ABSTRACT

Text-to-speech synthesis (TTS) has been used as a data augmentation approach for automatic speech recognition (ASR), leveraging additional texts for ASR training. However, in low resource tasks, usually only a limited number of speakers are available, leading to the lack of speaker variations in synthetic speech. In this paper, we propose a novel speaker augmentation approach which can synthesize data with sufficient speaker and text diversity. Here, an end-to-end TTS system is trained with speaker representations from a variational auto-encoder (VAE), which enables TTS to synthesize speech from unseen new speakers via sampling from the trained latent distribution. As a new type of data augmentation approach, speaker augmentation can be combined with traditional feature augmentation approaches, such as SpecAugment. Experiments on a switchboard task show that, given 50 hours of data, the proposed speaker augmentation with SpecAugment significantly reduces word error rate (WER) by 30% relative compared to the system without any data augmentation, and about 18% relative compared to the system with SpecAugment.

Index Terms— Low resource, speech recognition, speech synthesis, variational autoencoder

1. INTRODUCTION

Training data for automatic speech recognition (ASR) in certain speaking style or certain language is sometimes limited. Though collecting additional data is a simple solution, it may be difficult in many cases. Data augmentation is an alternative that increases the quantity of training data.

The first type of data augmentation methods is producing variations on acoustic features. Noise augmentation[1] and speed perturbation[2] have been successfully applied to low resource ASR[3]. Recently, SpecAugment[4] is proposed as a powerful data augmentation approach using time warping and frequency masking.

The second type of data augmentation approach is leveraging additional texts with text-to-speech synthesis (TTS).


However, in low resource tasks, dataset often contains small number of speakers. Most prior works use only speakers that appear in training dataset for speech synthesis. Therefore, synthetic speech could provide only limited speaker diversity for data augmentation.

To address this problem, we propose a speaker augmentation approach. We train a Tacotron2[10], an end-to-end speech synthesis model, conditioned on speaker representations from a variational autoencoder (VAE)[11], similar to [12, 13]. Additionally, we jointly train a speaker classifier that takes latent variables as input. This encourages audio encoder to produce latent variables containing speaker information, assisting the model convergence. Applying the above techniques, our TTS model is able to synthesize speech from unseen new speakers via sampling from the trained latent distribution, providing sufficient speaker variations in synthetic speech for data augmentation. In our experiments, the baseline system uses only speakers that appear in the real data for speech synthesis. Given 5 hours real data, the proposed speaker augmentation approach reduces WER by 7% relative from the baseline. Our later experiments demonstrate that ASR still benefits from our approach when SpecAugment is combined, especially when more real data is available.

Then we investigate how our approach performs as the texts for speech synthesis increase. Given 50 hours data, we use additional text from Fisher corpus and try to find the convergence of WER reduction. The results show that our approach with SpecAugment significantly reduces word error rate (WER) by 30% relative compared to the system without any data augmentation, and about 18% relative compared to the system with SpecAugment.

In the rest of the paper, we first introduce our speaker aug-
mentation method in Section 2. Section 3 provides the setup of our experiments. The results and analysis are presented in Section 4.

2. SPEAKER AUGMENTATION

2.1. Speech synthesis with speaker variations

We propose a speaker augmentation approach for data augmentation in low resource ASR. Our dataset consists of text and mel sequence pairs \((X, Y) = \{(x_i, y_i)\}\). Each utterance \(i\) has speaker label \(s_i\). Our TTS model is based on Tacotron2[10].

In order to improve diversity of speakers in synthetic speech, we take latent variables \(z\) as speaker embeddings instead of a vector from speaker embedding table, similar to the architecture in [12, 13]. Prior distribution of \(z\) is isotropic standard Gaussian \(N(0, I)\). An audio encoder is designed to map a mel spectrogram to two vectors, representing the mean and log variance of the posterior probability distribution of latent variables, denoted as \(q(z|y)\). Tacotron2 is conditioned on \(z\) sampled from \(q(z|y)\) in training and from \(p(z)\) in generation.

We also find that TTS model tends to ignore latent variables and learn no speaker information in low resource tasks. In this paper, we add a linear layer to the audio encoder for speaker classification jointly trained with TTS. It accepts \(z\) as input, and outputs speaker class prediction \(C(s; z)\). Therefore, audio encoder is encouraged to produce \(z\) containing speaker information, assisting the model convergence.

Applying the above techniques, the loss function for TTS training can be formulated as

\[
L^{TTS} = E_q(z|y)[\log p(y; x, z)] + \lambda_1 \cdot D_{KL}[q(z|y)||N(0, I)] + \lambda_2 \cdot CE[C(s; z), s]
\]

(1)

where \(\lambda_1\) and \(\lambda_2\) are hyper-parameters to tune the relative weights between the three terms. The first term denotes reconstruction loss between generated mel-spectrogram and corresponding target. The second term is KL-divergence between \(q(z|y)\) and \(N(0, I)\). The last term represents the cross entropy loss between the speaker posterior and speaker label.

2.2. Data augmentation for low resource ASR

Figure 1 illustrates the complete architecture of our data augmentation strategy. Given a low resource dataset, a TTS model is first trained jointly with a variational autoencoder described above. Then we synthesize speech of additional text for data augmentation. The speaker of each utterance is described above. Then we synthesize speech of additional

\[
\text{Fig. 1. Architecture of data augmentation for low resource ASR}
\]

Therefoer, we mix the two parts in 1:1 proportion, maintaining good ASR performance on real data and leveraging synthetic data for generalization.

In addition, we attempt to combine our approach with SpecAugment for further improvements. Specifically, SpecAugment applies time warping and frequency masking to both real and synthetic data.

3. EXPERIMENTAL SETUP

Switchboard corpus[LDC97S62] consists of about 260 hours 8kHz 16bit telephone conversations. Our experiments are done on part of Switchboard, simulating low resource situations. We use eval2000 as our text set consisting of two partitions, namely swbd and callhm.

3.1. TTS training and inference

The overall TTS architecture is as described in Section 2.1 based on Tacotron2[10] with several modifications. We use phone sequence as input instead of character sequence. We also add speaker embedding projection layers to its encoder, decoder and postnet module, enabling multi-speaker training and generation. The input phone sequence is converted from character sequence with the Switchboard lexicon containing 42 types of non-silence phones. The output 80-dimension mel spectrogram is computed from 50ms windows shifted by 12.5ms. We apply forward attention mechanism[14] to speed up convergence and use Griffin-Lim[15] algorithm to reconstruct the waveform from predicted 80-dimension mel spectrogram.

The audio encoder in this work is designed to be simple for low resource tasks. It maps a mel spectrogram to two vectors, representing the mean and log variance of the posterior probability distribution of 512-dimension latent variables \(z\). The input mel spectrogram is passed through three convolutional layers, which contains 512 filters of shape \(3 \times 1, 9 \times 1\).
and $3 \times 1$ respectively. It is followed by an LSTM layer of 256 units and a mean pooling layer across time. The output of these layers is linearly projected to predict the posterior means and log variance of $z$. Speaker embedding, namely $z$ in this work, is sampled from $q(z|y)$ in training and from $p(z)$ in inference, conditioning the TTS generation.

We find that TTS model is hard to converge if we train the model directly on the low resource ASR dataset, so we first initialize the TTS model using LJSpeech corpus[16] for 20 epochs, which contains approximate 24 hours reading speech from a single female speaker. After the initialization, we train our model on the low resource ASR dataset to maximum Equation 1 with $\lambda_1 = 10^{-5}$ and $\lambda_2 = 0.1$. We use Adam optimizer[17] with an initial learning rate of $10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$. We validate our model every 2,000 steps and halve the learning rate by half whenever the reconstruction loss on the validation set does not decline.

In order to improve stability in inference, we use windowing technique[14] in attention calculation during autoregressive decoding. Specifically, we only consider a subsequence $[\alpha_{k-1}, \alpha_k, \alpha_{k+1}, \alpha_{k+2}]$ of the whole alignment sequence $\alpha$, where $\alpha_k$ is the maximum in $\alpha$. Another problem in autoregressive decoding is that the model fails to end due to long input sequence or bad synthesis. Therefore, we set a maximum decoding steps of 1,000 to avoid the situation. All utterances that reach the maximum decoding steps are discarded for all our experiments.

### 3.2. ASR training

We train our Transformer-based sequence-to-sequence(S2S) ASR model following [18]. It takes a sequence of 83-dim log-mel filterbank frames with pitch features as input, and outputs a sequence of byte pair encoding (BPE)[19]. The input is subsampled by two layers of 2D convolution with 256 filters, stride size 2 and kernel size 3, then concatenated with sinusoidal positional encoding and passed through 12 layers of Transformer blocks[20] with 4 self-attention heads and 256 hidden size. The encoder outputs are used for both CTC[21, 22] and the decoder. The decoder contains 6 layers of Transformer blocks and a linear projection layer predicting the posterior distribution of the next BPE. The loss function for ASR training is formulated as:

$$L^{ASR} = -\alpha \cdot \log p_{a2s}(x; y) - (1 - \alpha) \cdot \log p_{ctc}(x; y)$$

where $\alpha$ is a hyper-parameter to tune the relative weights between S2S and CTC loss. We set $\alpha = 0.8$ for all experiments. Both real and synthetic data are used for ASR training. We use Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and an initial learning rate of 5.0. The learning rate over the course of training is varied following [23].

### 4. RESULT AND DISCUSSION

#### 4.1. Leveraging virtual speakers from latent space

We first gives an upperbound that utilizes all 260 hours Switchboard data for ASR training. Then we assume that only 5 hours Switchboard data is available, containing 25 speakers. We train a TTS system on the 5 hours data and synthesize the rest of Switchboard, about 255 hours speech, using only transcriptions. In baseline system, speaker of each synthetic utterance is selected randomly from the 25 speakers that appear in 5 hours real data. Our proposed method enables us to sample unseen new speakers namely virtual speakers from latent space, improving the speaker diversity in synthetic speech. We sample various number of virtual speakers for data augmentation and train ASR on both real and synthetic data. Results are shown in Table 1. Generally, ASR performance is improved as more virtual speakers are sampled. We achieve the best result on both swbd and callhm test set when 300 virtual speakers are utilized, reducing relative WER of 6.5% and 7.7% from the baseline system respectively.

<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Aug</th>
<th>V.Sprk</th>
<th>swbd</th>
<th>callhm</th>
</tr>
</thead>
<tbody>
<tr>
<td>260 h</td>
<td>12.4</td>
<td>23.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 h</td>
<td>84.1</td>
<td>89.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 h S.P.</td>
<td>72.0</td>
<td>77.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baseline</td>
<td>35.6</td>
<td>48.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 h TTS</td>
<td>33.5</td>
<td>48.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 h TTS</td>
<td>33.7</td>
<td>44.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 h TTS</td>
<td>33.3</td>
<td>44.5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. WER(%) on swbd and callhm test set. ”Real” means the quantity of available real data, ”Aug” denotes data augmentation approach. ”S.P.” is the abbreviation for speed perturb. ”V.Sprk” represents the number of virtual speakers used for speech synthesis.

Figure 2 demonstrates a group of mel spectrograms that correspond to the same transcription which is not contained in low resource training set. As we expected, Figure 2(b) is similar to 2(a), while 2(c) differs from 2(a) and 2(b) in both duration and fundamental frequency.

#### 4.2. Combining with SpecAugment

SpecAugment[4] is a powerful data augmentation method for ASR. We further investigate whether ASR could benefit from our approach when SpecAugment is combined. $F$ and $T$ of SpecAugment set to 30 and 40 respectively, $m_F$ and $m_T$ are both set to 2, and $W$ is set to 5. As shown in Table 2, less relative WER reduction is obtained compared with Table 1. This is partially because SpecAugment also brings diversity
Fig. 2. A group of mel spectrograms that correspond to the same transcription

Table 2. WER(%) on swbd and callhm test set assuming 5 hours Switchboard is available. “S.A.” is the abbreviation for SpecAugment.

<table>
<thead>
<tr>
<th>Aug</th>
<th>V.Spkr</th>
<th>swbd</th>
<th>callhm</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS</td>
<td>-</td>
<td>25.6</td>
<td>39.2</td>
</tr>
<tr>
<td>TTS + S.A.</td>
<td>-</td>
<td>20.2</td>
<td>32.5</td>
</tr>
<tr>
<td>Baseline</td>
<td>TTS + S.A.</td>
<td>0</td>
<td>21.1</td>
</tr>
<tr>
<td>Proposed</td>
<td>TTS + S.A.</td>
<td>25</td>
<td>17.8</td>
</tr>
<tr>
<td>Proposed</td>
<td>TTS + S.A.</td>
<td>100</td>
<td>16.5</td>
</tr>
<tr>
<td>Proposed</td>
<td>TTS + S.A.</td>
<td>300</td>
<td><strong>16.5</strong></td>
</tr>
</tbody>
</table>

Table 3. WER(%) on swbd and callhm test set assuming 50 hours Switchboard is available. “S.A.” is the abbreviation for SpecAugment.

<table>
<thead>
<tr>
<th>Aug</th>
<th>V.Spkr</th>
<th>swbd</th>
<th>callhm</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>-</td>
<td>25.6</td>
<td>39.2</td>
</tr>
<tr>
<td>S.A.</td>
<td>-</td>
<td>20.2</td>
<td>32.5</td>
</tr>
<tr>
<td>TTS</td>
<td>0</td>
<td>21.1</td>
<td>35.2</td>
</tr>
<tr>
<td>Baseline</td>
<td>TTS + S.A.</td>
<td>0</td>
<td>17.8</td>
</tr>
<tr>
<td>Proposed</td>
<td>TTS + S.A.</td>
<td>25</td>
<td>17.2</td>
</tr>
<tr>
<td>Proposed</td>
<td>TTS + S.A.</td>
<td>100</td>
<td>16.5</td>
</tr>
<tr>
<td>Proposed</td>
<td>TTS + S.A.</td>
<td>300</td>
<td><strong>16.5</strong></td>
</tr>
</tbody>
</table>

Fig. 3. ASR performance as the texts for TTS increase
6. REFERENCES


