^u|}^u)\}$. Despite considerable effort in training the base TTS model, errors in the synthetic speech (e.g., mispronunciations, silences, repeated phrases) may remain due to the limited amount of labeled data. Data filtering is typically performed to reduce errors in synthetic speech prior to 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

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

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

where $\tilde{y}_i$ is the hypothesis made by the base ASR model for $\tilde{x}_i$, $edit_distance()$ 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.

In the final stage, the synthetic speech whose $\mathcal{W}^{base}$ exceeds the given threshold will be filtered out of the pseudo-labeled dataset $\tilde{\mathcal{D}}^u$. Then, the union of the real-labeled dataset $\mathcal{D}^l = \{(x_1^l, y_1^l), \ldots, (x_{|\mathcal{D}^l|}^l, y_{|\mathcal{D}^l|}^l)\}$ and $\tilde{\mathcal{D}}^u$ gives us an enlarged one $\mathcal{D} = \mathcal{D}^l \cup \tilde{\mathcal{D}}^u$, 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 $\mathcal{B}^l = \{(x_1^l, y_1^l), \ldots, (x_{|\mathcal{B}^l|}^l, y_{|\mathcal{B}^l|}^l)\}$ and pseudo-labeled data $\mathcal{B}^u = \{(\tilde{x}_1, y_1^u), \ldots, (\tilde{x}_{|\mathcal{B}^u|}, y_{|\mathcal{B}^u|}^u)\}$ proportionally fetched from $\mathcal{D}^l$ and $\tilde{\mathcal{D}}^u$. 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 controlled by a manually-set weight λ,

$$\mathcal{L} = (1 - \lambda)\mathcal{L}^l + \lambda\mathcal{L}^u. \quad (6)$$

2.2. ASR→TTS Chain

The ASR→TTS chain is the inverse of the TTS→ASR chain. The base ASR and TTS models are first trained on labeled data (Sec. 5.1). Then, the base ASR model transcribes unlabeled speech to form the pseudo-labeled dataset, which augments TTS training. The same data-filtering, multi-dataloader batching, and weighted loss (λ) mechanisms from the TTS→ASR chain apply.

3. FRAMEWORK CHARACTERISTICS

While the base model is trained using the standard supervised training scheme in many toolkits, the last two stages require more tailored designs to improve pipeline efficiency. Our framework implements the following unique functionalities to simplify the TTS→ASR pipeline.

3.1. Fast model inference with uniform data structure

Due to the auto-regressive nature of E2E ASR & TTS models, inference time can be lengthy when large amounts of unlabeled data are used in the second stage. Our framework 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 runs an exclusive process that decodes a batch of unlabeled data in its assigned portion at each inference step. After all GPUs have completed their assignments, we combine their inference results to form the final pseudo-labeled dataset. Distributed inference reduces synthetic data generation time as more GPUs are used, enabling large-scale augmentation. For reference, all TTS→ASR experiments (base and target models, synthesis, and self-training) were run on NVIDIA RTX A5000 GPUs (24 GB each); the target TTS→ASR recipe used three GPUs and completed within five hours, and the base autoregressive TTS recipe completes within 23 hours. The Conformer ASR model and all ASR→TTS experiments were run on an NVIDIA RTX 5090 GPU. These figures are implementation references rather than hardware-independent benchmarks. Fig. 2 shows an illustration of TTS decoding by multiple GPUs.

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

3.2. Batch generation by multiple dataloaders

When training the target ASR model in the final stage, it is not advisable to mix the real-labeled dataset with the pseudo-labeled dataset and dynamically fetch data from the combined dataset. In most cases, unlabeled data far outweighs labeled

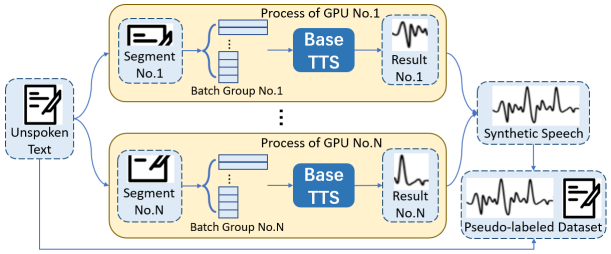


Figure 2: *Batch-level TTS inference by multiple GPUs. We group unlabeled text by the sum of their lengths to ensure a consistent number of synthetic frames in each batch. ASR beam searching shares a similar mechanism.*

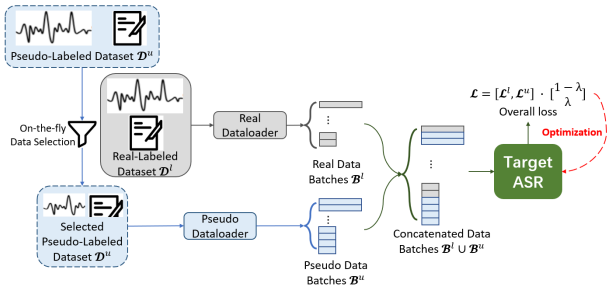


Figure 3: *Data loading by two dataloaders with on-the-fly data selection. Real and pseudo-batches with similar utterance lengths are concatenated to reduce the number of model forward passes.*

data, leading to many data batches dominated by pseudo-labeled data. Therefore, it is hard to control the gradient direction within each data batch, leading to unstable training.

Our open-source SpeeChain framework 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 computed for each batch but also introduces an evident real-pseudo composition within each batch. The clear composition enables us to design more advanced algorithms to better utilize pseudo-labeled data during training.

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

Our framework 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 the given metadata during training. We also provide histograms of metadata distributions to help users select appropriate thresholds to filter out low-quality pseudo-labeled data.

4. TTS→ASR EXPERIMENTS

4.1. ASR baseline and topline of supervised learning

LS-100 and LT-100 denote the respective *train_clean_100* portions of LibriSpeech and LibriTTS, whereas LS-360 and LT-360 denote their respective *train_clean_360* portions. The 100-hour and 360-hour portions are disjoint: no utterance from the latter is used to train either base model. Thus, the real data retained in the target-training stage are the same 100-hour source portions used for base-model training, while the 360-hour portion

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

Work (Model)	dev_clean	test_clean	test_other
Baseline on LS-100			
[7] (LAS [20])	-	12.46%	34.00%
[21] (TDS [22])	14.00%	14.85%	39.95%
[23] (Transformer [18])	12.20%	12.90%	31.10%
Ours (Transformer)	11.90%	12.20%	30.19%
Ours (Conformer)	**7.92%**	**7.95%**	**23.31%**
Topline on LS-100 & LS-360			
[7] (LAS)	-	6.30%	22.41%
[16] (LAS)	-	6.50%	-
Ours (Transformer)	**5.08%**	**5.65%**	**15.91%**
Topline on LS-100 & LT-360			
Ours (Transformer)	6.14%	6.57%	19.50%

supplies only the unspoken text (TTS→ASR) or untranscribed speech (ASR→TTS).

For all ASR results, we apply the LibriSpeech no-punctuation, lowercase text processing to both references and decoded hypotheses, split the resulting strings on whitespace, and compute word-level Levenshtein distance with the `editdistance` package. This same implementation is used for base-WER filtering and reported WER, and is included in the released SpeeChain recipes.

Our ASR models are based on Speech-Transformer [18], with encoder and decoder comprising 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 use 16-beam ASR decoding without a CTC branch across all experiments. We also evaluated Conformer [19] as another ASR architecture for the baseline with the same settings as the Transformer model. Although the Conformer has the lower standalone WER, we use the Transformer for all TTS→ASR chain comparisons because the Conformer experiment was conducted after all the other experiments; this also keeps the target architecture, decoding procedure, and filtering model fixed across chain comparisons and allows direct comparison with earlier Transformer-based synthetic-speech studies. The framework is architecture-agnostic; a controlled Conformer chain study is outside the scope of this work. For ASR→TTS, we instead use the stronger Conformer as the pseudo-transcription model to reduce label noise. The results of our baseline and topline models are shown in Table 1. The cited rows in the LS-100 block are supervised seed models reported in papers that subsequently study self-training; they are included as supervised baselines, not as self-training results. The best baseline model will be used as the base ASR model in the ASR→TTS chain.

4.2. Base TTS model training

Since LibriSpeech [24] was designed for ASR with a sampling rate of 16 kHz, we downsampled LibriTTS [25] from the original 24 kHz to 16 kHz for both base TTS training (LT-100) and speech synthesis (LT-360). Our TTS model is based on MultiSpeech [26], where the encoder and decoder are composed of the same 6 Transformer layers (d_{model}=512, n_{head}=8,

Table 2: *WER of target ASR models (Transformers)*

Filtering Thresholds	dev_clean	test_clean	test_other
TTS→ASR chain by LT-360			
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 LS-360			
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%**

d_{ff}=2048). The encoder directly consumes characters as the input.

We observed that TTS is data-hungrier than ASR. Therefore, we convert all text to uppercase and remove all punctuation except commas, periods, and single quotes to improve TTS convergence. We make our base TTS model generate four consecutive frames at each decoding step to speed up the speech synthesis. We do not use a neural vocoder; instead, we use the same 80-dimensional log-Mel spectral features extracted from 50-ms windows with a 10-ms window shift for both ASR and TTS training. Also, we found it helpful for base TTS training to scale the encoder and decoder embedding vectors by $\sqrt{d_{model}}$ before adding them to the positional encodings. For each LT-360 sentence, we sample a reference-speaker embedding from the LT-100 speaker pool; the speaker is drawn at random with a fixed seed, which supplies the multi-speaker conditioning for synthesis. Only the LT-360 text is used; its audio is not.

For reproducibility, the base TTS and all target Transformer ASR models use Adam ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$) with the Noam schedule: linear warm-up followed by inverse-square-root decay. The base TTS uses 8,000 warm-up steps, and the target ASR uses 16,000; the default Adam peak learning rate is 10^{-3}. We allow at most 1,000 epochs, use early stopping with patience 10, and average the ten most recent validation-selected checkpoints. These settings were selected during base-system development on *dev_clean* and then kept fixed for every filtering and batch-composition row; we did not tune them per chain condition.

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 to balance real and pseudo data. We compare our TTS→ASR chain with self-training of ASR [21], in which the base ASR model generates pseudo text for untranscribed speech to expand the training set on its own. In the ASR self-training experiments, we used LibriSpeech LS-360 as the unlabeled data and ASR decoding confidence as the filtering criterion.

The results show that base WER filtering does not improve, and at stricter thresholds reduces, the ASR improvement. Note that stricter thresholds also remove synthetic data, so the effect of noise and of data quantity cannot be separated here. One possible explanation is that base ASR WER does not directly correlate with downstream usefulness, and that slightly noisy synthetic speech adds beneficial variability similar to data augmentation; we did not test this hypothesis directly. Compared with the supervised LS-100 baseline in Table 1, the target ASR models also use the larger topline configuration, so part of the gain may come from model size rather than synthetic data alone.

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.*

Batch Composition	dev_clean	test_clean	test_other
Topline on LibriSpeech-*train_clean_100* & LibriTTS-*train_clean_360*			
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 2 characterizes synthetic-speech linguistic consistency through the base-ASR WER: it evaluates thresholds from 10% to no filtering under an otherwise fixed recipe. It is not a perceptual TTS-quality measure. An informative real-speech upper bound is the "Topline on LS-100 & LT-360" row in Table 1: replacing synthetic LT-360 speech with the corresponding real speech gives 6.57% and 19.50% WER on *test_clean* and *test_other*, respectively, versus 8.07% and 24.17% for the best synthetic-data result. This remaining gap is consistent with synthesis errors and acoustic mismatch limiting augmentation.

We also observed that the improvement of the TTS→ASR chain on *test_other* is smaller than ASR self-training. We hypothesize an acoustic mismatch between the *clean* and *other* datasets. Since our TTS model is trained only on *clean* data, the synthetic speech does not provide the target ASR model with as many acoustic details as real untranscribed speech.

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.

The 10% filtering condition in Table 3 is included to represent a strict-selection regime, not as an additional tuning choice. Applying this threshold leaves a smaller synthetic set whose natural whole-set real-to-synthetic ratio is 2:1; the no-filtering set has ratio 1:2. We retain those ratios when comparing static and dynamic batching within each regime. Table 2 varies the filtering threshold with $\lambda = 0.5$ for all rows; as the retained set size changes with the threshold, the filtering effect there is not fully separated from the real-synthetic composition.

Table 4: *Objective TTS evaluation on LibriTTS* test-clean *(4,837 utterances). Duration Ratio closer to 1.0 is better; all other metrics are lower-is-better. Values reported as mean ± std.*

System	MCD ↓	MSD ↓	Log-F0 ↓	Dur. Ratio
Baseline	7.46 ± 3.8	15.03 ± 3.9	0.32 ± 0.2	0.84 ± 0.1
ASR→TTS	7.47 ± 3.8	15.06 ± 3.9	0.32 ± 0.2	$\mathbf{0.95 \pm 0.2}$

5. ASR→TTS EXPERIMENTS

5.1. Experimental setup

We use LibriTTS [25] LT-100 (33K utterances) as the labeled dataset and LibriSpeech LS-360 (104K utterances) as the unlabeled speech source. LS-360 is disjoint from LT-100 and from the LS-100 portion used to train the base ASR model, and it is not used in base-model training. Our base ASR model is a Conformer [19] with 1K BPE tokens, trained on LS-100 (the best model in Table 1). Pseudo transcriptions are generated by applying this ASR model to LS-360 audio; the original LS-360 transcripts are withheld during chain training. We do not train a TTS model on the ground-truth LS-360 transcripts, so a real-label TTS upper bound is left to future work.

Our TTS model is FastSpeech2 [27] with an ECAPA-TDNN [28] speaker encoder (192-dim). Phoneme sequences are obtained from the Montreal Forced Aligner (MFA) [29] using the *english_us_arpa* acoustic model and lexicon. All audio is downsampled to 16 kHz. The FastSpeech2 model has 4 Transformer layers (d_{model}=384, n_{head}=2, d_{ff}=1536). Both TTS systems are trained for up to 500 epochs with Adam ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$) and a Noam schedule with 10,000 warm-up steps followed by inverse-square-root decay. Architecture and optimization settings are selected on the LT-100 baseline and then fixed for the chain system; no chain-specific hyperparameter search is performed. We compare two systems:

- **Baseline TTS**: FastSpeech2 trained on LibriTTS LT-100 only (33K real utterances, punctuation text format);
- **ASR→TTS Chain**: FastSpeech2 trained on LibriTTS LT-100 (33K real) plus LibriSpeech LS-360 with ASR pseudo labels (104K utterances), totaling 137K utterances.

For testing, we evaluated four metrics on all 4,837 synthesized utterances in the LibriTTS *test-clean* set. The average ECAPA speaker embeddings were extracted from the *test-clean* reference speech as speaker embeddings. The four metrics are **MCD** (Mel-Cepstral Distortion, lower is better), **MSD** (Mel-Spectral Distortion, lower is better), **Log-F0** RMSE (lower is better), and **Duration Ratio** (synthesized / reference duration).

5.2. Baseline TTS vs. ASR→TTS Chain

Table 4 shows the results of adding pseudo-labels from ASR for TTS training. The ASR→TTS chain achieves a duration ratio of 0.952, compared with 0.843 for the baseline, a 12.9% relative increase that moves the mean ratio closer to the natural value of 1.0, although its standard deviation also increases (0.2 vs. 0.1). This is the clearest observed benefit of pseudo-label augmentation, and it may be due to exposure to more diverse additional speech.

In contrast, the differences in MCD (+0.003), MSD (+0.027), and Log-F0 (+0.001) are negligible (<0.31%) in this objective evaluation. The marginal differences may reflect label noise arising from ASR transcription errors in the pseudo-data; the slightly higher training loss of the ASR→TTS system (1.45 vs. 1.29 for the baseline) is compatible with this, although the two losses are computed on different training sets. We hypothesize that the larger, more prosodically diverse training set helps the FastSpeech2 duration predictor generalize, but we did not run ablations to confirm this. Future work may investigate confidence-filtered pseudo-label selection to further reduce label noise while retaining the benefits of diversity.

6. CONCLUSION AND FUTURE WORK

This paper introduces SpeeChain, a framework for scalable machine speech chains. This framework provides an easy-to-use pipeline for TTS data augmentation for semi-supervised ASR, as well as pseudo-labeled ASR data to augment TTS training. It also supports individual training of ASR and TTS systems. Our extensive experiments demonstrate that semi-supervised ASR models carefully developed via the TTS→ASR chain can reduce the gap toward a supervised model trained on larger real-labeled data. Using ASR-generated transcriptions of unpaired audio as pseudo-labels enables TTS training on four times as much data while maintaining comparable objective acoustic metrics, demonstrating that the ASR→TTS chain is a viable data augmentation strategy for low-resource TTS. A limitation is that MCD, MSD, Log-F0 RMSE, and duration ratio do not establish perceived naturalness or intelligibility; we did not conduct a listening study or statistical significance tests. Future work should include human evaluation, statistical testing, and confidence-filtered ASR→TTS data, as well as integration of self-supervised learning models (SSL) into SpeeChain.

7. Acknowledgements

This work was partly supported by JSPS KAKENHI (grant numbers JP21H05054, JP21H03467, and JP24K02967), a Google AI grant, and the JSPS Program for Forming Japan's Peak Research Universities (J-PEAKS), grant number JPJS00420240021.

8. References

[1] A. Tjandra, S. Sakti, and S. Nakamura, "Machine speech chain," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 976–989, 2020.

[2] ——, "Listening while speaking: Speech chain by deep learning," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 301–308.

[3] ——, "Machine speech chain with one-shot speaker adaptation," *arXiv preprint arXiv:1803.10525*, 2018.

[4] N. Rossenbach, M. Zeineldeen, B. Hilmes, R. Schlüter, and H. 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.

[5] J. Li, R. Gadde, B. Ginsburg, and V. Lavrukhin, "Training neural speech recognition systems with synthetic speech augmentation," *arXiv preprint arXiv:1811.00707*, 2018.

[6] N. Rossenbach, A. Zeyer, R. Schlüter, and H. 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.

[7] A. Rosenberg, Y. Zhang, B. Ramabhadran, Y. Jia, P. Moreno, Y. Wu, and Z. Wu, "Speech recognition with augmented synthesized speech," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 996–1002.

[8] S. Ueno, M. Mimura, S. Sakai, and T. 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.

[9] Y. Ren, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Almost unsupervised text to speech and automatic speech recognition," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5410–5419.

[10] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "ESPnet: End-to-end speech processing toolkit," *arXiv preprint arXiv:1804.00015*, 2018.

[11] T. Hayashi, R. Yamamoto, K. Inoue, T. Yoshimura, S. Watanabe, T. Toda, K. Takeda, Y. Zhang, and X. 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.

[12] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," *arXiv preprint arXiv:1904.01038*, 2019.

[13] C. Wang, W.-N. Hsu, Y. Adi, A. Polyak, A. Lee, P.-J. Chen, J. Gu, and J. Pino, "fairseq sˆ2: A scalable and integrable speech synthesis toolkit," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2021, pp. 143–152.

[14] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong *et al.*, "Speechbrain: A general-purpose speech toolkit," *arXiv preprint arXiv:2106.04624*, 2021.

[15] H. Zhang, T. Yuan, J. Chen, X. Li, R. Zheng, Y. Huang, X. Chen, E. Gong, Z. Chen, X. Hu *et al.*, "PaddleSpeech: An easy-to-use all-in-one speech toolkit," *arXiv preprint arXiv:2205.12007*, 2022.

[16] F. Yue, Y. Deng, L. He, and T. Ko, "Exploring machine speech chain for domain adaptation and few-shot speaker adaptation," *arXiv preprint arXiv:2104.03815*, 2021.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. 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.

[18] L. Dong, S. Xu, and B. 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.

[19] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *INTERSPEECH*, 2020, pp. 5036–5040.

[20] W. Chan, N. Jaitly, Q. Le, and O. 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.

[21] J. Kahn, A. Lee, and A. 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.

[22] A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions," *arXiv preprint arXiv:1904.02619*, 2019.

[23] Y. Higuchi, N. Moritz, J. Le Roux, and T. Hori, "Momentum pseudo-labeling: Semi-supervised ASR with continuously improving pseudo-labels," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1424–1438, Sep. 2022.

[24] 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)*. IEEE, 2015, pp. 5206–5210.

[25] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "LibriTTS: A corpus derived from LibriSpeech for text-to-speech," *arXiv preprint arXiv:1904.02882*, 2019.

[26] M. Chen, X. Tan, Y. Ren, J. Xu, H. Sun, S. Zhao, T. Qin, and T.-Y. Liu, "Multispeech: Multi-speaker text to speech with transformer," *arXiv preprint arXiv:2006.04664*, 2020.

[27] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "FastSpeech 2: Fast and high-quality end-to-end text to speech," in *International Conference on Learning Representations*, 2021.

[28] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *INTERSPEECH*, 2020, pp. 3830–3834.

[29] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi," in *INTERSPEECH*, 2017, pp. 498–502.