# Listening while Speaking:
# Speech Chain by Deep Learning

Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
Graduate School of Information Science
Nara Institute of Science and Technology, Japan
{andros.tjandra.ai6,ssakti,s-nakamura}@is.naist.jp

## 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 on each other. 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 speech chain model based on deep learning. The sequence-to-sequence model in close-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 model that integrates human speech perception and production behaviors. Our experimental results show that the proposed approach significantly improved the performance more than separate systems that were only trained with labeled data.

## 1  Introduction

The speech chain, which was first introduced by Denes et al. [1], describes the basic mechanism involved in speech communication when a spoken message travels from the speakers mind to the listeners 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 listeners auditory system to perceive what was said. Over the past few decades, tremendous research effort has struggled to understand the principles underlying natural speech communication. Many attempts have also been made to replicate human speech
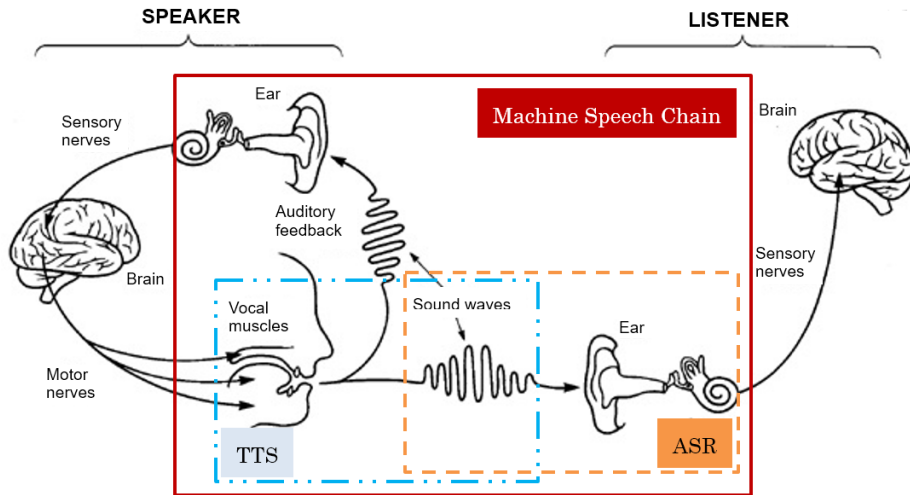
1

Figure 1: Speech chain [1] and related spoken language technologies.

perception and production by machines to support natural modality in human-machine interactions.

To date, the development of advanced spoken language technologies based on ASR and TTS has enabled machines to process and respond to basic human speech. Various ASR approaches have relied on acoustic-phonetics knowledge [2] in early 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 development has gradually shifted from the foundation of a rule-based system using waveform coding and an analysis-synthesis method [7, 8] to a waveform unit concatenation approach [9, 10] and a more flexible approach using the statistical modeling of hidden semi-Markov model-GMM (HSMM-GMM) [11, 12]. 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 [13, 14, 15] and TTS [16, 17, 18] tasks have been successfully constructed based on neural network frameworks.

However, despite the close relationship between speech perception and production, ASR and TTS researches have progressed more or less independently without exerting much mutual 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 speakers 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 speakers brain can plan what she will
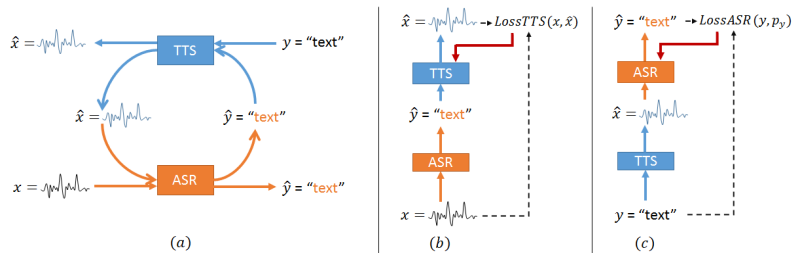
Figure 2: (a) Overview of machine speech chain architecture. Examples of unrolled process: (b) from ASR to TTS and (c) from TTS to ASR.

say next. Children who lose their hearing often have difficulty to produce clear speech due to inability to monitor their own speech.

Unfortunately, investigating the inherent links between these two processes is very challenging. Difficulties arise because methodologies and analysis 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 production. Until recently, it was impossible in a joint approach to reunite the problems shared by both modes. However, due to deep learnings representational power, many complicated hand-engineered models have been simplified by letting 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 [19, 20], speech-to-text sequences [21, 22], text-to-speech sequences [23], and image-to-text sequences [24], 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 reconstructs 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.

## 2   Machine Speech Chain

An overview of our proposed machine speech chain architecture is illustrated in Fig. 2(a). It 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,

---

**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
$(x^P, y^P) = ([x_1^P, .., x_{S_P}^P], [y_1^P, .., y_{T_P}^P])$
from $\mathcal{D}^P$ with speech length $S_P$ and text length $T_P$.

5:     Generate a text probability vector by ASR using teacher-forcing:
$p_{y_t} = P_{ASR}(\cdot|y_{<t}^P, x^P; \theta_{ASR}), \forall t \in [1..T_P]$

6:     Generate best predicted speech by TTS using teacher-forcing:
$\hat{x}_s^P = \underset{z}{\operatorname{argmax}}\, P_{TTS}(z|x_{<s}^P, y^P; \theta_{TTS}); \forall s \in [1..S_P]$

7:     Calculate the loss for ASR and TTS

$$L_P^{ASR} = LossASR(y^P, p_y; \theta_{ASR}) \tag{1}$$

$$L_P^{TTS} = LossTTS(x^P, \hat{x}^P; \theta_{TTS}) \tag{2}$$

8:     **B. Unsupervised training with unpaired speech and text**

9:     *# Unrolled process from TTS to ASR:*

10:     Sample text $y^U = [y_1^U, .., y_{T_U}^U]$ from $\mathcal{Y}^U$

11:     Generate speech by TTS: $\hat{x}^U \sim P_{TTS}(\cdot|y^U; \theta_{TTS})$

12:     Generate text probability vector by ASR from TTS's predicted speech using teacher-forcing:
$p_{y_t} = P_{ASR}(\cdot|y_{<t}^U, \hat{x}^U; \theta_{ASR}), \forall t \in [1..T_U]$

13:     Calculate the loss between original text $y^U$ and reconstruction probability vector $p_{y_t}$

$$L_U^{ASR} \quad = \quad LossASR(y^U, p_y; \theta_{ASR}) \tag{3}$$

14:     *# Unrolled process from ASR to TTS:*

15:     Sample speech $x^U = [x_1^U, .., x_{S_U}^U]$ from $\mathcal{X}^U$

16:     Generate text by ASR: $\hat{y}^U \sim P_{ASR}(\cdot|x^U; \theta_{ASR})$

17:     Generate speech by TTS from ASR's predicted text using teacher-forcing:
$\hat{x}_s^U = \underset{z}{\operatorname{argmax}}\, P_{TTS}(z|x_{<s}^U, \hat{y}^U; \theta_{TTS}); \forall s \in [1..S]$

18:     Calculate the loss between original speech $x^U$ and generated speech $\hat{x}^U$

$$L_U^{TTS} \quad = \quad LossTTS(x^U, \hat{x}^U; \theta_{TTS}) \tag{4}$$

19:     *# Loss combination:*

20:     Combine all weighted loss into a single loss variable

$$L = \alpha * (L_P^{TTS} + L_P^{ASR}) + \beta * (L_U^{TTS} + L_U^{ASR}) \tag{5}$$

21:     Calculate TTS and ASR parameters gradient with
the derivative of $L$ w.r.t $\theta_{ASR}, \theta_{TTS}$

$$G_{ASR} \quad = \quad \nabla_{\theta_{ASR}} L \tag{6}$$

$$G_{TTS} \quad = \quad \nabla_{\theta_{TTS}} L \tag{7}$$

22:     Update TTS and ASR parameters with gradient descent
optimization (SGD, Adam, etc)

$$\theta_{ASR} \leftarrow Optim(\theta_{ASR}, G_{ASR}) \tag{8}$$

$$\theta_{TTS} \leftarrow Optim(\theta_{TTS}, G_{TTS}) \tag{9}$$

23: **until** convergence of parameter $\theta_{TTS}, \theta_{ASR}$

---

the sequence-to-sequence model in closed-loop architecture allows us to train our model on the concatenation of both the labeled and unlabeled data. For supervised training with labeled data (speech-text pair data), both models can be trained independently by minimizing the loss between their predicted target sequence and the ground truth sequence. However, for unsupervised training with unlabeled data (speech only or text only), both models need to support each other through a connection.

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

- **Unrolled process from ASR to 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.

- **Unrolled process from TTS to 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.

With such autoencoder models, ASR and TTS are able to 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 Alg. 1.

## 3   Sequence-to-Sequence Model for ASR

A sequence-to-sequence model is a neural network that directly models conditional probability $p(y|x)$, where $x = [x_1, ..., x_S]$ is the sequence of the (framed) speech features with length $S$ and $y = [y_1, ..., y_T]$ is the sequence of label with length $T$. Fig. 3 shows the overall structure of the attention-based encoder-decoder model that consists of encoder, decoder and attention modules.
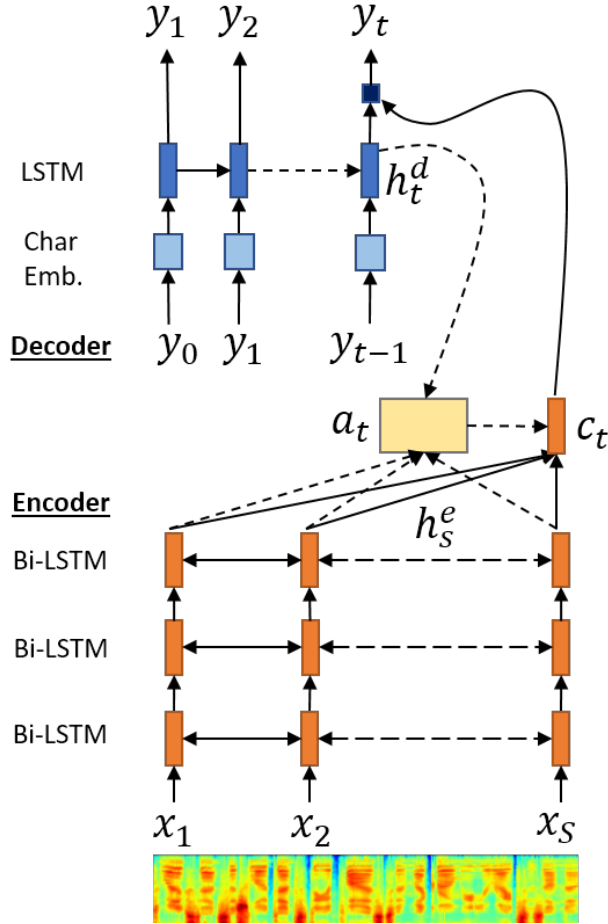
Figure 3: Sequence-to-sequence ASR architecture.

The encoder task processes input sequence $x$ and outputs representative information $h^e = [h_1^e, ..., h_S^e]$ for the decoder. The attention module is an extension scheme that assist the decoder to find relevant information on the encoder side based on the current decoder hidden states [19, 22]. Attention modules produces context information $c_t$ at time $t$ based on the encoder and decoder hidden states:

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

$$a_t(s) = \text{Align}(h_s^e, h_t^d)$$
$$= \frac{\exp(\text{Score}(h_s^e, h_t^d))}{\sum_{s=1}^{S} \exp(\text{Score}(h_s^e, h_t^d))} \qquad (11)$$

6

There are several variations for score functions [25]:

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

where $\text{Score} : (\mathbb{R}^M \times \mathbb{R}^N) \to \mathbb{R}$, $M$ is the number of hidden units for the encoder and $N$ is the number of hidden units for the decoder. Finally, the decoder task predicts target sequence probability $p_{y_t}$ at time $t$ based on previous output and context information $c_t$. The loss function for ASR can be formulated as:

$$Loss_{ASR}(y, p_y) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{c=1}^{C} \mathbb{1}(y_t = c) * \log p_{y_t}[c] \tag{13}$$

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

## 4    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 $p(x|y)$, where $y = [y_1, ..., y_T]$ is the sequence of characters with length $T$ and $x = [x_1, ..., x_S]$ is the sequence of (framed) speech features with length $S$. From the sequence-to-sequence ASR model perspective, now we have an inverse model for reconstructing the original speech given the text.
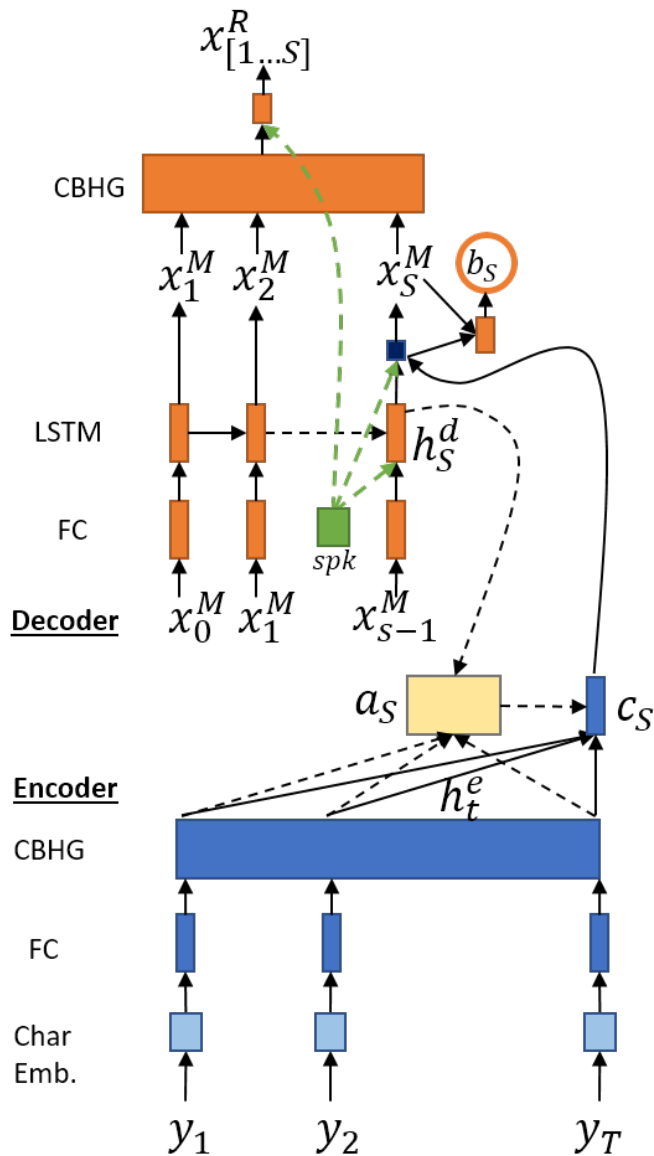
Figure 4: Sequence-to-sequence TTS (Tacotron) architecture with frame ending binary prediction. (FC = Fully Connected, CBHG = Convolution Bank + Highway + bi-GRU)

In this work, our core architecture is based on Tacotron [23] with several structural modifications. Fig. 4 illustrates our modified Tacotron. In the encoder sides, 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 from 1 to 8). The CBHG output is expected to produce representative information $h^e = [h_1^e, ..., h_T^e]$ 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 log Mel-scale spectrogram $x^M = [x_1^M, ..., x_S^M]$. 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 monotonic attention mechanism [26] to extract 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 log Mel-scale spectrogram.

The second output layer reconstructs log magnitude spectrogram $x^R = [x_1^R, ..., x_S^R]$ given the first layer generated output $x^M$. After we get complete 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 an 0 value at the end of the speech. We also enable our model to learn from multiple speaker by concatenating the projected speaker embedding into input before LSTM layer, first output regression layer, and second output regression layer.

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

$$
\begin{aligned}
Loss_{TTS}(x, \hat{x}, b, \hat{b}) = &\frac{1}{S} \sum_{s=1}^{S} (x_s^M - \hat{x}_s^M)^2 + (x_s^R - \hat{x}_s^R)^2 \\
&- (b_s \log(\hat{b}_s) + (1 - b_s) \log(1 - \hat{b}_s))
\end{aligned}
\tag{14}
$$

where $\hat{x}^M, \hat{x}^R, \hat{b}$ are the predicted log Mel-scale spectrogram, the log magnitude spectrogram and the end-of-frame probability, and $x^M, x^R, b$ is the ground truth. In the decoding process, we use Griffin-Lim [27] algorithm to iteratively estimate the phase spectrogram and reconstruct the signal with inverse STFT.

## 5 Experiment on Single-Speaker Task

To verify our proposed method, first 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.

To gather a large single speaker speech dataset, we utilized Google TTS [1] to generate a large set of speech waveform based on basic travel expression corpus (BTEC) [28] English sentences. For training and development we used part of the BTEC1 dataset, and for testing we used the default BTEC test set. For supervised training on both the ASR and TTS models, we chose 10,000 speech utterances that were paired with their corresponding text. For our development set, we selected another 3000 speech utterances and paired them with corresponding text. For our test set, we used all 510 utterances from the BTEC default test set. For the unsupervised learning step, we chose 40,000 speech utterances just from BTEC1 and 40,000 text utterances from BTEC1. None of these sets overlap to each other.

## 5.1 Features Extraction

For the speech features, we used a log magnitude spectrogram extracted by short-time Fourier transform (STFT) from the Librosa library [29]. First, we applied wave-normalization (scaling raw wave signals into a range of [-1, 1]) per utterance, followed by pre-emphasis (0.97), and extracted the spectrogram with STFT (50-ms frame length, 12.5-ms frame shift, 2048-point FFT). After getting the spectrogram, we used the squared magnitude and a Mel-scale filterbank with 40 filters to extract the Mel-scale spectrogram. After getting the Mel-spectrogram, we squared the magnitude spectrogram features. In the end, we transformed each speech utterance into a log-scale and normalized each feature into 0 mean and unit variances. Our final set is comprised of 40 dims log Mel-spectrogram features and a 1025 dims log magnitude spectrogram.

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 start, end of sentence, and spaces between words.

## 5.2 Model Details

Our ASR model is a standard 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 ($l = 1e-2$) [30] activation function and processed by three stacked BiLSTM layers with 256 hidden units for each direction (512 hidden units). We applied sequence subsampling [31, 22] on the top two layers and reduced the length of the speech features by a factor of 4. 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. Both the ASR and TTS models are implemented with the PyTorch library [2].

---

[1] Google TTS https://pypi.python.org/pypi/gTTS
[2] PyTorch https://github.com/pytorch/pytorch

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 sides, the CBHG used $K = 8$ different filter banks instead of 16 to reduce our GPU memory consumption. For the decoder sides, 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.

## 5.3   Experiment Result

Table 1 shows our result on the single-speaker ASR and TTS experiments. For the ASR experiment, we generated best hypothesis with beam search (size= 5). We used a character error rate (CER) for evaluating the ASR model. For the TTS experiment, we reported the MSE between the predicted log Mel and the log magnitude spectrogram to the ground truth. We also report the accuracy of our model that predicted the last speech frame. We used different values for $\alpha$ and text decoding strategy for ASR (in the unsupervised learning stage) with a greedy search or a beam search.

Table 1: Experiment result for single-speaker test set.

| Data | Hyperparameters | | | ASR | TTS | | |
|---|---|---|---|---|---|---|---|
| | $\alpha$ | $\beta$ | gen. mode | CER (%) | Mel | Raw | Acc (%) |
| Paired (10k) | - | - | - | 10.06 | 7.068 | 9.376 | 97.7 |
| + Unpaired (40k) | 0.25 | 1 | greedy | 5.83 | 6.212 | 8.485 | 98.4 |
| | 0.5 | 1 | greedy | 5.75 | 6.247 | 8.418 | 98.4 |
| | 0.25 | 1 | beam 5 | 5.44 | 6.243 | 8.441 | 98.3 |
| | 0.5 | 1 | beam 5 | 5.77 | 6.201 | 8.435 | 98.3 |

The result show that after ASR and TTS models have been trained with a small paired dataset, they start to teach each other using unpaired data and generate useful feedback. Here we improved both ASR and TTS performance. Our ASR model reduced CER by 4.6% compared to the system that was only trained with labeled data. In addition to ASR, our TTS also decreased the MSE and the end of speech prediction accuracy.

# 6   Experiment on Multi-Speaker Task

Based on our good result on a single speaker, we extended our initial work to a multi-speaker experiment. Unlike our previous experiment, we used a real natural speech corpus instead of synthesized speech. We used the BTEC ATR-EDB [32] corpus, which contains about 180,000 speech utterances from six different regions (Australia, British, US West, US Northeast, US South, and US West ). In this experiment, we only used the US Northeast portion in the dataset (contains about 22000 utterances). The US Northeast contains about 50

different speakers (25 males, 25 females) and about 440 utterances per speaker. For training, validation, and testing sets, we split the dataset by 20 utterances per speaker for validation, 20 utterances per speaker for testing, and the rest for training. For the paired speech and text, we got 80 pairs per speaker and the rest of the speech and text were used as unsupervised training sets. None of these training sets (paired and unpaired) or the validation and test sets overlap.

## 6.1 Model Details

In this experiment, we used the same ASR model as in the previous section without any modifications. However, for the TTS, we used the modified Tacotron with speaker embedding. Here, we used the pre-trained model from a single speaker and transferred the weight except for the speaker-embedding layer.

## 6.2 Experiment Result

Table 2: Experiment result for multi-speaker test set.

| Data | Hyperparameters | | | ASR | TTS | | |
|---|---|---|---|---|---|---|---|
| | $\alpha$ | $\beta$ | gen. mode | CER (%) | Mel | Raw | Acc (%) |
| Paired (80 utt/spk) | - | - | - | 26.47 | 10.213 | 13.175 | 98.6 |
| + Unpaired (remaining) | 0.25 | 1 | greedy | 23.03 | 9.137 | 12.863 | 98.7 |
| | 0.5 | 1 | greedy | 20.91 | 9.312 | 12.882 | 98.6 |
| | 0.25 | 1 | beam 5 | 22.55 | 9.359 | 12.767 | 98.6 |
| | 0.5 | 1 | beam 5 | 19.99 | 9.198 | 12.839 | 98.6 |

Table 2 reports our result on the multi-speaker ASR and TTS experiments. Similar with the single speaker result, we improved both the ASR and TTS models by additional training on unpaired datasets. However, in this experiment, we found that a different ASR performance $\alpha = 0.5$ produced a larger improvement than $\alpha = 0.25$. We hypothesize that because the baseline model is not as good as the previous single speaker experiment, we should put a larger coefficient on the loss and the gradient provided by the paired training set.

# 7  Related Works

Approaches that utilize learning from source-to-target and vice-versa as well as feedback links remain scant. He et al. [33] 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. [34]. 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 (mono-lingual 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.

However, no studies have yet 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 neural machine translation, 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 the performance more than separate systems that were only trained with labeled data.

# 8    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 further improved the performance by teaching each other using only unpaired data. 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).

# 9   Acknowledgements

# References

[1] P. Denes and E. Pinson, *The Speech Chain*, ser. Anchor books. 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," *Acoustic Society of America*, pp. 627–642, 1952.

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

[4] H. Sakoe and S. Chiba, "Dynamic programming algorithm quantization for spoken word recognition," *IEEE Transaction on Acoustics, Speech and Signal Processing*, vol. ASSP-26, no. 1, pp. 43–49, 1978.

[5] F. Jelinek, "Continuous speech recognition by statistical methods," *IEEE*, vol. 64, pp. 532–536, 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 Transaction on Acoustics, Speech and Signal Processing*, vol. 38, no. 11, pp. 1870–1878, 1990.

[7] J. P. Olive, "Rule synthesis of speech from dyadic units," in *Proceedings of ICASSP*, 1977, pp. 568–570.

[8] Y. Sagisaka, "Speech synthesis by rule using an optimal selection of non-uniform synthesis units," in *Proceedings of ICASSP*, 1988.

[9] Y. Sagisaka, N. Kaiki, N. Iwahashi, and K. Mimura, "Atr $v$-talk speech," in *Proceedings of ICSLP*, 1992, pp. 483–486.

[10] A. Hunt and A. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Proceedings of ICASSP*, 1996, pp. 373–376.

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

[12] K. Tokuda, T. Kobayashi, and S. Imai, "Speech parameter generation from HMM using dynamic features," in *Proceedings of ICASSP*, 1995, pp. 660–663.

[13] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on.* IEEE, 2013, pp. 6645–6649.

[14] D. Palaz, M. M. Doss, and R. Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on.* IEEE, 2015, pp. 4295–4299.

[15] T. N. Sainath, R. J. Weiss, A. W. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns." in *Interspeech*, vol. 2015, 2015.

[16] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 7962–7966.

[17] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[18] S. O. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman, S. Sengupta, and M. Shoeybi, "Deep voice: Real-time neural text-to-speech," *arXiv preprint arXiv:1702.07825*, 2017.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[20] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence-to-Sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[21] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent NN: First results," *arXiv preprint arXiv:1412.1602*, 2014.

[22] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on.* IEEE, 2016, pp. 4960–4964.

[23] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: A fully end-to-end text-to-speech synthesis model," *arXiv preprint arXiv:1703.10135*, 2017.

[24] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 2048–2057.

[25] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[26] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[27] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[28] G. Kikui, E. Sumita, T. Takezawa, and S. Yamamoto, "Creating corpora for speech-to-speech translation," in *Eighth European Conference on Speech Communication and Technology*, 2003.

[29] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, and et al., "librosa 0.5.0," Feb 2017.

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

[31] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[32] S. Sakti, M. Paul, A. Finch, X. Hu, J. Ni, N. Kimura, S. Matsuda, C. Hori, Y. Ashikari, H. Kawai, H. Kashioka, E. Sumita, and S. Nakamura, "Distributed speech translation technologies for multiparty multilingual communication," *ACM Trans. Speech Lang. Process.*, vol. 9, no. 2, pp. 4:1–4:27, Aug. 2012. [Online]. Available: http://doi.acm.org/10.1145/2287710.2287712

[33] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T. Liu, and W.-Y. Ma, "Dual learning for machine translation," in *Advances in Neural Information Processing Systems*, 2016, pp. 820–828.

[34] Y. Cheng, W. Xu, Z. He, W. He, H. Wu, M. Sun, and Y. Liu, "Semi-supervised learning for neural machine translation," *arXiv preprint arXiv:1606.04596*, 2016.