DATA AUGMENTATION FOR ASR USING TTS VIA A DISCRETE REPRESENTATION

Sei Ueno[1], *Masato Mimura*[1], *Shinsuke Sakai*[1], *Tatsuya Kawahara*[1]

[1]Graduate School of Informatics, Kyoto University,
Sakyo-ku, Kyoto, Japan

ABSTRACT

While end-to-end automatic speech recognition (ASR) has achieved high performance, it requires a huge amount of paired speech and transcription data for training. Recently, data augmentation methods have actively been investigated. One method is to use a text-to-speech (TTS) system to generate speech data from text-only data and use the generated speech for data augmentation, but it has been found that the synthesized log Mel-scale filterbank (lmfb) features could have a serious mismatch with the real speech features. In this study, we propose a data augmentation method via a discrete speech representation. The TTS model predicts discrete ID sequences instead of lmfb features, and the ASR also uses the ID sequences as training data. We expect that the use of a discrete representation based on vq-wav2vec not only makes TTS training easier but also mitigates the mismatch with real data. Experimental evaluations show that the proposed method outperforms the data augmentation method using the conventional TTS. We found that it reduces speaker dependency, and the generated features are distributed more closely to the real ones.

Index Terms: Speech recognition, Sequence-to-sequence model, Data augmentation, Vq-wav2vec, Speech synthesis

1. INTRODUCTION

The deep learning-based, neural end-to-end approach is capable of high-quality text-to-speech (TTS) and state-of-the-art automatic speech recognition (ASR). With regard to the end-to-end ASR model, the connectionist temporal classification (CTC)-based model [1], the attention-based encoder-decoder model [2], recurrent neural network transducer (RNN-T) [3], and transformer-based model [4, 5] have achieved high performance. However, these end-to-end models need a huge amount of paired data of speech and transcription for training.

To reduce the amount of data needed for training, a number of studies have tried data augmentations using unpaired text data. Several of these studies focused on building a language model (LM) with text data and then integrating the LM with or transferring its knowledge to the ASR decoder [6, 7]. Another direction that has been pursued is incorporation of unpaired text data into (part of) the ASR model training [8, 9, 10, 11, 12]. Tjandra *et al.* [8, 9] investigated joint optimization of end-to-end ASR and TTS in a speech chain model. Hayashi *et al.* [10] introduced back-translation for utilizing a large amount of unpaired text data. Masumura *et al*. [11] prepared a shared decoder to train phoneme-to-grapheme (text-only data) and ASR tasks (paired data) and pre-trained the decoder on a phoneme-to-grapheme task. Tang *et al.* [12] jointly trained speech-to-subword and phone-to-subword tasks.

In addition to the above studies, many recent studies have leveraged TTS [13, 14, 15, 16, 17, 18, 19, 20, 21] in training ASR models. The TTS model generates speech data from text-only data, which are mixed with real speech to train the ASR model. Tacotron 2 [22] and transformer-based models [23] can now synthesize human-like speech when they are given a sufficient amount of speech data from a single speaker. TTS is particularly useful for covering out-of-domain words. However, the use of synthesized data brings only a limited improvement compared with using real data since the synthesized features are rather different from real speech. Moreover, the TTS system often generates unrealistic speech.

In this paper, we propose a novel data augmentation scheme using a discrete representation. In this scheme, TTS predicts discrete ID sequences from texts instead of log Mel-scale filterbank (lmfb) features. When we train an ASR model, we also use acoustic features converted from the discrete ID. We adopt vq-wav2vec [24], which is an unsupervised training method to produce a discrete representation. We believe that the use of a discrete representation has two benefits over the standard use of TTS by experimental evaluations: (1) a discrete representation is much easier to predict than the lmfb features of continuous values; (2) it reduces speaker dependency, which the TTS has trouble separating from the segmental information.

2. MODELS FOR TTS AND DISCRETE REPRESENTATION

2.1. Vq-wav2vec

Vq-wav2vec learns a discrete representation of speech frames through a self-supervised future time-step prediction task. The model is based on three convolutional neural networks, in which the encoder produces a representation $\boldsymbol{z_i}$ for each time step i with a rate of 100 Hz; a quantization module converts $\boldsymbol{z_i}$ to a discrete representation $\hat{\boldsymbol{z}}_i$, and an aggregator combines the multiple encoder time steps into a new representation $\boldsymbol{c_i}$. The quantization module replaces the original representation $\boldsymbol{z_i}$ by $\hat{\boldsymbol{z}}_i = \text{concat}(\boldsymbol{e}_{i,1}, ..., \boldsymbol{e}_{i,G})$ from a shared fixed-size codebook $\boldsymbol{E} \in \mathbb{R}^{V \times d/G}$, which contains V representations of size d/G, where d is the dimension of $\hat{\boldsymbol{z}}_i$ and G is the number of groups. We represent each row by an integer index and represent the feature vector $\hat{\boldsymbol{z}}_i$ by the indices $\boldsymbol{w}_i \in [V]^G$, where each element $w_{i,g}$ corresponds to a fixed codebook vector. For instance, when $G = 2$, two-dimensional code is generated, e.g. (15, 24) and (36, 87). We concatenate the elements and make the vocabulary for BERT, e.g. “15-24” and “36-87”, when feeding data to BERT.

In the context prediction, wav2vec loss [25] is defined for steps $k = 1, ..., K$ as:

$$\mathcal{L}_k^{wav2vec} = -\sum_{i=1}^{T-k} \Big(\log \sigma(\hat{\boldsymbol{z}}_{i+k}^{\mathsf{T}}\, h_k(\boldsymbol{c}_i)) + \lambda \mathop{\mathbb{E}}_{\tilde{\boldsymbol{z}} \sim p_n} [\log \sigma(-\tilde{\hat{\boldsymbol{z}}}^{\mathsf{T}}\, h_k(\boldsymbol{c_i}))] \Big) \tag{1}$$

where T is the sequence length, σ is a sigmoid function, h_k is a step-specific affine transformation, and thus $\sigma(\hat{\boldsymbol{z}}_{i+k}^{\mathsf{T}}\, h_k(\boldsymbol{c}_i))$ is the probability of a sample $\boldsymbol{z}_{i+k}$, that is k-steps in the future, being correctly predicted. $\tilde{\hat{\boldsymbol{z}}}$ are negative samples drawn from a uniform distribution p_n on the same utterance. In addition to the wav2vec loss, the Gumbel-Softmax or K-means loss is added between $\hat{\boldsymbol{z}}_i$ and $\boldsymbol{z}_i$. Here, we used K-means clustering and added the loss of vector quantization [26] to $\mathcal{L}_k^{wav2vec}$. As a result, the vq-wav2vec model maximizes $\mathcal{L}^{vq-wav2vec}$ as follows:

$$\mathcal{L}^{vq-wav2vec} = \sum_{k=1}^{K} \mathcal{L}_k^{wav2vec} + (\|\text{sg}(\boldsymbol{z}) - \hat{\boldsymbol{z}}\|^2 + \gamma \|\boldsymbol{z} - \text{sg}(\hat{\boldsymbol{z}})\|^2) \tag{2}$$

where $\text{sg}(x) \equiv x$, $\frac{\mathrm{d}}{\mathrm{d}x}\text{sg}(x) \equiv 0$, and γ is a hyperparameter.

BERT is also used for training a more sophisticated representation of the context features. BERT was trained on vq-codes of 960h training data using the MLM task. We use the last layer's hidden states as the input features of the ASR model.

2.2. FastSpeech 2

For data augmentation of ASR, we need to generate a huge amount of data [13]. We have recently learned that non-autoregressive approaches to TTS are much faster in generating speech compared to autoregressive ones such as Tacotron 2 that we have been utilizing in our work [13] In this work, we chose to use FastSpeech 2 [27], which is a non-autoregressive model based on the transformer-based architecture. It predicts not only lmfb features but also other prosodic features such as duration, F0, and energy. It has CNN-based layers called a duration predictor between the encoder and the decoder that predict the phone duration before the decoder. When we use predictors for other prosodic features, they are also placed before the decoder. The decoder generates lmfb features in parallel from the predicted duration and other information. To train FastSpeech 2, we need to prepare ground-truth values of the duration and other prosodic features. In this study, we used a multi-speaker corpus for training since multi-speaker speech has been shown to be more effective for data augmentation in ASR [14]. In particular, we added a speaker ID and speaker embedding layers [28] between the multi-head attention and the feed-forward module of each encoder and the decoder layer. In inference, we randomly select speaker IDs of the training speakers.

3. PROPOSED METHOD

3.1. Conventional data augmentation by TTS

The conventional data augmentation scheme using TTS has four steps: (1) train the TTS that predicts the lmfb features from the phone sequence; (2) generate lmfb features using unpaired texts; (3) convert lmfb features into a waveform by using a vocoder; (4) perform ASR training using the generated waveform data mixed with real data. In this scheme, the vocoder bridges the gap between the lmfb features of the ASR model and the TTS model.

Although the generated speech data give some improvement in ASR performance, the gain is limited since the TTS does not completely reproduce the real speech. To alleviate this problem, Mimura *et al.* [13] froze the ASR acoustic encoder when training with the synthesized data. Wang *et al.* [18, 19] investigated consistency regularization when TTS is incorporated with ASR. Zheng *et al.* [20] introduced a loss for regularization of the decoder when the ASR model is finetuned for out-of-vocabulary words. Fazel *et al.* [21] investigated a multi-stage training strategy by combining weighted multi-style training, data augmentation, encoder freezing, and parameter regularization. Several studies have used speaker information to generate multi-speaker speech such as a speaker ID [14], VAE latent variables [15], pre-trained speaker verification model [21], and a global style token (GST) [16]. Chen *et al.* [17] jointly trained the pre-trained TTS and ASR using a GAN-based model to increase the acoustic diversity in the synthesized data.

3.2. Data augmentation via discrete ID sequences

A major problem with the conventional data augmentation using TTS is the mismatch between the synthetic and real speech data. Conventionally, lmfb features are used as an intermediate representation for both TTS and ASR. Since an lmfb feature is a continuous value and the loss used for training does not address phonetic constraints, the TTS model often generates unrealistic data that do not exist in real speech. Moreover, neural TTS tends to generate many errors such as too short or repeated speech [29], and multi-speaker TTS

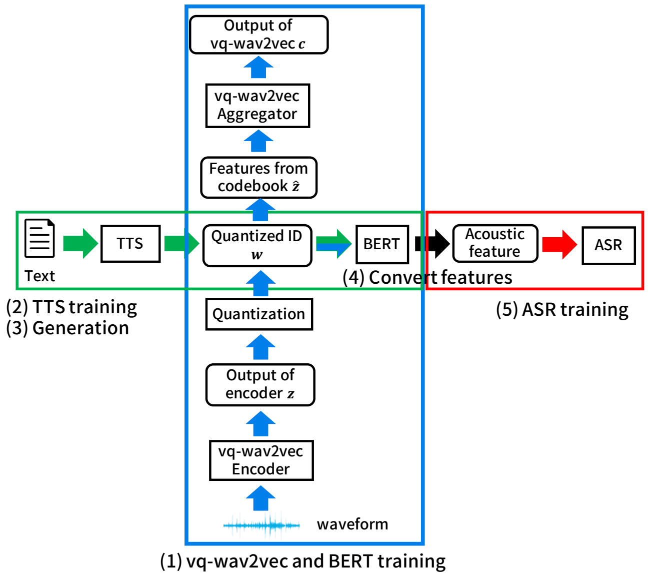

Fig. 1. Overall architecture of the proposed data augmentation for ASR. (1) Train vq-wav2vec and BERT using the real waveform. (2) Perform TTS training using discrete IDs and texts (phones). (3) Generate discrete IDs from texts (phones). (4) Generate features from discrete IDs via BERT (5) Perform ASR training using the final hidden states of BERT and texts (subwords).

model causes more errors than the single speaker model. It is much more difficult to train a multi-speaker TTS model since the amount of training data available is usually quite limited per speaker and multi-speaker features have more variety. These problems pose a bottleneck for effective data augmentation for ASR model training.

To address the above two problems, we introduce a discrete representation to be used for both ASR and TTS. Theoretically, the unreal features which do not exist in real features are not generated because the TTS model selects the discrete IDs from a finite set. Moreover, using discrete IDs for a TTS target makes the TTS task easier since selecting IDs from the fixed classes is considered to be easier than predicting a continuous values. Fig. 1 shows an overview of the proposed method. In this work, we use the vq-wav2vec module for the intermediate representation to convert waveform into the discrete IDs because the vq-wav2vec achieved promising performance of ASR [24]. The output layer of the TTS is a softmax layer corresponding to the discrete IDs. The proposed architecture has five steps.

1. **Train vq-wav2vec and BERT**.
2. Train TTS to predict **discrete IDs** from texts using paired training data.
3. Generate **discrete ID sequences** using TTS from text-only training data for ASR.
4. Convert the ID sequences to ASR features through BERT.
5. Perform ASR training using the generated data mixed with real data.

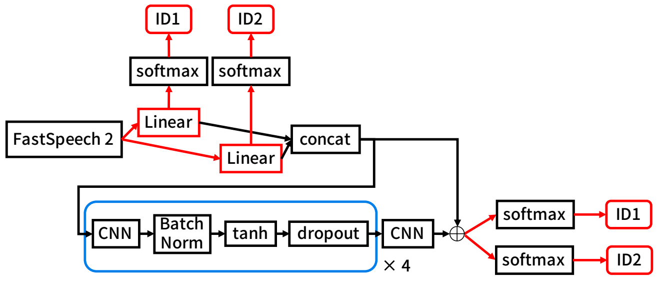

Fig. 2. Postnet architecture of the proposed TTS system. The vq-wav2vec model generates two IDs from a 10ms segment of speech, which the TTS system predicts.

In step 1, we use the original vq-wav2vec with which the two discrete IDs ($G = 2$) are generated every 10ms.

Then, we use the FastSpeech 2-based model to synthesize discrete ID sequences. For this purpose, we replace the lmfb prediction layer of FastSpeech 2 with two output layers to predict these discrete IDs. We add a Postnet in order to divide the hidden state from FastSpeech 2 into two (G) streams of representations for two (G) IDs and to smooth this sequence of representations. Fig. 2 shows the Postnet architecture of the proposed method. The Postnet concatenates the hidden states of the linear layers corresponding to ID1 and ID2 and then applies five convolution layers. We finally sum the outputs of the linear layers and CNN and separate them into two outputs corresponding to ID1 and ID2. For the training, we used two softmax cross-entropy losses for the FastSpeech 2 output and Postnet output. When generating IDs, we use the Postnet output.

For the FastSpeech 2 model training, we also need to align between the transcriptions and audio in advance. For this purpose, we train a CTC-based ASR model on the same data and conduct forced alignment. The original FastSpeech 2 model uses additional prosodic information such as F0 and energy. In particular, we use F0 and energy to build a baseline FastSpeech 2 model that generates lmfb features. However, we do not use this information in our method that predicts discrete ID sequences. In inference, we generate an ID sequence from a phone sequence. Unlike the standard lmfb-output FastSpeech 2 model, we predict IDs by selecting the index with the highest probability in the softmax layer. The generated IDs are concatenated and fed into BERT to generate the ASR features.

4. EXPERIMENTAL EVALUATIONS

4.1. Datasets and tasks

All experiments were conducted with English TTS and ASR models using the LibriSpeech [30] corpus. We converted the sampling rate of the waveforms to 16 kHz for all datasets. To train the TTS model, we used *train-clean-100* of the LibriTTS corpus [31], which is derived from LibriSpeech and designed for TTS tasks. In LibriTTS, *train-clean-100*, which is subset of LibriSpeech *train-clean-100*, has 53 hours' worth of paired data from 247 speakers (male: 123, female: 124).

We used 85 phones and the speaker ID as inputs to the TTS. We converted each word sequence to a phone sequence with an open-source grapheme-to-phone tool[1]. To obtain the ground-truth alignment for FastSpeech 2 training, we also used a CTC-based ASR model trained with LibriTTS *train-clean-100*. We also trained a standard FastSpeech 2 model as a baseline, which generated 80-dimensional lmfb features based on a 50-ms window with a shift of 12.5ms. To obtain F0, we used WORLD [32]. We also used MelGAN [33] conditioning on the lmfb features trained with LibriTTS *train-clean-100* to generate a waveform and then converted it into the ASR-matched lmfb features again.

After data augmentation with the TTS, we trained and evaluated the ASR models with LibriSpeech and TED-LIUM release-2 [34]. We used real speech and transcription data of LibriSpeech *train-clean-100* for the baseline. We augmented the data by using the text data of *train-clean-360* for LibriSpeech testset (Section 4.3.1) and TED-LIUM 2 training set for TED-LIUM 2 testset (Section 4.3.2). We trained ASR models on three training data on each corpus.

- **Baseline model**: *train-clean-100* (real).
- **Augmented model**: *train-clean-100* (real) + synthesized data of *train-clean-360* or TED-LIUM 2.
- **Oracle model**: *train-clean-100* (real) + **real** data of *train-clean-360* or TED-LIUM 2.

We also prepared standard lmfb-input ASR systems for comparison. We used 80-dimensional lmfb features based on a 25-ms window with a shift of 10 ms. In the vq-input ASR systems, we used the 1024-dimensional final hidden states of BERT as the input. We used the pre-trained vq-wav2vec with K-means clustering and RoBERTa$_{\mathrm{BASE}}$ models[2], which were trained with waveforms of LibriSpeech 960h. In the experiments on the vq-input ASR, we did not finetune the vq-wav2vec and BERT models. In all tasks, we used 1000-class subwords based on byte-pair encoding [35]. We used the transcription and the official text data for training the language model of each dataset on the basis of the same subwords.

[1]https://github.com/Kyubyong/g2p

[2]https://github.com/pytorch/fairseq/tree/master/examples/wav2vec

4.2. Network configurations

We built an lmfb-output TTS model and a discrete ID-output TTS model based on FastSpeech 2. The FastSpeech 2-based models had six transformer layers in the encoder and decoder with 384 model dimensions, 1536 feed-forward network dimensions, and 4 attention heads. The Postnet has five CNN layers with kernel size 5. The learning rate was warmed up over the first 1000 updates and then linearly decayed. In the discrete ID-output TTS, we used an L1 loss for the alignment prediction and two softmax cross-entropy losses for the FastSpeech 2 output and Postnet output, which corresponded to IDs. In the lmfb-output TTS, we added the Postnet without blocks for generating discrete IDs for fair comparison. We used five L1 losses in order to predict the alignment, F0, energy, FastSpeech 2-output lmfb features, and Postnet-output lmfb features.

We also built an lmfb-input ASR model and vq-input ASR, whose input consisted of BERT's hidden states. The ASR models were attention-based encoder-decoder models. The encoder had 5-layer BiLSTMs with 320-dimensional hidden states. The decoder was composed of a 1-layer unidirectional LSTM with an attention mechanism. SpecAugment [36] is applied, with two frequency masks with $F = 27$ and two time masks with $T = 100$ in the lmfb-input ASR model and two frequency masks with $F = 260$ and two time masks with $T = 100$ in the vq-input ASR model. We sorted all of the training data in ascending order of speech length and trained the ASR model epoch by epoch. In the augmentation, we also sorted the mixed data and did not try to balance the real and synthetic speech in a batch.

In inference, we set the beam search width to 4 and applied shallow fusion [37] with an LM weight of 0.2. The language models for LibriSpeech and TED-LIUM 2 were composed of four unidirectional LSTM layers with 512 dimensional hidden states.

The pre-trained vq-wav2vec is composed of eight CNN encoder layers with 512 channels and 12 CNN aggregator layers with 512 channels [24]. K-means clustering with two groups of 320 classes ($V = 320, G = 2, d = 512$) was used for vector quantization. The codebook was shared between the two groups. For BERT-based ASR feature generation, we also used pre-trained RoBERTa$_{\mathrm{BASE}}$ models with 12 layers, 768 model dimensions, 3072 feed-forward network dimensions, and 12 attention heads to generate ASR features [24].

4.3. Results

4.3.1. LibriSpeech

Table 1 lists the word error rates (WERs) on the LibriSpeech dev and test sets. With the baseline model using LibriSpeech 100h, the WERs with both lmfb-input and vq-input ASR were over 9% in the "clean" settings and 25% in the "other" settings. When augmented with 360 hours of TTS data, the

Table 1. ASR results (WER) on LibriSpeech.

Method		dev-clean	dev-other	test-clean	test-other
Baseline					
Real 100h	lmfb	9.40	30.22	9.76	32.08
	vq	9.59	25.34	10.14	26.20
Augmented					
Real 100h	lmfb	7.12	29.41	7.87	30.16
+ TTS 360h	**vq**	**6.50**	**21.00**	**6.96**	**22.33**
Oracle					
Real 100h	lmfb	4.70	18.40	5.06	18.65
+ Real 360h	vq	5.54	18.16	5.76	18.92

Table 2. ASR results (WER) on TED-LIUM 2

Method		dev	test
Baseline			
Real 100h	lmfb	34.58	33.75
	vq	31.29	31.99
Augmented			
Real 100h	lmfb	28.11	30.05
+ TTS TED-LIUM 2	**vq**	**21.49**	**22.49**
Oracle			
Real 100h	lmfb	10.56	9.88
+ Real TED-LIUM 2	vq	13.24	13.33

lmfb-based model achieved 2.28-, 0.81-, 1.89-, and 1.92-point improvements from the baseline on dev-clean, dev-other, test-clean, and test-other evaluation sets, respectively. On the other hand, the proposed vq-based augmentation achieved 3.09-, 4.34-, 3.18-, and 3.87-point improvements from the baseline. We assume that the WER reductions from the baseline given by the oracle models trained with 100h + 360h of clean speech are the upper limits of the WER reduction possibly achievable by training data augmentation with synthetic speech. When we see the WERs obtained with the proposed vq-based approach, we find that they have achieved 76.3%, 60.4%, 73.6%, 53.2% of the WER reductions given by the oracle model for dev-clean, dev-other, test-clean, and test-other, respectively. These ratios are much better than those achieved by lmfb-input data augmentation, which were 48.5%, 6.9%, 40.2%, and 14.3%, respectively. From these results, we see that the use of vq-wav2vec-based representation in ASR reduces the discrepancy between real and synthetic speech. When we look at the baseline results, we find that this representation is also less sensitive than lmfb features to the difference between "clean" and "other" settings that is supposed to reflect the differences in recording quality and accents [30]. This may be the reason that we have more accuracy gains in "other" settings compared to lmfb features.

4.3.2. TED-LIUM 2

In the experiment described in the previous section, we used LibriSpeech data to train all of the vq-wav2vec, TTS, and ASR models. In the experiment described in this section, we applied the proposed model to a completely different task (TED-LIUM 2). Table 2 shows the WERs on the dev and test set of TED-LIUM 2. With the baseline using only LibriSpeech 100h, the WERs of the lmfb-input and vq-input ASR models were both over 30% apparently due to the speaking style difference. While the data augmentation with lmfb TTS yielded 6.5-point and 3.7-point improvements on the the dev and test sets respectively, vq-wav2vec TTS achieved much higher 9.8-point and 9.5-point improvements on the respective set. The proposed model filled 54.3% of the WER gap between the baseline and the oracle while the lmfb-based model fills only 26.9% of the gap, when we look at the dev set results. From these results, we see that the proposed vq-based approach is also effective to alleviate the mismatch between the domain of read speech and spontaneous presentations.

Table 3. ASR results (WER) using only synthetic features. In the LibriSpeech task, we evaluated the ASR models on dev-clean and test-clean.

Method			dev	test
TTS LibriSpeech	460h	lmfb	53.97	52.16
		vq	**15.18**	**16.85**
TTS TED-LIUM 2	211h	lmfb	90.41	88.34
		vq	**40.04**	**43.54**

4.3.3. ASR training using only synthetic features

In order to see how it is realistic or not to train ASR models only with synthetic data, we generated 460 hours of synthetic speech and trained ASR models. Table 3 shows WER results. Even though the TTS model was trained with the LibriSpeech dataset, the WERs of the lmfb-input ASR model were over 50% for the LibriSpeech clean data sets. This is because there is a serious mismatch between the generated and real lmfb features. On the other hand, the proposed model gave much smaller WERs of 15.18% and 16.85% on the dev-clean and test-clean sets, respectively. From these results, we see that having at least some 20 percent of real speech or not in the training data makes a huge difference in the ASR accuracy. In this experiment, we again see that the discrepancy from real speech appears much smaller with the vq-based approach and the WER results are encouraging. With the TED-LIUM 2 task, presumably due to the differences in speaking style and recording condition, the lmfb results are drastically deteriorated to around 90 percent while the WERs with the vq-based approach still stay around 40 percent.

5. DISCUSSIONS

Fig. 3 shows a t-SNE visualization for the phones IY (top) and OW (bottom) using 5-speaker lmfb features and vq-wav2vec features extracted from real speech. We see that speakers

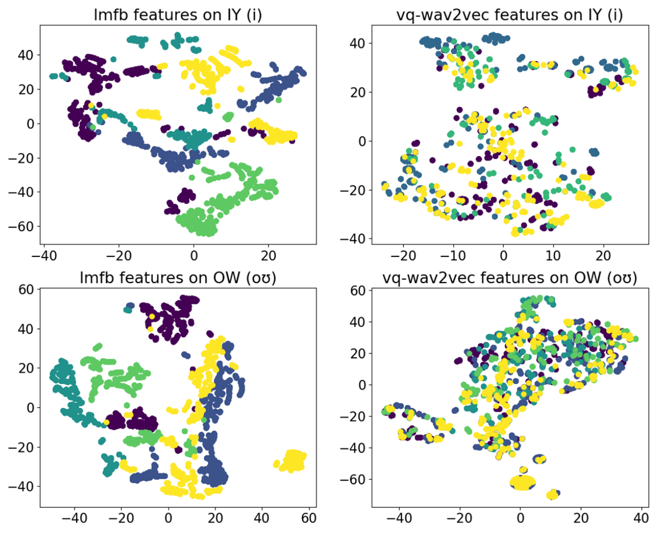

Fig. 3. Comparison of t-SNE visualizations of lmfb and vq-wav2vec features. Each color represents a different speaker.

form clusters within a phone in the lmfb feature distributions whereas speakers are randomly scattered within a phone in the vq-wav2vec feature distributions. This result suggests that vq-wav2vec reduces speaker-dependent information. This property was helpful for stable training of multi-speaker TTS models.

Fig. 4 compares t-SNE visualizations of the real and synthetic features. The real features were extracted from one LibriSpeech dev-clean speaker and the synthetic features were generated using TTS trained with LibriSpeech train-clean-100 data. Speaker IDs were randomly given. We can see that the features are pretty well separated by phones for both lmfb and vq-wav2vec features. On the other hand, real and synthesized features are well mixed for vq-wav2vec but they form clear clusters with lmfb features. This suggests the proposed approach of selecting vq codes from pre-trained codebooks indeed helps to avoid generating many unnatural sequences of features, which, in turn, gives less damage to the training of ASR models using both natural and synthetic data.

6. CONCLUSIONS

We have proposed a novel data augmentation method for ASR that leverages TTS via a discrete representation. The conventional method has a serious mismatch between the generated and real speech, which often results in a limited improvement from TTS-based data augmentation. To mitigate this mismatch, we have introduced vq-wav2vec-based IDs as an intermediate representation instead of lmfb features. In the experimental evaluations, the proposed method resulted in a more effective data augmentation. In addition, the vq-based features were found to exhibit much reduced speaker identities and much less differences between synthesized and real data.

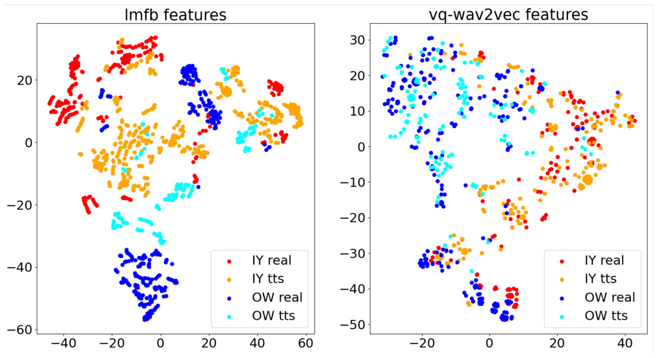

Fig. 4. Comparison of t-SNE visualizations using lmfb and vq-wav2vec features on two phones (IY and OW). Bluish colors represent OW sounds and reddish colors represent IY sounds, darker being real and lighter synthetic.

7. ACKNOWLEDGEMENT

This work was supported by JSPS KAKENHI Grant Number JP19J21031.

8. REFERENCES

[1] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, “Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks,” in *International Conference on Machine Learning (ICML)*, 2006, pp. 369–376.

[2] J. Chorowski and N. Jaitly, “Towards better decoding and language model integration in sequence to sequence models,” in *INTERSPEECH*, 2017, pp. 523–527.

[3] E. Battenberg, J. Chen, R. Child, A. Coates, Y. Gaur Yi Li, H. Liu, S. Satheesh, A. Sriram, and Z. Zhu, “Exploring neural transducers for end-to-end speech recognition,” in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 206–213.

[4] L. Dong, S. Xu, and B. Xu, “Speech-transformer: a no-recurrence sequence-to-sequence,” in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018, pp. 5884–5888.

[5] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, “Conformer: Convolution-augmented Transformer for Speech Recognition,” in *INTERSPEECH*, 2020, pp. 5036–5040.

[6] A. Sriram, H. Jun, S. Satheesh, and A. Coates, “Cold fusion: Training seq2seq models together with language models,” in *INTERSPEECH*, 2018, pp. 387–391.

[7] Y. Bai, J. Yi, J. Tao, Z. Tian, and Z. Wen, “Learn Spelling from Teachers: Transferring Knowledge from Language Models to Sequence-to-Sequence Speech Recognition,” in *INTERSPEECH*, 2019, pp. 3795–3799.

[8] A. Tjandra, S. Sakti, and S. Nakamura, “Attention-based wav2text with feature transfer learning,” in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 309–315.

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

[10] T. Hayashi, S. Watanabe, Y. Zhang, T. Toda, T. Hori, R. Astudillo, and K. Takeda, “Back-translation-style data augmentation for end-to-end asr,” in *Workshop on Spoken Language Technology (SLT)*, 2018, pp. 426–433.

[11] R. Masumura, N. Makishima, M. Ihori, A. Takashima, T. Tanaka, and S. Orihashi, “Phoneme-to-Grapheme Conversion Based Large-Scale Pre-Training for End-to-End Automatic Speech Recognition,” in *INTERSPEECH*, 2020, pp. 2822–2826.

[12] Y. Tang, J. Pino, C. Wang, X. Ma, and D. Genzel, “A general multi-task learning framework to leverage text data for speech to text tasks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6194–6198.

[13] M. Mimura, S. Ueno, H. Inaguma, S. Sakai, and T. Kawahara, “Leveraging sequence-to-sequence speech synthesis for enhancing acoustic-to-word speech recognition,” in *Workshop on Spoken Language Technology (SLT)*, 2018, pp. 477–484.

[14] 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 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6161–6165.

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

[16] N. Rossenbach, A. Zeyer, R. Schluter, and H. Ney, “Generating Synthetic Audio Data for Attention-based Speech Recognition Systems,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7064–7068.

[17] Z. Chen, A. Rosenberg, Y. Zhang, G. Wang, B. Ramabhadran, and P. J. Moreno, “Improving Speech Recognition Using GAN-Based Speech Synthesis and Contrastive Unspoken Text Selection,” in *INTERSPEECH*, 2020, pp. 556–560.

[18] G. Wang, A. Rosenberg, Z. Chen, Y. Zhang, B. Ramabhadran, Y. Wu, and P. Moreno, “Improving speech recognition using consistent predictions on synthesized speech,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7029–7033.

[19] G. Wang, A. Rosenberg, Z. Chen, Y. Zhang, B. Ramabhadran, and P. J. Moreno, “SCADA: Stochastic, Consistent and Adversarial Data Augmentation to Improve ASR,” in *INTERSPEECH*, 2020, pp. 2832–2836.

[20] X. Zheng, Y. Liu, D. Gunceler, and D. Willett, “Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5659–5663.

[21] A. Fazel, W. Yang, Y. Liu, R. Barra-Chicote, Y. Meng, R. Maas, and J. Droppo, “SynthASR: Unlocking Synthetic Data for Speech Recognition,” *arXiv preprint arXiv:2106.07803*, 2021.

[22] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan *et al.*, “Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions,” in *INTERSPEECH*, 2017, pp. 4779–4783.

[23] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, “Neural Speech Synthesis with Transformer Network,” in *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.

[24] A. Baevski, S. Schneider, and M. Auli, “vq-wav2vec: Self-supervised learning of discrete speech representations,” in *International Conference on Learning Representations (ICLR)*, 2020.

[25] S. Schneider, A. Baevski, R. Collobert, and M. Auli, “wav2vec: Unsupervised pre-training for speech recognition,” *arXiv preprint arXiv:1904.05862*, 2019.

[26] A. van den Oord, O. Vinyals, and k. kavukcuoglu, “Neural discrete representation learning,” in *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[27] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, “FastSpeech 2: Fast and high-quality end-to-end text to speech,” in *International Conference on Learning Representations (ICRL)*, 2020.

[28] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 2962–2970.

[29] I. Elias, H. Zen, J. Shen, Y. Zhang, Y. Jia, R. Weiss, and Y. Wu, "Parallel tacotron: Non-autoregressive and controllable tts," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5694–5698.

[30] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.

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

[32] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems*, vol. E99.D, no. 7, pp. 1877–1884, 2016.

[33] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "MelGAN: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[34] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, 2014, pp. 3935–3939.

[35] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1715–1725.

[36] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *INTERSPEECH*, 2019, pp. 2613–2617.

[37] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.