# Machine Speech Chain

Andros Tjandra, Sakriani Sakti, *Member, IEEE*, and Satoshi Nakamura, *Fellow, IEEE*

***Abstract*—Despite the close relationship between speech perception and production, research in automatic speech recognition (ASR) and text-to-speech synthesis (TTS) has progressed more or less independently without exerting much mutual influence. In human communication, on the other hand, a closed-loop speech chain mechanism with auditory feedback from the speaker's mouth to her ear is crucial. In this paper, we take a step further and develop a closed-loop machine speech chain model based on deep learning. The sequence-to-sequence model in closed-loop architecture allows us to train our model on the concatenation of both labeled and unlabeled data. While ASR transcribes the unlabeled speech features, TTS attempts to reconstruct the original speech waveform based on the text from ASR. In the opposite direction, ASR also attempts to reconstruct the original text transcription given the synthesized speech. To the best of our knowledge, this is the first deep learning framework that integrates human speech perception and production behaviors. Our experimental results show that the proposed approach significantly improved performance over that from separate systems that were only trained with labeled data.**

***Index Terms*—Speech chain, ASR, TTS, deep learning.**

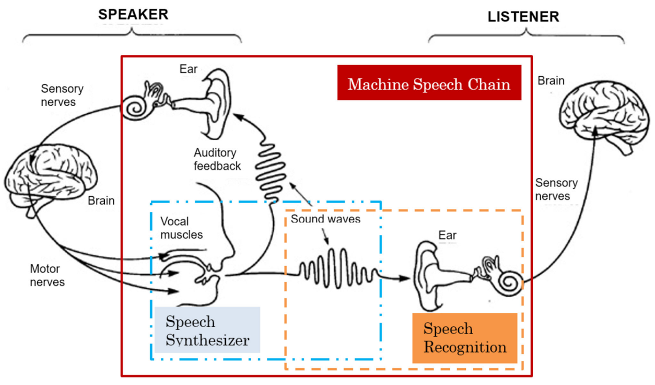


Fig. 1. Speech chain [1] and related spoken language technologies.

## I. Introduction

SPEECH chain, a concept introduced by Denes *et al.* [1], describes the basic mechanism involved in speech communication when a spoken message travels from the speaker‘s mind to the listener's mind (Fig. 1). It consists of a speech production mechanism in which the speaker produces words and generates speech sound waves, transmits the speech waveform through a medium (i.e., air), and creates a speech perception process in a listener's auditory system to perceive what was said. Over the past few decades, researchers have struggled to understand the principles underlying natural speech communication. Many attempts have also been made to replicate human speech perception and production with machines to support natural modality in human-machine interactions.

To date, the development of advanced spoken language technologies based on automatic speech recognition (ASR) and text-to-speech (TTS) has enabled machines to process and respond to basic human speech. Various ASR approaches have relied on acoustic-phonetics knowledge [2] in earlier works to template-based schemes with dynamic time warping (DTW) [3], [4] and data-driven approaches with rigorous statistical modeling of a hidden Markov model-Gaussian mixture model (HMM-GMM) [5], [6]. In a similar direction, TTS technology has gradually shifted from the foundation of a rule-based system using waveform coding and an analysis-synthesis method [7]–[9] to waveform unit concatenation [10], [11] and a more flexible approach using the statistical modeling of a hidden semi-Markov model-GMM (HSMM-GMM) [12], [13]. Recently, after the resurgence of deep learning, interest has also surfaced in the possibility of utilizing a neural approach for ASR and TTS systems. Many state-of-the-art performances in ASR [14]–[16] and TTS [17]–[19] tasks have been successfully constructed based on neural network frameworks.

However, despite the close relationship between speech perception and production, ASR and TTS research has progressed more or less independently without exerting much influence on each other. In human communication, on the other hand, a closed-loop speech chain mechanism has a critical auditory feedback mechanism from the speaker's mouth to her ear (Fig. 1). In other words, the hearing process is critical, not only for the listener, but also for the speaker. By simultaneously listening and speaking, the speaker can monitor her volume, articulation, and the general comprehensibility of her speech. Processing the information further, the speaker's brain can plan what she will say next. Children who lose their hearing often have difficulty producing clear speech due to their inability to monitor their own speech [20].

Unfortunately, investigating the inherent links between these two processes is very challenging. Difficulties arise because methodologies and analyses are necessarily quite different when they are extracting the underlying messages from speech waveforms, as in speech perception, or generating an optimum dynamic speaking style from the intended message, as in speech

Manuscript received June 19, 2019; revised November 8, 2019 and January 22, 2020; accepted February 18, 2020. Date of publication March 2, 2020; date of current version March 19, 2020. This work was supported in part by JSPS KAKENHI under Grants JP17H06101 and JP17K00237, and in part by NII CRIS Contract Research 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Xiaodong Cui. *(Corresponding author: Andros Tjandra.)*

The authors are with the Center for Advanced Intelligence Project AIP, Nara Institute of Science and Technology, Japan and RIKEN, Ikoma 630-0192, Japan (e-mail: andros.tjandra.ai6@is.naist.jp; ssakti@is.naist.jp; s-nakamura@is.naist.jp).

Digital Object Identifier 10.1109/TASLP.2020.2977776

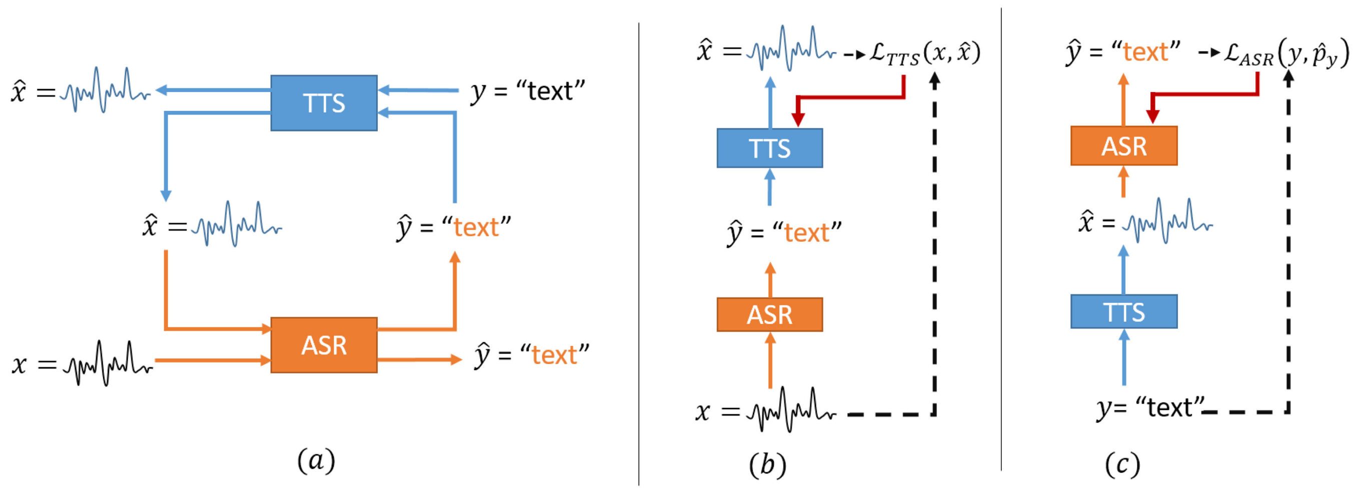


Fig. 2. (a) Overview of machine speech chain architecture. Examples of unrolled process: (b) from ASR to TTS and (c) from TTS to ASR.

production. Until recently, it was impossible in a joint approach to reunite the problems shared by both modes. However, due to deep learning's representational power, many complicated hand-engineered models have been simplified by letting deep neural nets (DNNs) learn their way from input to output spaces. With this newly emerging approach to sequence-to-sequence mapping tasks, a model with a common architecture can directly learn the mapping between variable-length representations of different modalities: text-to-text sequences [21], [22], speech-to-text sequences [23], [24], text-to-speech sequences [25], and image-to-text sequences [26], etc.

Therefore, in this paper, we take a step further and develop a closed-loop speech chain model based on deep learning and construct a sequence-to-sequence model for both ASR and TTS tasks, as well as a loop connection between these two processes. The sequence-to-sequence model in closed-loop architecture allows us to train our model on the concatenation of both labeled and unlabeled data. While ASR transcribes the unlabeled speech features, TTS attempts to reconstruct the original speech waveform based on text from ASR. In the opposite direction, ASR also reconstructs the original text transcription given the synthesized speech. To the best of our knowledge, this is the first deep learning model that integrates human speech perception and production behaviors.

Our contributions in this paper include:

1) Basic machine speech chain that integrates ASR and TTS and performs on single-speaker task.
2) Multi-speaker speech chain with a speaker-embedding network for handling speech with different voice characteristics.
3) Machine speech chain with a straight-through estimator to allow end-to-end feedback loss through discrete units or subwords.

## II. Related Works

Approaches that utilize learning from source-to-target and vice-versa, as well as feedback links, remain scant. He *et al.* [27] quite recently published a work that addressed a mechanism called dual learning in neural machine translation. Their system has a dual task: source-to-target language translation (primal) versus target-to-source language translation (dual). The primal and dual tasks form a closed loop and generate informative feedback signals to train the translation models, even without the involvement of a human labeler. This approach was originally proposed to tackle training data bottleneck problems. With a dual-learning mechanism, the system can leverage monolingual data (in both the source and target languages) more effectively. First, they construct one model to translate from the source to the target language and another to translate from the target to the source language. After both the first and second models have been trained with a small parallel corpus, they start to teach each other using monolingual data and generate useful feedback with language model likelihood and reconstruction error to further improve the performance.

Another similar work in neural machine translation was introduced by Cheng *et al.* [28], [29]. This approach also exploited monolingual corpora to improve neural machine translation. Their system utilizes a semi-supervised approach for training neural machine translation (NMT) models on the concatenation of labeled (parallel corpora) and unlabeled (monolingual corpora) data. The central idea is to reconstruct monolingual corpora using an autoencoder in which the source-to-target and target-to-source translation models serve as the encoder and decoder, respectively.

In this manuscript, we addressed similar problems in spoken language processing tasks. This paper presents a novel mechanism that integrates human speech perception and production behaviors. With a concept that resembles dual learning in NMT, we utilize the primal model (ASR) that transcribes the text given the speech versus the dual model (TTS) that synthesizes the speech given the text. However, the main difference between NMT is that the domain between the source and the target here are different (speech versus text). While ASR transcribes the unlabeled speech features, TTS attempts to reconstruct the original speech waveform based on the text from ASR. In the opposite direction, ASR also attempts to reconstruct the original text transcription given the synthesized speech. Nevertheless, our experimental results show that the proposed approach also identified a successful learning strategy and significantly improved performance over that of separate systems that were only trained with labeled data.

After our preliminary work, several works have discussed our methods and built on top of them. Karita *et al.* [30] form a text and speech autoencoder and train unpaired data with reconstruction loss. Ren *et al.*, [31] replaced the LSTM-based encoder-decoder with Transformer modules for both ASR and TTS and achieved good performance with small paired speech-text in single speaker dataset. Rosenberg *et al.* [32] explored the effect of data augmentation by using TTS on the larger experiment. Kurata *et al.* [33] improved ASR performance by adding feature reconstruction loss during training. Ueno *et al.* [34] use synthetic speeches from multi-speaker TTS to improve their Acoustic2Word speech recognition system. Baskar *et al.* [35] proposed an alternative to backpropagate through discrete variables by using a policy-gradient method, compared to our proposal using a straight-through estimator. Hori *et al.* [36] replaced TTS with text-to-encoder (TTE) to avoid the need for modeling the speaking style during the reconstruction.

## III. Basic Machine Speech Chain

### A. Overview

We start by explaining an overall basic machine speech chain mechanism. For a better understanding, we illustrated the speech chain loop in Fig. 2(a). Speech chain consists of a sequence-to-sequence ASR, a sequence-to-sequence TTS, and a loop connection from ASR to TTS and from TTS to ASR. The key idea is to jointly train both the ASR and TTS models. As mentioned above, the sequence-to-sequence model in closed-loop architecture allows us to train our model on the concatenation of both the labeled (paired) and unlabeled (unpaired) data.

To further clarify the learning process during supervised and unsupervised training, we unrolled the architecture as follows:

1) *Paired speech-text training for ASR and TTS:* Given the labeled data (speech-text paired data), both models can be trained independently by minimizing the loss between their predicted target sequence and the ground truth sequence via teacher forcing.
2) *Unpaired speech data only (ASR $\rightarrow$ TTS):* Given the unlabeled speech features, ASR transcribes the unlabeled input speech, while TTS reconstructs the original speech waveform based on the output text from ASR. Fig. 2(b) illustrates the mechanism. We may also treat it as an autoencoder model, where the speech-to-text ASR serves as an encoder and the text-to-speech TTS as a decoder.
3) *Unpaired text data only (TTS $\rightarrow$ ASR):* Given only the text input, TTS generates speech waveform, while ASR also reconstructs the original text transcription given the synthesized speech. Fig. 2(c) illustrates the mechanism. Here, we may also treat it as another autoencoder model, where the text-to-speech TTS serves as an encoder and the speech-to-text ASR as a decoder.

**Algorithm 1:** Speech Chain Algorithm.

```
1:  Input: Paired speech and text dataset $\mathcal{D}^P$, text-only dataset $\mathcal{Y}^U$, speech-only dataset $\mathcal{X}^U$, supervised loss coefficient $\alpha$, unsupervised loss coefficient $\beta$
2:  repeat
3:    A. Supervised training with speech-text data pairs
4:    Sample paired speech and text
      $(\mathbf{x}^P, \mathbf{y}^P) = ([x_1^P, .., x_{S_P}^P], [y_1^P, .., y_{T_P}^P])$
5:    from $\mathcal{D}^P$ with speech length $S_P$ and text length $T_P$.
      Generate a text probability vector by ASR using teacher forcing:
6:    $p_{y_t} = P(y_t|y_{<t}^P, \mathbf{x}^P; \theta_{ASR}), \forall t \in [1..T_P]$ Generate best predicted speech by TTS using teacher forcing:
7:    $\hat{x}_s^P = \text{argmax}_z P(z|\mathbf{x}_{<s}^P, \mathbf{y}^P; \theta_{TTS}); \forall s \in [1..S_P]$
8:    Calculate the loss for ASR and TTS    ▷ Eq. (13) & (14)
      $\ell_{ASR}^P = \mathcal{L}_{ASR}(\mathbf{y}^P, \mathbf{p}_y; \theta_{ASR})$    (1)
      $\ell_{TTS}^P = \mathcal{L}_{TTS}(\mathbf{x}^P, \hat{\mathbf{x}}^P; \theta_{TTS})$    (2)
9:    B. Unsupervised training with unpaired speech and text
10:   # Unpaired text data (TTS → ASR):
11:   Sample text $\mathbf{y}^U = [y_1^U, .., y_{T_U}^U]$ from $\mathcal{Y}^U$
12:   Generate speech by TTS: $\hat{\mathbf{x}}^U \sim P_{TTS}(\cdot|\mathbf{y}^U; \theta_{TTS})$
      Generate text probability vector by ASR from TTS's predicted speech using teacher forcing:
13:   $p_{y_t} = P(y_t|\mathbf{y}_{<t}^U, \hat{\mathbf{x}}^U; \theta_{ASR}); \quad \forall t \in [1..T_U]$
      Calculate the loss between original text $\mathbf{y}^U$ and reconstruction probability vector $\mathbf{p}_y$
      $\ell_{ASR}^U = \mathcal{L}_{ASR}(\mathbf{y}^U, \mathbf{p}_y; \theta_{ASR})$    (3)
14:   # Unpaired speech data (ASR → TTS):
15:   Sample speech $\mathbf{x}^U = [x_1^U, .., x_{S_U}^U]$ from $\mathcal{X}^U$
16:   Generate text by ASR: $\hat{\mathbf{y}}^U \sim P_{ASR}(\cdot|\mathbf{x}^U; \theta_{ASR})$
17:   Generate speech by TTS from ASR's predicted text using teacher forcing:
18:   $\hat{x}_s^U = \text{argmax}_z P_{TTS}(z|\mathbf{x}_{<s}^U, \hat{\mathbf{y}}^U; \theta_{TTS}); \quad \forall s \in [1..S]$ Calculate the loss between original speech $\mathbf{x}^U$ and generated speech $\hat{\mathbf{x}}^U$
      $\ell_{TTS}^U = \mathcal{L}_{TTS}(\mathbf{x}^U, \hat{\mathbf{x}}^U; \theta_{TTS})$    (4)
19:   # Loss combination:
20:   Combine all weighted loss into a single loss variable
      $\ell_{ALL} = \alpha * (\ell_{TTS}^P + \ell_{ASR}^P) + \beta * (\ell_{TTS}^U + \ell_{ASR}^U)$    (5)
21:   Calculate TTS and ASR parameters gradient with the derivative of $\ell_{ALL}$ w.r.t $\theta_{ASR}, \theta_{TTS}$
      $G_{ASR} = \nabla_{\theta_{ASR}} \ell$    (6)
      $G_{TTS} = \nabla_{\theta_{TTS}} \ell$    (7)
22:   Update TTS and ASR parameters with gradient descent optimization (SGD, Adam, etc.)
      $\theta_{ASR} \leftarrow Optim(\theta_{ASR}, G_{ASR})$    (8)
      $\theta_{TTS} \leftarrow Optim(\theta_{TTS}, G_{TTS})$    (9)
23: until convergence of parameter $\theta_{TTS}, \theta_{ASR}$
```

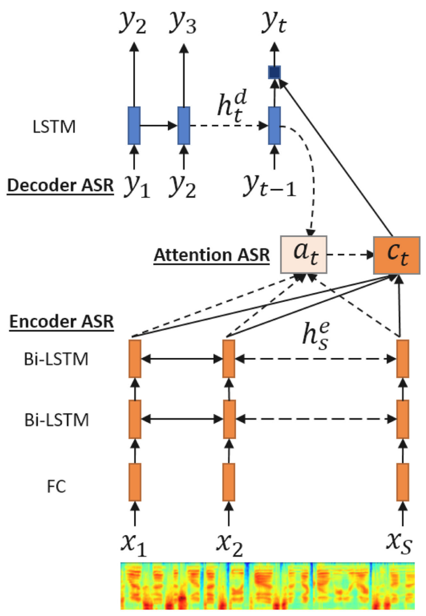


Fig. 3. Sequence-to-sequence ASR architecture: the encoder consists of a fully connected layer + stack bidirectional LSTM and the decoder consists of a unidirectional LSTM with attention mechanism.

With such autoencoder models, ASR and TTS can teach each other by adding a reconstruction term of the observed unlabeled data to the training objective. Details of the algorithm can be found in Algorithm 1.

## B. Sequence-to-Sequence Model for ASR

A sequence-to-sequence model is a neural network that directly models conditional probability $P(\mathbf{y}|\mathbf{x})$, where $\mathbf{x} = [x_1, \ldots, x_S]$ is the sequence of the (framed) speech features with length $S$ and $\mathbf{y} = [y_1, \ldots, y_T]$ is the sequence of labels with length $T$. Fig. 3 shows the overall structure of the attention-based encoder-decoder model that consists of encoder, decoder, and attention modules.

The encoder task processes input sequence $\mathbf{x}$, projects it through several layers (e.g., long short-term memory (LSTM) [37]/gated recurrent unit (GRU) [38], convolution, fully connected), and outputs representative information $\mathbf{h}^e = [h_1^e, \ldots, h_S^e]$ for the decoder. The attention module is an extension scheme that helps the decoder find relevant information on the encoder side based on the current decoder hidden states [21], [24]. Attention modules produce context information $c_t$ at time $t$ based on the encoder hidden states $h_e^s$ and decoder hidden states $h_t^d$:

$$c_t = \sum_{s=1}^{S} a_t(s) * h_s^e \tag{10}$$

$$\begin{aligned} a_t(s) &= \text{Align}(h_s^e, h_t^d) \\ &= \frac{\exp(\text{Score}(h_s^e, h_t^d))}{\sum_{s=1}^{S} \exp(\text{Score}(h_s^e, h_t^d))}. \end{aligned} \tag{11}$$

There are several variations for score functions [39]:

$$\text{Score}(h_s^e, h_t^d) = \begin{cases} \langle h_s^e, h_t^d \rangle, & \text{dot product} \\ h_s^{e\intercal} W_s h_t^d, & \text{bilinear} \\ V_s^\intercal \tanh(W_s[h_s^e, h_t^d]), & \text{MLP} \end{cases} \tag{12}$$

where Score : $(\mathbb{R}^M \times \mathbb{R}^N) \rightarrow \mathbb{R}$, $M$ is the number of hidden units for the last layer of encoder and $N$ is the number of hidden units for the decoder. Finally, the decoder is an autoregressive model that predicts target sequence probability $p_{y_t} = P(y_t|c_t, h_t^d, \mathbf{y}_{<t}; \theta_{ASR}) = P(y_t|\mathbf{x}, \mathbf{y}_{<t}; \theta_{ASR})$ at time $t$ based on previous output $\mathbf{y}_{<t}$, current context information $c_t$, and hidden state $h_t^d$. The loss function for ASR can be formulated as:

$$\ell_{ASR} = \mathcal{L}_{ASR}(\mathbf{y}, \mathbf{p}_y) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{c=1}^{C} \mathbb{1}(y_t = c) * \log p_{y_t}[c], \tag{13}$$

where $C$ is the number of output classes. Input $x$ for speech recognition tasks is a sequence of feature vectors like the log Mel-scale spectrogram. Therefore, $\mathbf{x} \in \mathbb{R}^{S \times D}$, where D is the number of feature dimensions and S is the total frame length for an utterance. Output $\mathbf{y}$, which is a speech transcription sequence, can be either a phoneme or grapheme (character) sequence.

## C. Sequence-to-Sequence Model for TTS

Parametric speech synthesis resembles a sequence-to-sequence task where we generate speech given a sentence. Using a sequence-to-sequence model, we model the conditional probability between $P(\mathbf{x}|\mathbf{y})$, where $\mathbf{y} = [y_1, \ldots, y_T]$ is the sequence of characters with length $T$ and $\mathbf{x} = [x_1, \ldots, x_S]$ is the sequence of (framed) speech features with length $S$. From the sequence-to-sequence ASR model perspective, we now have an inverse model for reconstructing the original speech given the text.

In this work, our core architecture is based on Tacotron [25] with several structural modifications. Fig. 4 illustrates our modified Tacotron. On the encoder side, we project our input characters with an embedding layer. The character vectors are fed into several fully connected layers followed by a non-linear activation function. We pass the result into the CBHG block (1-D **C**onvolution **B**ank + **H**ighway + bidirectional **G**RU) with eight filter banks (filter size ranging from 1 to 8). The CBHG output is expected to produce representative information $h^e = [h_1^e, \ldots, h_T^e]$ for the decoder.

Our modified decoder has one input layer and three output layers (instead of two as in the original Tacotron). The first output layer generates a sequence of log Mel-scale spectrogram frames $\mathbf{x}^M = [x_1^M, \ldots, x_S^M]$. At the $s$-th step, the input layer is fed by a previous step-log Mel-scale spectrogram $x_{s-1}^M$, and then several fully connected layers and a non-linear activation function are processed. Next, we use a stacked LSTM with a multilayer perceptron (MLP) attention with alignment and context history [40] to extract the expected context $c_s$ information based on the current decoder input and encoder states $h^e$. We project the context with a fully connected layer to predict the Mel-scale spectrogram.

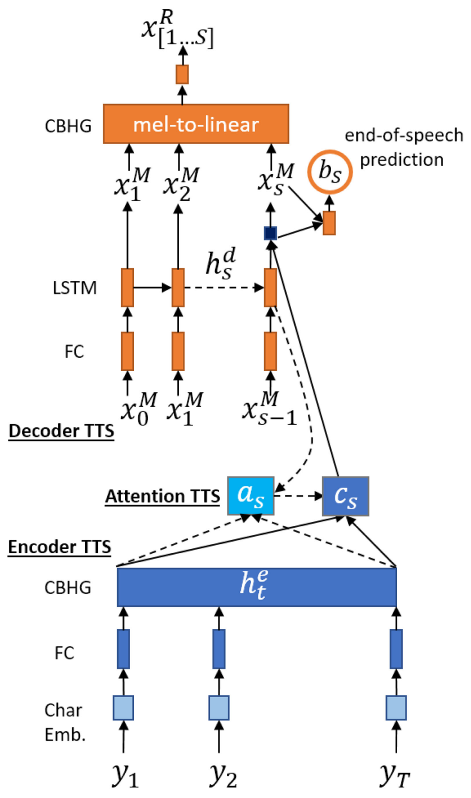


Fig. 4. Sequence-to-sequence TTS (Tacotron) architecture with frame ending binary prediction. (FC = Fully Connected, CBHG = Convolution Bank + Highway + bi-GRU)

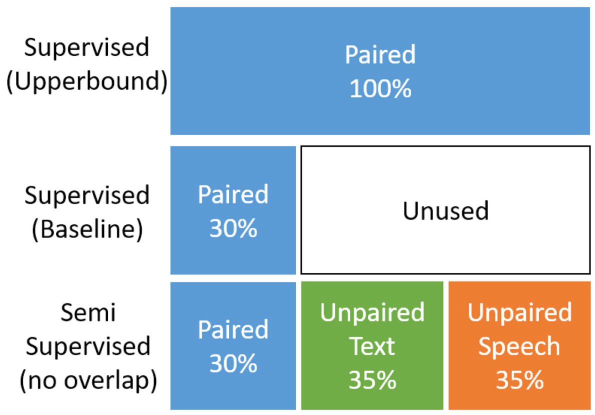


Fig. 5. Illustration for training data split between three different scenarios.

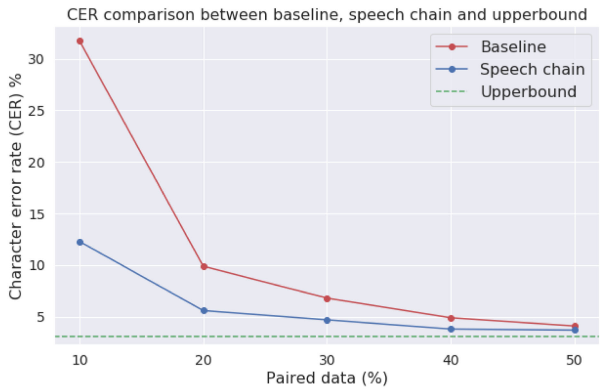


Fig. 6. Side-by-side CER (%) comparison between baseline, speech chain and upperbound with different percentage of paired speech-text data.

The second output layer reconstructs log-magnitude spectrogram $\mathbf{x}^R = [x^R_1, \ldots, x^R_S]$ given the first layer generated output $\mathbf{x}^M$. After we get complete the sequences of the log Mel-scale spectrogram, we feed them into a CBHG block followed by a fully connected layer to predict the log magnitude spectrogram.

The third output layer generates binary prediction $b_s \in [0, 1]$ (1 if the $s$-th frame is the end of speech, otherwise 0) based on the current log-Mel spectrogram generated by the first output layer and expected context $c_s$ from the decoder with the attention layer. We add the binary prediction layer because the output from the first and second decoder layers is a real value vector, and we cannot use an end-of-sentence (eos) token to determine when to stop the generation process. Based on our initial experiment, we found that our modification helped Tacotron determine the end of speech more robustly than forcing the decoder to generate frames with a 0 value at the end of the speech. We also enable our model to learn from multiple speakers by concatenating the projected speaker embedding into the input before the LSTM layer, first output regression layer, and second output regression layer.

For training the TTS model, we used the following loss function:

$$\ell_{TTS} = \mathcal{L}_{TTS}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{S}\sum_{s=1}^{S} \|x^M_s - \hat{x}^M_s\|^2_2 + \|x^R_s - \hat{x}^R_s\|^2_2 - (b_s \log(\hat{b}_s) + (1 - b_s)\log(1 - \hat{b}_s)), \quad (14)$$

where $\hat{\mathbf{x}} = (\hat{\mathbf{x}}^M, \hat{\mathbf{x}}^R, \hat{b})$ are the predicted Mel-scale spectrogram, the magnitude spectrogram, and the end-of-frame probability, and $\mathbf{x} = (\mathbf{x}^\mathbf{M}, \mathbf{x}^\mathbf{R}, b)$ is the ground truth. In the decoding process, we use the Griffin-Lim [41] algorithm to iteratively estimate the phase spectrogram and reconstruct the signal with the inverse short-time Fourier transform (STFT) from the predicted magnitude and phase spectrogram.

## D. Experiment on Single-Speaker Task

To verify our proposed method, we experimented on a corpus with a single speaker because, until recently, most TTS systems by deep learning are trained on a single speaker dataset.

We utilized a natural speech single-speaker dataset named LJSpeech [42] that contains about 13,100 utterances. Because there is no official dev and test split from this dataset, we shuffled it and randomly took 94% (total 12,314 utts) for training, 3% (total 393 utts) for dev, and 3% (total 393 utts) for the test set. Later, we split the train-set again to smaller ratio to compare the result between the paired only and paired + unpaired data. In Fig. 5, we illustrate how we split the training data for supervised (baseline) with small ratio of paired data, supervised (upperbound) with full paired data and semi-supervised with paired, unpaired text and unpaired speech. For the unpaired speech and text, there are 70% remaining unused data. To get the unpaired text, we take it by randomly sample 35% (without replacement) from the unused data. For the unpaired speech, we take it from the remaining 35% data. Therefore, there is no overlap between unpaired speech and unpaired text dataset.

*1) Feature Extraction:* For the speech features, we extracted two different sets of features: Mel spectrogram and magnitude spectrogram. Both the Mel spectrogram and magnitude spectrogram are extracted based on STFT with the librosa package [43]. All speech waveforms were sampled at 16 kHz. Given the raw speech waveform, we applied pre-emphasis (coefficient 0.97) and extracted the spectrogram with STFT (50-ms frame length, 12.5-ms frame shift, 2048-points FFT). After getting the spectrogram, we applied absolute and log operations to extract the log magnitude spectrogram features. To generate the Mel spectrogram features, we extracted the 80-dims Mel-scale coefficients from the magnitude spectrogram followed by log operation. Our final set is comprised of an 80-dimension log-Mel spectrogram and 1025-dimension log magnitude spectrogram. The log magnitude spectrogram features are used by TTS and the log-Mel spectrogram features are used by both TTS and ASR.

For the text, we converted all of the sentences into lowercase and replaced some punctuation marks (for example, ” into ’). In the end, we have 26 letters (a-z), six punctuation marks (,:’?.-), and three special tags (<s>, </s>, <spc>) to denote the start, end of sentence, and spaces between words.

*2) Model Details:* Our ASR model is an encoder-decoder with an attention mechanism. On the encoder side, we used a log-Mel spectrogram as the input features (in unsupervised process, the log-Mel spectrogram was generated by TTS), which are projected by a fully connected layer and a LeakyReLU ($l = 1e - 2$) [44] activation function, and processed by three stacked bidirectional LSTM (BiLSTM) layers with 256 hidden units for each direction (512 hidden units). We applied sequence subsampling [24], [45] to reduce the memory usage and computation time on the each LSTM layer and reduced the length of the speech features eight times shorter. On the decoder side, the input character is projected with a 128-dims embedding layer and fed into a one-layer LSTM with 512 hidden units. We calculated the attention matrix with an MLP scorer (Eq. 12) followed by a fully connected layer and a softmax function. In the decoding phase, the transcription was generated by beam-search decoding (size =5), and we normalized the log-likelihood score by dividing it by its own length to prevent the decoder from favoring the shorter transcriptions. We did not use any language model or lexicon dictionary in this work.

Our TTS model hyperparameters are generally the same as the original Tacotron, except that we used LeakyReLU instead of ReLU for most of the parts. On the encoder side, the CBHG used $K = 8$ different filter banks instead of 16 to reduce our GPU memory consumption. On the decoder side, we used a two-stacked LSTM instead of a GRU with 256 hidden units. Our TTS predicted four consecutive frames in one time-step to reduce the number of time-steps in the decoding process.

Both the ASR and TTS models are implemented with the PyTorch library.[1]

[1][Online]. Available: PyTorch https://github.com/pytorch/pytorch

TABLE I
ASR EXPERIMENT RESULT FOR LJSPEECH SINGLE-SPEAKER NATURAL SPEECH DATASET

| Model | Paired | Unpaired | | CER (%) |
|---|---|---|---|---|
| | | Text | Speech | |
| **Supervised (Baseline)** | | | | |
| Enc-Dec Att | 10% | - | - | 31.7 |
| Enc-Dec Att | 20% | - | - | 9.9 |
| Enc-Dec Att | 30% | - | - | 6.8 |
| Enc-Dec Att | 40% | - | - | 4.9 |
| Enc-Dec Att | 50% | - | - | 4.1 |
| **Semi-supervised (Speech Chain)** | | | | |
| Enc-Dec Att | 10% | 45% | 45% | 12.3 |
| Enc-Dec Att | 20% | 40% | 40% | 5.6 |
| Enc-Dec Att | 30% | 35% | 35% | 4.7 |
| Enc-Dec Att | 40% | 30% | 30% | 3.8 |
| Enc-Dec Att | 50% | 25% | 25% | 3.5 |
| **Supervised (Upperbound)** | | | | |
| Enc-Dec Att | 100% | - | - | 3.1 |

TABLE II
TTS EXPERIMENT RESULT FOR LJSPEECH SINGLE-SPEAKER NATURAL SPEECH DATASET

| Model | Paired | Unpaired | | L2-norm$^2$ |
|---|---|---|---|---|
| | | Text | Speech | |
| **Supervised (Baseline)** | | | | |
| Enc-Dec Att | 10% | - | - | 1.05 |
| Enc-Dec Att | 20% | - | - | 0.91 |
| Enc-Dec Att | 30% | - | - | 0.71 |
| Enc-Dec Att | 40% | - | - | 0.69 |
| Enc-Dec Att | 50% | - | - | 0.66 |
| **Semi-supervised (Speech Chain)** | | | | |
| Enc-Dec Att | 10% | 45% | 45% | 0.87 |
| Enc-Dec Att | 20% | 40% | 40% | 0.73 |
| Enc-Dec Att | 30% | 35% | 35% | 0.66 |
| Enc-Dec Att | 40% | 30% | 30% | 0.65 |
| Enc-Dec Att | 50% | 25% | 25% | 0.64 |
| **Supervised (Upperbound)** | | | | |
| Enc-Dec Att | 100% | - | - | 0.606 |

*3) Experiment Results:* For the ASR, we compare the character error rate (CER) between different scenarios in Table I. For the TTS experiment, we did both objective and subjective evaluations. In the objective evaluation, we compare the L2-norm squared between the predicted and ground truth log Mel-spectrogram in Table II. We experimented with a different ratio between the paired and unpaired data from the LJSpeech dataset. In the subjective evaluation, based on the quality of the synthesized speech using mean opinion score (MOS) test based on five-point scale (5: very good - 1: very poor). We compare three systems: 1) baseline with paired speech-text 30%, 2) speech chain with paired speech-text 30%, unpaired text 35% and unpaired speech 35 (no overlap)% and 3) upperbound paired speech-text 100%. To generate the samples, we randomly picked 20 utterances from the test set. In total, we have 27 subjects and each subject evaluates 60 utterances. We report the subjective evaluation result in Fig. 7.

The results show that after the ASR and TTS models are trained with a small paired dataset, they start to teach each other using unpaired data and generate useful feedback. Here, we improved both the ASR and TTS performance significantly compared to only using a portion of the paired dataset. We provided some samples from the single speaker

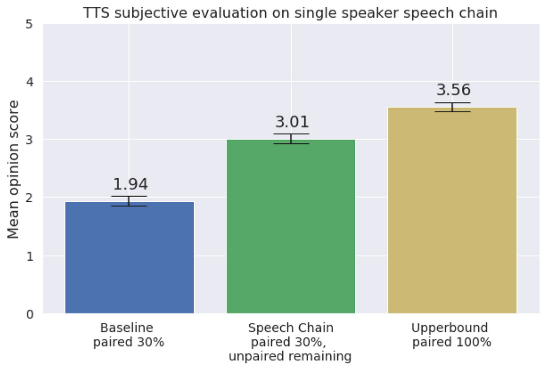


Fig. 7. MOS with 95% confidence interval between baseline, speech chain and upperbound scenario.

speech chain TTS experiments on https://speech-chain-single-spk-demo.netlify.com/https://speech-chain-single-spk-demo.netlify.com/.

### E. Discussion

In this section, we presented a basic speech chain mechanism and demonstrated the ability to train both ASR and TTS modules with paired and unpaired speech and a text dataset. However, there is a limitation in the unpaired training:

1) For training unpaired text, given an unpaired text, we can only generate speech with a specific speaking style. The speaking style is limited based on the speaker set that we used in the supervised TTS training.
2) In the unpaired speech training, the ASR transcribes a sentence. However, our TTS can only reconstruct the speech if the speaker identity from the unpaired speech is provided and the speaker embedding for that person has been seen during the supervised training.

## IV. Machine Speech Chain Framework With Speaker Adaptation

### A. Overview

Fig. 8 illustrates the updated speech chain mechanism. Similar to the earlier version, it consists of a sequence-to-sequence ASR [24], [46], a sequence-to-sequence TTS [25], and a loop connection from ASR to TTS and from TTS to ASR. The key idea is to jointly train the ASR and TTS models. The difference is that, in this version, we integrate a speaker recognition (SPKEMB) model inside the loop illustrated in Fig. 8(a). As mentioned above, we can train our model on the concatenation of both labeled (paired) and unlabeled (unpaired) data. We describe the learning process below.

1) *Paired speech-text dataset (see Fig. 8(a)):* Given the speech utterances $\mathbf{x}$ and the corresponding text transcription $\mathbf{y}$ from dataset $\mathcal{D}^P$, both the ASR and TTS models can be trained independently. Here, we can train ASR by calculating the ASR loss $\ell^P_{ASR}$ directly with teacher forcing. For TTS training, we generate a speaker-embedding vector $z = \text{SPKEMB}(\mathbf{x})$, integrate $z$ information with TTS, and calculate the TTS loss $\ell^P_{TTS}$ via teacher forcing.
2) *Unpaired speech data only (see Fig. 8(b)):* Given only the speech utterances $\mathbf{x}$ from unpaired dataset $\mathcal{D}^U$, ASR generates the text transcription $\hat{\mathbf{y}}$ (with greedy or beam-search decoding) and SPKEMB provides a speaker-embedding vector $z = \text{SPKEMB}(\mathbf{x})$. Given the generated text and the original speaker vector $z$, TTS then reconstructs the speech waveform $\hat{\mathbf{x}} = TTS(\hat{\mathbf{y}}, z)$ via teacher forcing. We then calculate the loss $\ell^U_{TTS}$ between $\mathbf{x}$ and $\hat{\mathbf{x}}$.
3) *Unpaired text data only (see Fig. 8(c)):* Given only the text transcription $\mathbf{y}$ from unpaired dataset $\mathcal{D}^U$, we need to sample speech from the available dataset $\tilde{\mathbf{x}} \sim (\mathcal{D}^P \cup \mathcal{D}^U)$ and generate a random speaker vector $\tilde{z} = \text{SPKEMB}(\tilde{\mathbf{x}})$ from SPKEMB. Then, TTS generates the speech utterance $\hat{\mathbf{x}}$ with greedy decoding. Given the generated speech $\hat{\mathbf{x}}$, ASR reconstructs the text $\hat{\mathbf{y}} = ASR(\hat{\mathbf{x}})$ via teacher forcing. We then calculate the loss $\ell^U_{ASR}$ between $\mathbf{y}$ and $\hat{\mathbf{y}}$.

We combine all losses together and update both the ASR and TTS model:

$$\ell = \alpha * (\ell^P_{ASR} + \ell^P_{TTS}) + \beta * (\ell^U_{ASR} + \ell^U_{TTS}) \quad (15)$$

$$\theta_{ASR} \leftarrow Optim(\theta_{ASR}, \nabla_{\theta_{ASR}} \ell) \quad (16)$$

$$\theta_{TTS} \leftarrow Optim(\theta_{TTS}, \nabla_{\theta_{TTS}} \ell), \quad (17)$$

where $\alpha$, $\beta$ are hyperparameters to scale the loss between the supervised (paired) and unsupervised (unpaired) loss, and $\nabla_{\theta_{ASR}} \ell$, $\nabla_{\theta_{TTS}} \ell$ are the gradient of combined loss $\ell$ w.r.t. ASR $\theta_{ASR}$ and TTS parameters $\theta_{TTS}$.

Fig. 8 illustrates the updated speech chain mechanism. Similar to the earlier version, it consists of a sequence-to-sequence ASR [24], [46], a sequence-to-sequence TTS [25], and a loop connection from ASR to TTS and from TTS to ASR. The key idea is to jointly train the ASR and TTS models. The difference is that, in this version, we integrate a speaker embedding (SPKEMB) model inside the loop illustrated in Fig. 8(a). As mentioned above, we can train our model on the concatenation of both labeled (paired) and unlabeled (unpaired) data. We describe the learning process below.

1) *Paired speech-text dataset (see Fig. 8(a)):* Given the speech utterances $\mathbf{x}$ and the corresponding text transcription $\mathbf{y}$ from dataset $\mathcal{D}^P$, both the ASR and TTS models can be trained independently. Here, we can train ASR by calculating the ASR loss $\ell^P_{ASR}$ directly with teacher forcing. For TTS training, we generate a speaker-embedding vector $z = \text{SPKEMB}(\mathbf{x})$, integrate $z$ information with TTS, and calculate the TTS loss $\ell^P_{TTS}$ via teacher forcing.
2) *Unpaired speech data only (see Fig. 8(b)):* Given only the speech utterances $\mathbf{x}$ from unpaired dataset $\mathcal{D}^U$, ASR generates the text transcription $\hat{\mathbf{y}}$ (with greedy or beam-search decoding) and SPKEMB provides a speaker-embedding vector $z = \text{SPKEMB}(\mathbf{x})$. Given the generated text and the original speaker vector $z$, TTS then reconstructs the speech waveform $\hat{\mathbf{x}} = TTS(\hat{\mathbf{y}}, z)$ via teacher forcing. We then calculate the loss $\ell^U_{TTS}$ between $\mathbf{x}$ and $\hat{\mathbf{x}}$.
3) *Unpaired text data only (see Fig. 8(c)):* Given only the text transcription $\mathbf{y}$ from unpaired dataset $\mathcal{D}^U$, we need to sample speech from the available dataset $\tilde{\mathbf{x}} \sim (\mathcal{D}^P \cup \mathcal{D}^U)$

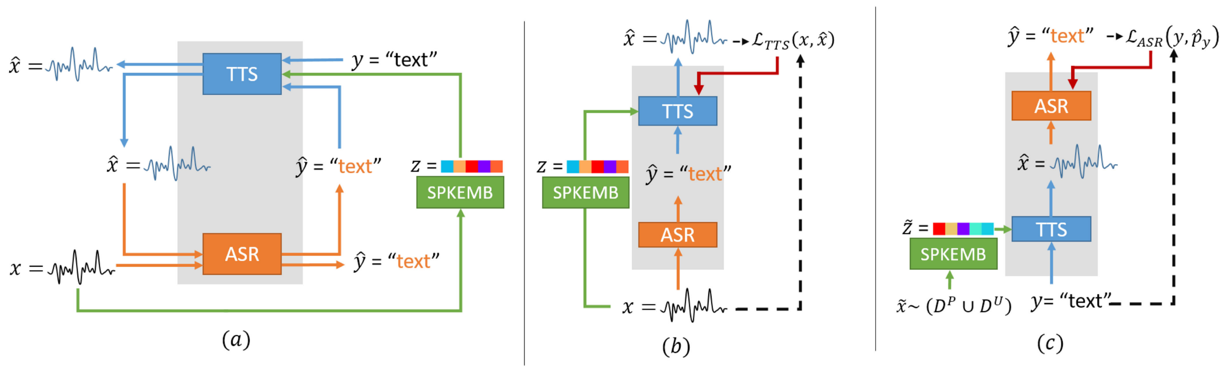


Fig. 8. (a) Overview of proposed machine speech chain architecture with speaker recognition; (b) Unrolled process with only speech utterances and no text transcription (speech → **[ASR,SPKEMB]** → [text + speaker vector] → **TTS** → speech); (c) Unrolled process with only text, but no corresponding speech utterance ([text + speaker vector by sampling **SPKEMB**] → **TTS** → speech → **ASR** → text). Note: grayed box is the original speech chain mechanism.

and generate a random speaker vector $\tilde{z} = \text{SPKEMB}(\tilde{\mathbf{x}})$ from SPKEMB. Then, TTS generates the speech utterance $\hat{\mathbf{x}}$ with greedy decoding. Given the generated speech $\hat{\mathbf{x}}$, ASR reconstructs the text $\hat{\mathbf{y}} = ASR(\hat{\mathbf{x}})$ via teacher forcing. We then calculate the loss $\ell^U_{ASR}$ between $\mathbf{y}$ and $\hat{\mathbf{y}}$.

We combine all losses together and update both the ASR and TTS model:

$$\ell = \alpha * (\ell^P_{ASR} + \ell^P_{TTS}) + \beta * (\ell^U_{ASR} + \ell^U_{TTS}) \quad (18)$$

$$\theta_{ASR} \leftarrow Optim(\theta_{ASR}, \nabla_{\theta_{ASR}}\ell) \quad (19)$$

$$\theta_{TTS} \leftarrow Optim(\theta_{TTS}, \nabla_{\theta_{TTS}}\ell), \quad (20)$$

where $\alpha$, $\beta$ are hyperparameters to scale the loss between the supervised (paired) and unsupervised (unpaired) loss, and $\nabla_{\theta_{ASR}}\ell$, $\nabla_{\theta_{TTS}}\ell$ are the gradient of combined loss $\ell$ w.r.t. ASR $\theta_{ASR}$ and TTS parameters $\theta_{TTS}$.

## B. *Speaker Recognition and Embedding*

Speaker recognition is a task to determine the identity of the speaker based on a spoken utterances. Another related tasks to speaker recognition is speaker identification, where the speaker identification model needs to predict if a pair of speech are come from same identity or not. By generating a embedding that correspond to the speaker identity, it can be used to predict both tasks. There are several traditional methods for speaker recognition such as i-vectors [47] and PLDA-based approach [48]. Since the deep learning approach become more popular, several deep learning architectures (DeepSpeaker [49], [50]) have been proposed to directly learn speaker representation from speech features. In Fig. 9 we illustrate DeepSpeaker architecture in more details.

To generate a speaker representation for speaker recognition task, we assume our input is a speech feature $\mathbf{x} \in \mathbb{R}^{S\times d_{in}}$. Then, we construct a deep neural network by stacking convolution, recurrent, pooling, etc and generate a fixed size vector $z \in \mathbb{R}^{d_z}$. On the top of $d_z$, we attach a linear projection and softmax activation function to calculate the probability along all possible $N$ speakers.

$$z = \text{SPKEMB}(\mathbf{x}) \quad (21)$$

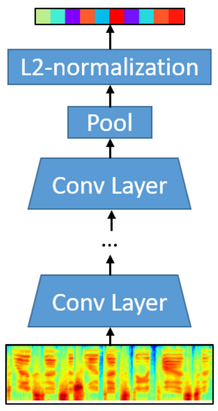


Fig. 9. Deep learning based speaker embedding (DeepSpeaker) architecture.

$$p_y = \text{Softmax}(zW_z) \quad (22)$$

To optimize the speaker representation model, there are several loss functions such as negative log-likehood:

$$\ell_{NLL} = -\sum_{n=1}^{N} \mathbb{1}(y=n) * \log p_y[n], \quad (23)$$

or distance-based such as triplet loss [51], [52]:

$$\ell_{TRI} = \sum_{\substack{a,p,n \\ y_a=y_p\neq y_n}} \max(\|z_a - z_p\|_2^2 + \|z_a - z_n\|_2^2, 0) \quad (24)$$

where $a, p, n$ are the anchor, positive and negative example and $z_a, z_p, z_n$ are their embedding respectively. In the training stage, those losses could be combined together and improved the final model performance [49].

## C. *Sequence-to-Sequence TTS With One-Shot Speaker Adaptation*

A parametric TTS can be formulated as a sequence-to-sequence model where the source sequence is a text utterance $\mathbf{y} = [y_1, .., y_T]$ with length $T$ and the target sequence is a speech feature $\mathbf{x} = [x_1, .., x_S]$ with length $S$. Our model objective is to maximize $P(\mathbf{x}|\mathbf{y};\theta_{TTS})$ w.r.t. TTS parameter $\theta_{TTS}$. We build our model upon the basic structure of the "Tacotron" TTS [25] and "Deep Speaker" models (Section IV-B).

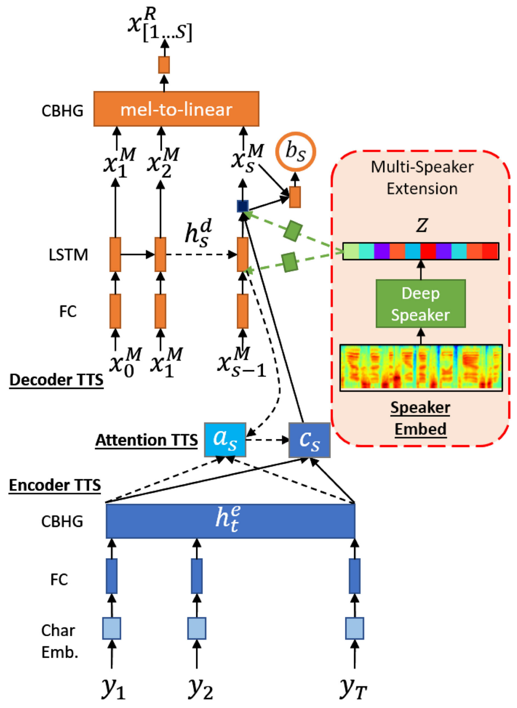


Fig. 10. Proposed model: sequence-to-sequence TTS (Tacotron) + speaker information via neural speaker embedding (Deep Speaker).

The original Tacotron is a single-speaker TTS system based on a sequence-to-sequence model. Given a text utterance, Tacotron produces the Mel spectrogram and linear spectrogram followed by the Griffin-Lim algorithm to recover the phase and reconstruct the speech signal. However, the original model is not designed to incorporate speaker identity or to generate speech from different speakers.

On the other hand, Deep Speaker is a deep neural speaker-embedding system (here denoted as "SPKEMB"). Given a sequence of speech features $\mathbf{x} = [x_1, .., x_S]$, Deep Speaker generates an L2-normalized continuous vector embedding $z$. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are spoken by the same speaker, the trained Deep Speaker model will produce the vector $z_1 = \text{SPKEMB}(\mathbf{x}_1)$ and the vector $z_2 = \text{SPKEMB}(\mathbf{x}_2)$, which are close to each other. Otherwise, the generated embeddings $z_1$ and $z_2$ will be far from each other. By combining Tacotron with Deep Speaker, we can do "one-shot" speaker adaptation by conditioning the Tacotron with the generated fixed-size continuous vector $z$ from Deep Speaker with a single speech utterance from any speaker.

Here, we adopt both systems by modifying the original Tacotron TTS model to integrate the Deep Speaker model. Fig. 10 illustrates our proposed model. From the encoder module side, the architecture and building blocks are same as the description from Section III-C.

On the decoder side, we have an autoregressive decoder. To produce different speech based on the original speaker target, we generate speaker-embedding vector $z = SPKEMB(\mathbf{x}^M)$. This speaker embedding $z$ is generated using only one utterance of the target speaker; thus it is called "one-shot" speaker adaptation. After that, we integrate speaker vector $z$ with a linear projection and sum it with the last output from the FC layer. Then, we apply two LSTM layers to generate current decoder state $h^d_s$. To retrieve the relevant information between the current decoder state and the entire encoder state, we calculate the attention probability $a_s(t) = Align(h^e_t, h^d_s); \forall t \in [1..T]$ and the expected context vector $c_s = \sum_1^T a_s(t) * h^e_t$. Then, we concatenate the decoder state $h^d_s$, context vector $c_s$, and projected speaker-embedding $z$ together into a vector, followed by two fully connected layers to produce the current time-step Mel spectrogram output $x^M_s$.

In the training stage, we optimized our proposed model by minimizing the following loss function:

$$\begin{aligned}\ell_{TTS} &= \mathcal{L}_{TTS}(\mathbf{x}, \hat{\mathbf{x}}, z, \hat{z}) \\ &= \left( \sum_{s=1}^{S} \gamma_1 \left( \|x^M_s - \hat{x}^M_s\|_2^2 + \|x^R_s - \hat{x}^R_s\|_2^2 \right) \right. \\ &\quad \left. - \gamma_2 \left( b_s \log(\hat{b}_s) + (1 - b_s) \log(1 - \hat{b}_s) \right) \right) \\ &\quad + \gamma_3 \left( 1 - \frac{<\hat{z}, z>}{\|\hat{z}\|_2 \, \|z\|_2} \right), \qquad (25)\end{aligned}$$

where $\gamma_1, \gamma_2, \gamma_3$ are our sub-loss hyperparameters, and $\mathbf{x}^M, \mathbf{x}^R, b, z$ are the ground-truth Mel spectrogram, linear spectrogram, and end-of-speech label and speaker-embedding vector from the real speech data, respectively. $\hat{\mathbf{x}}^M, \hat{\mathbf{x}}^R, \hat{b}$ represent the predicted Mel spectrogram, linear spectrogram, and end-of-speech label, respectively, and speaker-embedding vector $\hat{z} = \text{SPKEMB}(\hat{\mathbf{x}}^M)$ is the predicted speaker vector from the Tacotron output. Here, $\ell_{TTS}$ consists of three different loss formulations: Eq. (25) line 1 applies L2-norm squared error between the ground truth and predicted speech as a regression task, Eq. (25) line 2 applies binary cross entropy for end-of-speech prediction as a classification task, and Eq. (25) line 3 applies cosine distance between the ground-truth speaker-embedding $z$ and predicted speaker-embedding $\hat{z}$, which is the common metric for measuring the similarity between two vectors; furthermore, by minimizing this loss, we also minimize the global loss of speaker style [53], [54].

## D. Experiment on Multi-Speaker Task

*1) Corpus Dataset:* In this study, we ran our experiment on the Wall Street Journal (WSJ) CSR Corpus [55]. The complete data are contained in an SI284 (SI84+SI200) dataset. We followed the standard Kaldi [56] s5 recipe to split the training set, development set, and test set. To reformulate the speech chain as a semi-supervised learning method, we prepared SI84 and SI200 as paired and unpaired training sets, respectively. SI84 consists of 7138 utterances (about 16 hours of speech) spoken by 83 speakers, and SI200 consists of 30,180 utterances (about 66 hours) spoken by 200 speakers (without any overlap with speakers of SI84). We use "dev93" to denote the development and "eval92" for the test set.

*2) Feature and Text Representation on WSJ Dataset:* For the feature extraction, we use the same configuration as Section III-D1 The text utterances were tokenized as characters and mapped into a 33-character set: 26 alphabetic letters (a-z), 3 punctuation marks ('.-), and 4 special tags ⟨`noise`⟩,

TABLE III
CHARACTER ERROR RATE (CER (%)) COMPARISON BETWEEN RESULTS OF SUPERVISED LEARNING AND THOSE OF A SEMI-SUPERVISED LEARNING METHOD, EVALUATED ON TEST_EVAL92 SET (WITHOUT ANY LEXICON & LANGUAGE MODEL ON THE DECODING STEP)

| Model | CER (%) |
|---|---|
| **Supervised training: WSJ *train_si84* (paired) → Baseline** | |
| Att Enc-Dec [58] | 17.01 |
| Att Enc-Dec [59] | 17.68 |
| Att Enc-Dec (ours) | 17.35 |
| **Supervised training: WSJ *train_si284* (paired) → Upperbound** | |
| Att Enc-Dec [58] | 8.17 |
| Att Enc-Dec [59] | 7.69 |
| Att Enc-Dec (ours) | 7.12 |
| **Semi-supervised training: WSJ *train_si84* (paired) + *train_si200* (unpaired)** | |
| Label propagation (greedy) | 17.52 |
| Label propagation (beam=5) | 14.58 |
| **Proposed speech chain (Sec. IV)** | **9.86** |

⟨spc⟩, ⟨s⟩, and ⟨/s⟩ as noise, space, start-of-sequence, and end-of-sequence tokens, respectively. Both the ASR input and TTS output shared the same text representation.

*3) Model Details:* For the ASR and TTS encoder-decoder, we use a same setting as Section III-D2. We set the sub-loss hyperparameter in Eq. 25 with $\gamma_1 = 1, \gamma_2 = 1, \gamma_3 = 0.25$.

For the speaker recognition model, we used the Deep Speaker model and followed the original hyperparameters in the previous paper. However, our Deep Speaker is only trained on the WSJ SI84 set with 83 unique speakers. Thus, the model is expected to generalize effectively across all remaining unseen speakers to assist the TTS and speech chain training. We used Adam optimization with a learning rate of $5e-4$ for the ASR and TTS models and $1e-3$ for the Deep Speaker model. All of our models in this paper are implemented with PyTorch [57].

*4) Experiment Results:* Table III shows the ASR results from multiple scenarios evaluated on eval92. In the first block, we trained our baseline model by using paired samples from the SI84 set only, and we achieved 17.35% CER. In the second block, we trained our model with paired data of the full WSJ SI284 data, and we achieved 7.12% CER as our upperbound performance. In the last block, we trained our model with a semi-supervised learning approach using SI84 as paired data and SI200 as unpaired data. For comparison with other models trained with semi-supervised learning, we carried out label-propagation [60]. Label propagation is a simple way to do semi-supervised learning. First, we train initial model with paired speech-text $\mathcal{D}^P$. The pre-trained model is used to generate the hypothesis from the unpaired speech $\mathcal{X}^U$. Later, we add the unpaired speech and their correspondent hypothesis into training set and treat them as a paired dataset. Our result showed that by using label-propagation with beam-size $= 5$, we successfully reduced the CER to 14.58%. Nevertheless, our proposed speech-chain model could achieve a significant improvement over all baselines (paired only and label-propagation) with 9.89% CER, close to the upperbound results.

For the TTS experiment, we did both objective and subjective evaluations. In the objective evaluation, we calculated the difference with L2-norm squared between ground truth and the predicted log-Mel spectrogram and presented the result on Table IV. We observed similar trends with the ASR results, where the semi-supervised training with speech chain method improved significantly over the baseline and close to the upperbound result. In the subjective evaluation, based on the quality of the synthesized speech using mean opinion score (MOS) test based on five-point scale (5: very good - 1: very poor). To generate the samples, we randomly picked 20 utterances from the test set. In total, we have 26 subjects and each subject evaluates 60 utterances. We report the subjective evaluation result in Fig. 11. We provided some samples from multi-speaker speech chain TTS experiments on https://speech-chain-multi-spk-demo.netlify.com/https://speech-chain-multi-spk-demo.netlify.com/.

TABLE IV
L2-NORM SQUARED ON LOG-MEL SPECTROGRAM TO COMPARE THE SUPERVISED LEARNING AND THOSE OF A SEMI-SUPERVISED LEARNING METHOD, EVALUATED ON TEST_EVAL92 SET. NOTE: WE DID NOT INCLUDE STANDARD TACOTRON (WITHOUT SPKEMB) INTO THE TABLE SINCE IT CANNOT OUTPUT VARIOUS TARGET SPEAKERS

| Model | L2-norm$^2$ |
|---|---|
| **Supervised training: WSJ *train_si84* (paired) → Baseline** | |
| Proposed Tacotron (Sec. IV-C) (ours) | 1.036 |
| **Supervised training: WSJ *train_si284* (paired) → Upperbound** | |
| Proposed Tacotron (Sec. IV-C) (ours) | 0.836 |
| **Semi-supervised training: WSJ *train_si84* (paired) + *train_si200* (unpaired)** | |
| **Proposed speech chain (Sec. IV + Sec. IV-C)** | **0.886** |

TABLE V
ASR EXPERIMENT RESULT ON WSJ DATASET TEST_EVAL92

| **Baseline ($\ell_{ASR}$)** | | | |
|---|---|---|---|
| **Model** | | | **CER (%)** |
| Att MLP [58] | | | 11.08 |
| Att MLP + Location [58] | | | 8.17 |
| Att MLP [59] | | | 7.12 |
| Att MLP-MA (ours) [40] | | | 6.43 |
| **Proposed ($\ell_{ASR} + \ell_{TTS}^{rec}$)** | | | |
| **Model** | **$p_{y_t}$ generation** | **ST** | **CER (%)** |
| Att MLP-MA | Teacher forcing | argmax | 5.75 |
| Att MLP-MA | | gumbel | **5.7** |
| Att MLP-MA | Greedy | argmax | 5.84 |
| Att MLP-MA | | gumbel | 5.88 |

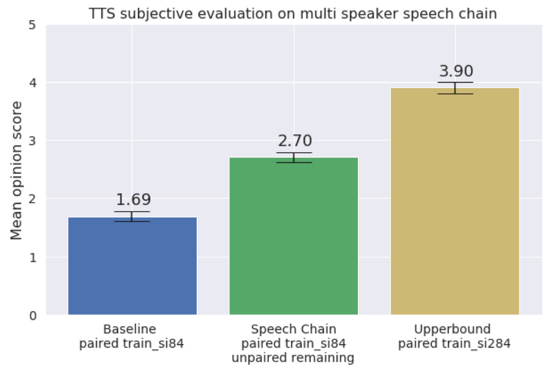


Fig. 11. MOS with 95% confidence interval between baseline, speech chain and upperbound scenario.

### E. Discussion

In this section, we introduced an improved speech chain mechanism by integrating a speaker recognition model inside the loop. By using the new system, we eliminated the downside from our basic speech chain, where we are unable to incorporate the data from unseen speakers. We also extended the capability of TTS to generate speech from an unseen speaker by implementing one-shot speaker adaptation. Thus, the TTS can generate speech with a similar voice characteristic only with a single utterance example. Inside the speech chain loop, the ASR also gets new data from the combination between a text sentence and an arbitrary voice characteristic. Our results show that after we deployed the speech-chain loop, the ASR system achieved significant improvement compared to the baseline (supervised training only) and other semi-supervised technique (label propagation). Like the trends in ASR, the TTS system also showed improvement compared to the baseline (supervised training only).

## V. End-to-End Feedback Loss on Speech Chain

### A. Overview

In the speech chain mechanism, given speech features $\mathbf{x} = [x_1, .., x_S]$ (e.g., Mel spectrogram) and text $\mathbf{y} = [y_1, .., y_T]$, we fed the speech to the ASR module and the ASR decoder generated continuous vector $h_t^d$ step by step. To calculate probability vector $\mathbf{p}_y = [p_{y_1}, .., p_{y_T}]$, we applied the softmax function $p_{y_t} = \mathtt{softmax}(h_t^d)$ to decoder output $h_t^d$. For each class probability mass in $p_{y_t}$, $p_{y_t}[c]$ was defined as:

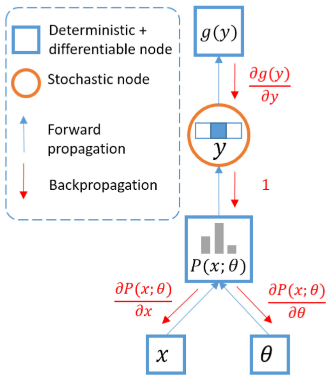
$$p_{y_t}[c] = \frac{\exp(h_t^d[c]/\tau)}{\sum_{i=1}^{C} \exp(h_t^d[i]/\tau)}, \quad \forall c \in [1..C]. \tag{26}$$

Here, $C$ is the total number of classes, $h_t^d \in \mathbb{R}^C$ are the logits produced by the last decoder layer, and $\tau$ is the temperature parameters. Setting temperature $\tau$ using a larger value $(\tau > 1)$ produces a smoother probability mass over classes [61].

For the generation process, we generally have two different methods:

1) Conditional generation given ground truth (teacher forcing):
If we have paired speech and text $(\mathbf{x}, \mathbf{y})$, we can generate $p_{y_t}$ from autoregressive ASR decoder $Dec_{ASR}(y_{t-1}, \mathbf{h}^e)$, conditioned to ground-truth text $y_{t-1}$ in the current time-step and encoded speech feature $\mathbf{h}^e = Enc_{ASR}(\mathbf{x})$. At the end, the length of probability vector $\mathbf{p}_y$ is fixed to $T$ time-steps.
2) Conditional generation given previous step model prediction:
Another generation process to decode ASR transcription uses its own prediction to generate probability vector $p_{y_t}$. There are many different generation methods, such as greedy decoding (1-best beam-search) ($\tilde{y}_t = \text{argmax}_c\, p_{y_t}[c]$), beam-search, or stochastic sampling ($\tilde{y}_t \sim Cat(p_{y_t})$).

After the generation process, we obtained probability vector $\mathbf{p}_y$ and applied discretization from continuous probability vector $p_{y_t}$ to $\tilde{y}_t$ either by taking the class with the highest probability or sampling from a categorical random variable. After getting a single class to represent the probability vector, we encoded it into vector [0,0 ,..,1,..,0] with one-hot encoding representation and gave it to the TTS as the encoder input. The TTS reconstructs Mel spectrogram $\hat{\mathbf{x}}$ with the teacher-forcing approach. The reconstruction loss is calculated by:



Deterministic + differentiable node
Stochastic node
Forward propagation
Backpropagation
$g(y)$
$\frac{\partial g(y)}{\partial y}$
$y$
1
$P(x;\theta)$
$\frac{\partial P(x;\theta)}{\partial x}$
$\frac{\partial P(x;\theta)}{\partial \theta}$
$x$
$\theta$

Fig. 12. **Straight-through estimator on** $\arg max$ **function**. Given input $x$ and model parameters $\theta$, we calculate categorical probability mass $P(x;\theta)$ and apply discrete operation $\text{argmax}$. In the backward pass, the gradient from stochastic node $y$ to $P(x;\theta)$, $\partial y / \partial P(x;\theta) \approx \mathbb{1}$ is approximated by identity.

$$\ell_{TTS}^{rec} = \mathcal{L}_{TTS}^{rec}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{S}\sum_{s=1}^{S} \|x_s^M - \hat{x}_s^M\|_2^2, \tag{27}$$

where $\hat{x}_s^M$ is the predicted (or reconstructed) Mel spectrogram and $x_s^M$ is the ground-truth spectrogram at $s$-th time-step.

We directly calculated the gradient from the reconstruction loss w.r.t. the TTS parameters $(\partial\ell_{TTS}^{rec}/\,\partial\theta_{TTS})$ because all the operations inside the TTS module are continuous and differentiable. However, we could not calculate the gradient from the reconstruction loss w.r.t. the ASR parameters $(\partial\ell_{TTS}^{rec}/\partial\theta_{ASR})$ because we have a discretization operation from $p_{y_t} \rightarrow \mathtt{onehot}(\tilde{y}_t)$. Therefore, we applied a straight-through estimator to enable the loss from $\ell_{TTS}^{rec}$ to pass through discrete variable $\tilde{y}_t$.

### B. End-to-End Feedback Loss

*1) Straight-Through Argmax:* The straight-through estimator [62], [63] is a method for estimating or propagating gradients through stochastic discrete variables. Its main idea is to backpropagate through discrete operations (e.g., $\text{argmax}_c\, p_{y_t}[c]$ or sampling $\tilde{y}_t \sim Cat(p_{y_t})$) like an identity function. We describe the forward process and the gradient calculation with a straight-through estimator in Fig. 12.

In the implementation, we created a function with different forward and backward operations. For the $\text{argmax}$ one-hot encoding function, we formulated the forward operation:

$$\tilde{z}_t = \underset{c}{\text{argmax}}\, p_{y_t}[c] \tag{28}$$

$$\tilde{y}_t = \mathtt{onehot}(\tilde{z}_t). \tag{29}$$

Here, we describe $\tilde{y}_t$ as a one-hot encoding vector with the same length as the $p_{y_t}$ vector. When the loss is calculated and the gradients are backpropagated from loss $\ell_{TTS}^{rec}$, we formulate the backward operation:

$$\frac{\partial \tilde{y}_t}{\partial p_{y_t}} \approx \mathbb{1}. \tag{30}$$

Therefore, when we backpropagate the loss from Eq. (27) with the straight-through estimator approach, we calculate the TTS reconstruction loss gradient w.r.t. $\theta_{ASR}$:

$$\frac{\partial \ell_{TTS}^{rec}}{\partial \theta_{ASR}} = \sum_{t=1}^{T} \frac{\partial \ell_{TTS}^{rec}}{\partial \tilde{y}_t} \cdot \frac{\partial \tilde{y}_t}{\partial p_{y_t}} \cdot \frac{\partial p_{y_t}}{\partial \theta_{ASR}} \tag{31}$$

$$\approx \sum_{t=1}^{T} \frac{\partial \ell_{TTS}^{rec}}{\partial \tilde{y}_t} \cdot \mathbb{1} \cdot \frac{\partial p_{y_t}}{\partial \theta_{ASR}}. \tag{32}$$

*2) Straight-Through Gumbel-Softmax:* Besides taking argmax class from probability vector $p_{y_t}$, we also generated a one-hot encoding by sampling with the Gumbel-Softmax distribution [64], [65]. Gumbel-Softmax is a continuous distribution that approximates categorical samples and the gradients can be calculated with a reparameterization trick. For Gumbel-Softmax, we replaced the softmax formula for calculating $p_{y_t}$ (Eq. (26)):

$$p_{y_t}[c] = \frac{\exp((h_t^d[c] + g_c)/\tau)}{\sum_{i=1}^{C} \exp((h_t^d[i] + g_i)/\tau)}, \quad \forall c \in [1..C], \tag{33}$$

where $g_1, .., g_C$ are i.i.d. samples drawn from Gumbel (0, 1) and $\tau$ is the temperature. We sample $g_c$ by drawing samples from the uniform distribution:

$$u_c \sim \texttt{Uniform}(0, 1) \tag{34}$$

$$g_c = -\log(-\log(u_c)), \quad \forall c \in [1..C]. \tag{35}$$

To generate a one-hot encoding, we define our forward operation:

$$\tilde{z}_t = \operatorname*{argmax}_c p_{y_t}[c] \tag{36}$$

$$\tilde{y}_t = \texttt{onehot}(\tilde{z}_t). \tag{37}$$

At the backpropagation time, we use the same straight-through estimator (Eq. (30)) to allow the gradients to flow through the discrete operation.

*3) Combined Loss for ASR:* Our final loss function for ASR is a combination from negative likelihood (Eq. (13)) and TTS reconstruction loss (Eq. (27)) by sum operation:

$$\ell_{ASR}^{F} = \ell_{ASR} + \ell_{TTS}^{rec}. \tag{38}$$

To summarize our explanation in this section, we provide an illustration in Fig. 13 that explains how sub-losses $\ell_{ASR}$ and $\ell_{TTS}^{rec}$ are backpropagated to the rest of the ASR and TTS modules.

## C. Experiment on Multi-Speaker Task in Supervised Settings

*1) Dataset:* We evaluated the performance of our proposed method on the WSJ dataset [55]. Our settings for the training, development, and test sets are the same as the Kaldi s5 recipe [56]. We trained our model with WSJ-SI284 data. Our validation set was `dev_93` and our test set was `eval_92`. For the feature extraction and text tokenization, we use the same setting as Section IV-D2.

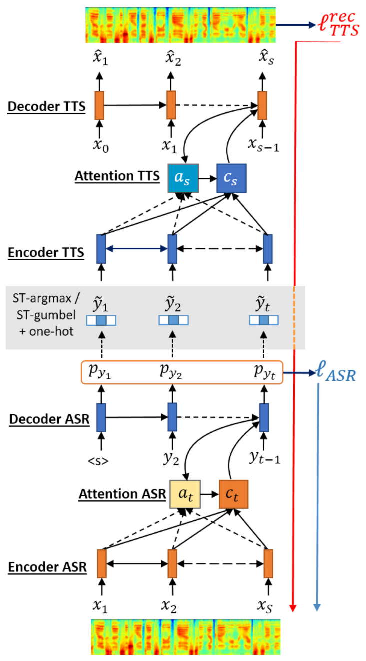


Fig. 13. Given speech feature $\mathbf{x}$, ASR generates a sequence of probability $\mathbf{p}_y = [p_{y_1}, p_{y_2}, \ldots, p_{y_T}]$. If we have a ground-truth transcription, we can calculate $\mathcal{L}_{ASR}$ (Eq. (13)). The TTS module generates speech features and we calculate reconstruction loss $\mathcal{L}_{TTS}^{rec}$ (Eq. (27)). After that, the gradients based on $\mathcal{L}_{ASR}$ are propagated through the ASR module, and the gradients based on $\mathcal{L}_{TTS}^{rec}$ are propagated through the TTS and ASR modules by a straight-through estimator.

*2) Model Details:* For the ASR model, we used a standard sequence-to-sequence model with an attention module (Section III-B). We use the same encoder setting as Section III-D2. On the decoder sides, we projected one-hot encoding from the previous character into a 256-dims continuous vector with an embedding matrix, followed by one unidirectional LSTM with 512 hidden units. For the attention module, we used the content-based attention + multiscale alignment (denoted as "Att MLP-MA") [40] with a 1-history size. In the evaluation stage, the transcription was generated by beam-search decoding (size =5), and we normalized the log-likelihood score by dividing it by its own length to prevent the decoder from favoring shorter transcriptions. We did not use any language model or lexicon dictionary in this work. In the training stage, we tried ST-argmax (Section V-B1) and ST-Gumbel softmax (Section V-B2). We also tried both teacher forcing and greedy decoding to generate ASR probability vectors $\mathbf{p}_y$ in the training stage. For each scenario, we treated temperature $\tau = [0.25, 0.5, 1, 2]$ as our hyperparameter and searched for the best temperature based on the CER on the development set.

For the TTS model, we used the modified Tacotron which is explained in Section III-C and we use same settings as Section III-D2.

*3) Experiment Results:* For our baseline, we trained an encoder-decoder with MLP + multiscale alignment with a 1-history size [40]. We also added several published results to our baseline. All of the baseline models were trained by minimizing negative log-likelihood $\ell_{ASR}$ (Eq. (13)).

All the models in the proposed section were trained with a combination from two losses, $\ell_{ASR} + \ell_{TTS}^{rec}$, and the ASR parameters were updated based on the gradient from the sum of the two losses. We have four different scenarios, most of which provide significant improvement compared to the baseline model that is only trained on $\mathcal{L}_{ASR}$ loss. With teacher forcing and sampling from Gumbel-softmax, we obtained 11% relative improvement compared to our best baseline Att MLP-MA.

## VI. Conclusion

This paper demonstrated a novel machine speech chain mechanism based on deep learning. The sequence-to-sequence model in closed-loop architecture allows us to train our model on the concatenation of both labeled and unlabeled data. We explored applications of the model in various tasks, including single speaker synthetic speech and multi-speaker natural speech. Our experimental results in both cases show that the proposed approach enabled ASR and TTS to improve performance by teaching each other using only unpaired data. We also introduced an extension to allow backpropagation through discrete output from the ASR module with a straight-through estimator. In the future, it is necessary to further validate the effectiveness of our approach on various languages and conditions (i.e., spontaneous, noisy, and emotion).

## References

[1] P. Denes and E. Pinson, *The Speech Chain*, (Anchor books.Series) New York, USA: Worth Publishers, 1993. [Online]. Available: https://books.google.co.jp/books?id=ZMTm3nlDfroC

[2] K. H. Davis, R. Biddulph, and S. Balashek, “Automatic recognition of spoken digits,” *Acoust. Soc. America*, vol. 24, pp. 627–642, 1952.

[3] T. K. Vintsyuk, “Speech discrimination by dynamic programming,” *Kibernetika*, vol. 4, pp. 81–88, 1968.

[4] H. Sakoe and S. Chiba, “Dynamic programming algorithm quantization for spoken word recognition,” *IEEE Trans. Acoust., Speech Signal Process.*, vol. ASSP-26, no. 1, pp. 43–49, Feb. 1978.

[5] F. Jelinek, “Continuous speech recognition by statistical methods,” *IEEE Proc.*, vol. 64, pp. 532–536, Apr. 1976.

[6] J. G. Wilpon, L. R. Rabiner, C. H. Lee, and E. R. Goldman, “Automatic recognition of keywords in unconstrained speech using hidden Markov models,” *IEEE Trans. Acoust., Speech Signal Process.*, vol. 38, no. 11, pp. 1870–1878, Nov. 1990.

[7] J. N. Holmes, I. G. Mattingly, and J. N. Shearme, “Speech synthesis by rule,” *Lang. Speech*, vol. 7, no. 3, pp. 127–143, 1964.

[8] J. P. Olive, “Rule synthesis of speech from dyadic units,” in *Proc. Int. Conf. Acoust., Speech Signal Process.*, 1977, pp. 568–570.

[9] Y. Sagisaka, “Speech synthesis by rule using an optimal selection of non-uniform synthesis units,” in *Proc. Int. Conf. Acoust., Speech Signal Process.*, 1988, pp. 679–682.

[10] Y. Sagisaka, N. Kaiki, N. Iwahashi, and K. Mimura, “Atr $\upsilon$-talk speech,” in *Proc. 4th Int. Conf. Spoken Lang. Process.*, 1992, pp. 483–486.

[11] A. Hunt and A. Black, “Unit selection in a concatenative speech synthesis system using a large speech database,” in *Proc. Int. Conf. Acoust., Speech Signal Process.*, 1996, pp. 373–376.

[12] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, “Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis,” in *Proc. Eurospeech*, 1999, pp. 2347–2350.

[13] K. Tokuda, T. Kobayashi, and S. Imai, “Speech parameter generation from HMM using dynamic features,” in *Proc. Int. Conf. Acoust., Speech Signal Process.*, 1995, pp. 660–663.

[14] A. Graves, A. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Proc. IEEE Acoust., Speech Signal Process. Int. Conf.*, 2013, pp. 6645–6649.

[15] D. Palaz, M. M. Doss, and R. Collobert, “Convolutional neural networks-based continuous speech recognition using raw speech signal,” in *Proc. IEEE Acoust., Speech Signal Process. Int. Conf.*, 2015, pp. 4295–4299.

[16] T. N. Sainath, R. J. Weiss, A. W. Senior, K. W. Wilson, and O. Vinyals, “Learning the speech front-end with raw waveform cldnns,” *Interspeech*, pp. 1–5, 2015.

[17] H. Zen, A. Senior, and M. Schuster, “Statistical parametric speech synthesis using deep neural networks,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 7962–7966.

[18] A. Oord *et al.*, “Wavenet: A generative model for raw audio,” 2016, *arXiv:1609.03499*.

[19] S. O. Arik *et al.*, “Deep voice: Real-time neural text-to-speech,” in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 195–204.

[20] N. R. Council, *Hearing Loss: Determining Eligibility for Social Security Benefits*, R. A. Dobie and S. V. Hemel, Eds. Washington, DC USA: The National Academies Press, 2004. [Online]. Available: https://www.nap.edu/catalog/11099/hearing-loss-determining-eligibility-for-social-security-benefits

[21] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, May 7–9, 2015. [Online]. Available: https://dblp.org/rec/journals/corr/BahdanauCB14.bib

[22] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence-to-Sequence learning with neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[23] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, “End-to-end continuous speech recognition using attention-based recurrent NN: First results,” in *Proc. Neural Inf. Process. Syst. 2014 Workshop Deep Learn.*, Dec. 2014.

[24] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in *Proc. IEEE Acoust., Speech Signal Process. Int. Conf.*, 2016, pp. 4960–4964.

[25] Y. Wang *et al.*, “Tacotron: Towards End-to-End Speech Synthesis,” in *Proc. Interspeech*, 2017, pp. 4006–4010.

[26] K. Xu *et al.*, “Show, attend and tell: Neural image caption generation with visual attention,” in *Proc. 32nd Int. Conf. Mach. Learn.,* Lille, France, Jul. 2015, pp. 2048–2057.

[27] D. He *et al.*, “Dual learning for machine translation,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 820–828.

[28] Y. Cheng *et al.*, “Semi-supervised learning for neural machine translation,” in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics Long Papers*, vol. 1. Berlin, Germany: Assoc. Comput. Linguistics, Aug. 2016, pp. 1965–1974. [Online]. Available: https://www.aclweb.org/anthology/P16-1185

[29] R. Sennrich, B. Haddow, and A. Birch, “Improving neural machine translation models with monolingual data,” in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics (Vol. 1: Long Papers)*, 2016, vol. 1, pp. 86–96.

[30] S. Karita, S. Watanabe, T. Iwata, A. Ogawa, and M. Delcroix, “Semi-supervised end-to-end speech recognition,” in *Proc. Interspeech*, 2018, pp. 2–6.

[31] Y. Ren, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, “Almost unsupervised text to speech and automatic speech recognition,” in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97. Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds., Long Beach, California, USA: PMLR, Jun. 9–15, 2019, pp. 5410–5419. [Online]. Available: http://proceedings.mlr.press/v97/ren19a/ren19a.pdf

[32] A. Rosenberg *et al.*, “Speech recognition with augmented synthesized speech,” *IEEE Autom. Speech Recognit. Understanding Workshop*, SG, Singapore, pp. 996–1002, Dec. 2019.

[33] G. Kurata and K. Audhkhasi, “Multi-task ctc training with auxiliary feature reconstruction for end-to-end speech recognition,” in *Proc. Interspeech*, 2019, pp. 1636–1640.

[34] 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 *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 6161–6165.

[35] M. K. Baskar, S. Watanabe, R. Astudillo, T. Hori, L. Burget, and J. Černockỳ, “Self-supervised sequence-to-sequence asr using unpaired speech and text,” in *Proc. Interspeech*, 2019, pp. 3790–3794, doi: 10.21437/Interspeech.2019-3167.

[36] T. Hori, R. Astudillo, T. Hayashi, Y. Zhang, S. Watanabe, and J. Le Roux, “Cycle-consistency training for end-to-end speech recognition,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 6271–6275.

[37] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” in *Proc. Neural Inf. Process. Syst. 2014 Workshop on Deep Learn.*, Dec. 2014.

[39] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proc. 2015 Conf. Empirical Methods in Natural Lang. Process.*, Lisbon, Portugal: Assoc. Comput. Linguistics, Sep. 2015, pp. 1412–1421. [Online]. Available: https://www.aclweb.org/anthology/D15-1166

[40] A. Tjandra, S. Sakti, and S. Nakamura, “Multi-scale alignment and contextual history for attention mechanism in sequence-to-sequence model,” in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2018, pp. 648–655.

[41] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 2, pp. 236–243, Apr. 1984.

[42] K. Ito, "The LJ speech dataset," 2017. [Online]. Available: https://keithito.com/LJ-Speech-Dataset/

[43] B. McFee *et al.*, "librosa 0.5.0," Feb. 2017.

[44] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*.

[45] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. 2014 Conf. Empirical Methods Natural Lang. Process.*, Doha, Qatar: Assoc. Comput. Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[46] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. IEEE Acoust., Speech Signal Process. Int. Conf.*, 2016, pp. 4945–4949.

[47] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 1695–1699.

[48] P. Matějka *et al.*, "Full-covariance UBM and heavy-tailed PLDA in i-vector speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2011, pp. 4828–4831.

[49] C. Li *et al.*, "Deep speaker: An end-to-end neural speaker embedding system," 2017, *arXiv:1705.02304*.

[50] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification," in *Proc. Interspeech*, 2018, pp. 3573–3577.

[51] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, pp. 207–244, 2009.

[52] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2015, pp. 815–823.

[53] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1558–1566.

[54] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vision.*, 2016, pp. 694–711.

[55] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus," in *Proc. Workshop Speech Natural Lang.* Association for Computational Linguistics, 1992, pp. 357–362.

[56] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*. Signal Processing Society, Dec. 2011, Catalog No.: CFP11SRW-USB.

[57] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[58] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. IEEE Acoust., Speech Signal Process. Int. Conf.*, 2017, pp. 4835–4839.

[59] A. Tjandra, S. Sakti, and S. Nakamura, "Attention-based wav2text with feature transfer learning," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop, ASRU* Okinawa, Japan, December 16-20, 2017, pp. 309–315. [Online]. Available: https://doi.org/10.1109/ASRU.2017.8268951

[60] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Sch. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CALD-02–107, 2002.

[61] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. Neural Inf. Process. Syst. Deep Learn. Representation Learn. Workshop*, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[62] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.

[63] G. Hinton, "Neural networks for machine learning, Coursera video lectures," 2012. [Online]. Available: https://www.coursera.org/course/neuralnets

[64] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://arxiv.org/abs/1611.01144

[65] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proc. Int. Conf. Learn. Representations*, 2016.

**Andros Tjandra** received the B.S. (*cum laude*) degree in computer science and the M.S. (*cum laude*) degree from the Faculty of Computer Science, Universitas Indonesia, Indonesia, in 2014 and 2015, respectively. He is currently working toward the Doctoral degree with the Graduate School of Information Science, Nara Institute of Technology, Ikoma, Japan. His research interests include machine learning, speech recognition, speech synthesis, and natural language processing.

**Sakriani Sakti** (Member, IEEE) received the B.E. (*cum laude*) degree in informatics from the Bandung Institute of Technology, Indonesia, in 1999, the M.Sc. degree from the University of Ulm, Ulm, Germany, in 2002, and the Ph.D. degree from the Dialog Systems Group University of Ulm, Ulm, Germany, in 2008. During her thesis work, she worked with the Daimler-Chrysler Research and Technology Speech Understanding Department in Ulm, Germany. Between 2003 and 2009, she worked as a Researcher with ATR SLC Labs, Japan, and during 2006 through 2011, she worked as an Expert Researcher with NICT SLC Groups, Japan. While working with ATR-NICT, Japan, she continued her study (2005–2008) with the Dialog Systems Group University of Ulm. She was actively involved in collaboration activities such as the Asia Pacific Telecommunity Project (2003–2007), A-STAR, and U-STAR (2006–2011). In 2009–2011, she served as a Visiting Professor with the Computer Science Department, University of Indonesia, Indonesia. From 2011, she has been an Assistant Professor with the Augmented Human Communication Laboratory, NAIST, Japan. She also served as a Visiting Scientific Researcher with INRIA Paris-Rocquencourt, France, from 2015 to 2016 under the JSPS Strategic Young Researcher Overseas Visits Program for Accelerating Brain Circulation. She is a member of JNS, SFN, ASJ, ISCA, and IEICE. Her research interests include statistical pattern recognition, speech recognition, spoken language translation, cognitive communication, and graphical modeling framework. In 2000, she was the recipient of DAAD-Siemens Program Asia 21st Century Award to study Communication Technology, University of Ulm, Ulm, Germany.

**Satoshi Nakamura** (Fellow, IEEE) received the B.S. degree from the Kyoto Institute of Technology, Kyoto, Japan, in 1981, and the Ph.D. degree from Kyoto University, Kyoto, Japan, in 1992. He is currently a Professor with the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma, Japan, Honorary Professor of the Karlsruhe Institute of Technology, Germany, and an ATR Fellow. He was an Associate Professor with the Graduate School of Information Science, Nara Institute of Science and Technology, from 1994 to 2000. He was a Director of the ATR Spoken Language Communication Research Laboratories from 2000 to 2008 and Vice President of ATR from 2007 to 2008. He was a Director General of Keihanna Research Laboratories and the Executive Director of the Knowledge Creating Communication Research Center, National Institute of Information and Communications Technology, Japan, from 2009 to 2010. He is currently a Director of the Augmented Human Communication Laboratory and a Full Professor of Nara Institute of Science and Technology's Graduate School of Information Science. He is interested in modeling and systems of speech-to-speech translation and speech recognition. He is one of the leaders in speech-to-speech translation research and has been participating in various speech-to-speech translation research projects throughout the world, including C-STAR, IWSLT, and A-STAR. He was the recipient of the Yamashita Research Award, Kiyasu Award from the Information Processing Society of Japan, Telecom System Award, AAMT Nagao Award, Docomo Mobile Science Award in 2007, and ASJ Award for Distinguished Achievements in Acoustics. He received the Commendation for Science and Technology from the Minister of Education, Science and Technology and the Commendation for Science and Technology from the Minister of Internal Affairs and Communications. He also received the LREC Antonio Zampolli Award 2012. He has been an Elected Board Member of the International Speech Communication Association, ISCA, since June 2011, *IEEE Signal Processing Magazine* Editorial Board Member since April 2012, IEEE SPS Speech and Language Technical Committee Member since 2013.