INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Improved Prosody from Learned F0 Codebook Representations for VQ-VAE Speech Waveform Reconstruction

Yi Zhao[1], *Haoyu Li*[1], *Cheng-I Lai*[2], *Jennifer Williams*[3], *Erica Cooper*[1], *Junichi Yamagishi*[1,3]

[1]National Institute of Informatics, Japan
[2]Massachusetts Institute of Technology, USA
[3]University of Edinburgh, UK

{zhaoyi,haoyuli,ecooper,jyamagis}@nii.ac.jp, clai24@mit.edu, j.williams@ed.ac.uk

Abstract

Vector Quantized Variational AutoEncoders (VQ-VAE) are a powerful representation learning framework that can discover discrete groups of features from a speech signal without supervision. Until now, the VQ-VAE architecture has previously modeled individual types of speech features, such as only phones or only F0. This paper introduces an important extension to VQ-VAE for learning F0-related suprasegmental information simultaneously along with traditional phone features. The proposed framework uses two encoders such that the F0 trajectory and speech waveform are both input to the system, therefore two separate codebooks are learned. We used a WaveRNN vocoder as the decoder component of VQ-VAE. Our speaker-independent VQ-VAE was trained with raw speech waveforms from multi-speaker Japanese speech databases. Experimental results show that the proposed extension reduces F0 distortion of reconstructed speech for all unseen test speakers, and results in significantly higher preference scores from a listening test. We additionally conducted experiments using single-speaker Mandarin speech to demonstrate advantages of our architecture in another language which relies heavily on F0.

Index Terms: VQ-VAE, speech synthesis, prosody, representation learning

1. Introduction

Speech signals contain rich factors existing at different linguistic layers such as prosody, content, and timbre simultaneously. Modelling and controlling these factors through neural representation learning is essential for many speech applications [1, 2, 3]. In this paper, we focus on Vector Quantized Variational AutoEncoder (VQ-VAE) [4] since it is a promising self-supervised representation learning approach suitable for tasks such as voice conversion (VC) [4, 5], text-to-speech (TTS) synthesis [6, 7, 8], speech coding [9], and even music generation [10] wherein we need to model and control content, prosody and speaker characteristics separately.

The VQ-VAE paradigm typically consists of three main components: an encoder network, VQ codebooks, and a decoder network. For speech-related applications, the expected role of the encoder is to extract phone- or syllable-equivalent segmental representations, while the decoder reconstructs the input raw speech waveform via a neural vocoder, such as WaveNet [11] or WaveRNN [12]. Between the encoder and decoder is a trainable VQ codebook. The VQ codebook transforms outputs of the encoder network into a set of discrete representations for the down-stream task.

A well-known application of VQ-VAE is voice conversion, which is done by conditioning the decoder with a speaker one-hot vector or a speaker-embedding vector during training, and then simply swapping the speaker vector to convert the speaker vector to a different speaker during inference. In [5], promising conversion results using an English speech database have been reported. However, suprasegmental features such as F0, another important cue in speech signals, are not properly modelled in the voice conversion framework.

From a preliminary experiment, we found that VQ-VAE-based speech waveforms typically have inappropriate prosodic structure in the case of Japanese. This is probably because Japanese is a pitch-accented language, which means that pitch accents directly affect the meaning of words and the perceived naturalness of the speech. However, unlike tonal languages such as Mandarin, in which each syllable is coupled with a specific tone index, Japanese pitch accentual patterns (high or low) of each mora are affected by the information at a different linguistic layer from the syllable layer, such as adjacent words. Hence, we hypothesize that a successful VQ-VAE architecture needs to simultaneously extract not only representations corresponding to the segmental features, but also another set of representations corresponding to the supra-segemental features.

Motivated by this, we propose an extension to VQ-VAE structure utilizing two encoders at the same time. One encoder uses a raw speech waveform for input exactly as the original VQ-VAE does. The other encoder uses F0 trajectory as input to separately learn the pitch patterns as well as other F0-related supra-segmental information. This model was trained using a loss function that jointly considers two types of VQ losses as well as the usual coarse-to-fine waveform losses used for training WaveRNN. Listening test results show that this simple yet effective extension significantly improves prosody and naturalness of reconstructed Japanese speech waveforms. The extended VQ-VAE structure was motivated by Japanese prosody, but it can be applied to speech in any language. Therefore, we also show results of the extended VQ-VAE using a Mandarin speech database for further analysis.

This paper is structured as follows: Section 2 gives an overview of the original VQ-VAE framework, and Section 3 explains our proposed extension for the F0 representation. Section 4 elaborates the details of the extended VQ-VAE. Section 5 shows experimental results in Japanese and Mandarin. Finally, we summarize our findings in Section 6.

2. Overview of VQ-VAE

VQ-VAE is a self-supervised learning technique that uses an encoder network, VQ codebooks, and a decoder network, as defined below:

$$\boldsymbol{z}_{1:N} = \text{Encoder}_{\boldsymbol{\Phi}_1}(\boldsymbol{o}_{1:T}), \tag{1}$$

$$\boldsymbol{e}_{1:N} = \text{Vector\_quantization}_{\boldsymbol{\Phi}_2}(\boldsymbol{z}_{1:N}), \tag{2}$$

$$\widehat{\boldsymbol{o}}_{1:T} = \text{Decoder}_{\boldsymbol{\Phi}_3}(\boldsymbol{e}_{1:N}, \boldsymbol{s}) \tag{3}$$

http://dx.doi.org/10.21437/Interspeech.2020-1615

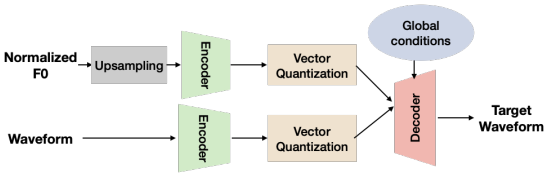

Figure 1: *Overall framework*

The encoder $\mathbf{\Phi}_1$ takes a raw speech waveform of length T, $\boldsymbol{o}_{1:T} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$, as the input and first encodes it into a raw latent vector sequence $\boldsymbol{z}_{1:N} = \{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N\}$. Using the raw latent vectors, the vector quantization function $\mathbf{\Phi}_2$ returns a quantized code vector (that is, centroid) sequence $\boldsymbol{e}_{1:N} = \{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_N\}$. Down-stream tasks can use *indices* of the quantized code vectors as categorical representations. For example, in TTS the indices would represent pseudo phone symbols. Finally, the decoder network $\mathbf{\Phi}_3$ such as WaveNet or WaveRNN reconstructs the speech waveform of the same length T using the quantized code sequence $\boldsymbol{e}_{1:N}$. The decoder network can optionally include a speaker vector $\boldsymbol{s}$ as a global condition.

This VQ-VAE is trained using a penalized log-likelihood function below:

$$\begin{aligned}\mathcal{L}(\mathbf{\Phi}) &= -\log p(\boldsymbol{o}_{1:T}|\boldsymbol{e}_{1:N}; \mathbf{\Phi}_3) \\ &+ \left\|\boldsymbol{e}_{1:N} - \mathrm{sg}[\boldsymbol{z}_{1:N}]\right\|_2^2 + \beta\left\|\boldsymbol{z}_{1:N} - \mathrm{sg}[\boldsymbol{e}_{1:N}]\right\|_2^2,\end{aligned} \tag{4}$$

The first term is a log-likelihood function of the decoder network that measures appropriateness of reconstructed speech waveforms. In the case of WaveRNN, this term corresponds to the multi-class cross-entropy loss using a dual softmax layer that predicts the coarse- and fine-waveform representations [12]. The second term drives the quantized vectors towards the raw latent vectors. The third term is a penalty term that prevents the output of the encoder from growing arbitrarily large [4]. The operator sg[·] zeroes out the gradient back-propagated to the argument. β is a hyper-parameter corresponding to the commitment loss to make sure the encoder commits to the codebook embedding.

3. Proposed Extension for F0

In this paper, we extend the VQ-VAE structure and introduce two encoders. One of them uses a raw speech waveform $\boldsymbol{o}_{1:T}$ of length T as the input, as the original VQ-VAE does. The other encoder uses an upsampled F0 trajectory $\boldsymbol{F}_{1:T} = \{F_1, \cdots, F_T\}$ of length T as the input, as shown in Figure 1:

$$\boldsymbol{z}^o_{1:N} = \mathrm{Encoder}_{\mathbf{\Phi}_{1_o}}(\boldsymbol{o}_{1:T}), \tag{5}$$

$$\boldsymbol{z}^F_{1:N} = \mathrm{Encoder}_{\mathbf{\Phi}_{1_F}}(\boldsymbol{F}_{1:T}), \tag{6}$$

$$\boldsymbol{e}^o_{1:N} = \mathrm{Vector\_quantization}_{\mathbf{\Phi}_{2_o}}(\boldsymbol{z}^o_{1:N}), \tag{7}$$

$$\boldsymbol{e}^F_{1:N} = \mathrm{Vector\_quantization}_{\mathbf{\Phi}_{2_F}}(\boldsymbol{z}^F_{1:N}), \tag{8}$$

$$\widehat{\boldsymbol{o}}_{1:T} = \mathrm{Decoder}_{\mathbf{\Phi}_3}(\boldsymbol{e}^o_{1:N}, \boldsymbol{e}^F_{1:N}, \boldsymbol{s}) \tag{9}$$

Outputs of the two encoders are separately quantized and the speech waveform is then reconstructed using the two sets of code vectors. In the extended VQ-VAE, two sets of representations, i.e., indices of the quantized code vectors for each encoder become available for the down stream tasks. We hope that the discrete representations learned from F0 capture pitch patterns and/or other suprasegmental information. The proposed extension is straightforward, but we observe that it results in impressive improvements for prosody, especially F0 of reconstructed speech waveforms in Japanese.

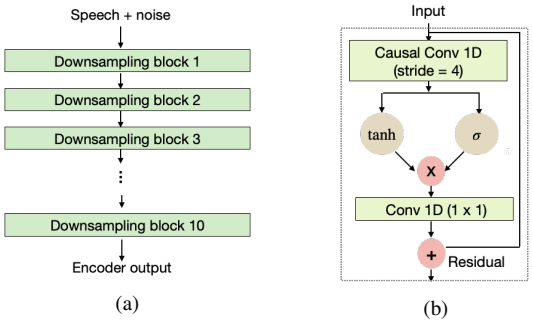

Figure 2: *Encoder framework (a) and downsampling block (b)*

Its loss function can also be defined in a similar way:

$$\begin{aligned}\mathcal{L}(\mathbf{\Phi}) &= -\log p(\boldsymbol{o}_{1:T}|\boldsymbol{e}^o_{1:N}, \boldsymbol{e}^F_{1:N}; \mathbf{\Phi}_3) \\ &+ \left\|\boldsymbol{e}^o_{1:N} - \mathrm{sg}[\boldsymbol{z}^o_{1:N}]\right\|_2^2 + \beta\left\|\boldsymbol{z}^o_{1:N} - \mathrm{sg}[\boldsymbol{e}^o_{1:N}]\right\|_2^2, \\ &+ \gamma(\left\|\boldsymbol{e}^F_{1:N} - \mathrm{sg}[\boldsymbol{z}^F_{1:N}]\right\|_2^2 + \beta\left\|\boldsymbol{z}^F_{1:N} - \mathrm{sg}[\boldsymbol{e}^F_{1:N}]\right\|_2^2),\end{aligned} \tag{10}$$

Here γ is a new hyper-parameter.

This extension is related to VQ-VAE based speech coding by Gârbacea *et al.* [9] wherein a second *decoder* was introduced to predict both a speech waveform and an F0 trajectory at the same time. F0 prediction loss was also introduced in the training process in [9]. However, unlike our extension, a common encoder and codebooks were used. Another related extension comes from *SPEECHSPLIT* [13] wherein three different auto-encoders were used to disentangle the speech signal into components, however their aim was to extract continuous latents.

4. Details of the Extended VQ-VAE

We implemented the extended VQ-VAE based on a public implementation of VQ-VAE[1]. Here we describe details of the extended VQ-VAE used for our experiments.

4.1. Waveform and Upsampled Normalized F0

The input waveform representation is linear PCM. The output waveform is parameterized for coarse and fine parts separately unlike the input waveform, and they are notated as $\widehat{\boldsymbol{o}}_{\boldsymbol{t}} = (c_t, f_t)$ in the following sections. Linear F0 including unvoiced regions is extracted frame-by-frame and then normalized sentence by sentence using the minimum and maximum values of each utterance. This normalization helps the model to focus on the pitch trajectory while casting away speaker information. It can also benefit the discretized latent representation learning since it shrinks the dynamic range of continuous F0 values. Then, the extracted F0 is upsampled via a transposed convolutional layer in order to be aligned with the waveform points. In our implementation, the input waveform is 16-bit linear PCM at 22.05k sampling frequency, and c_t and f_t are both 8-bit.

4.2. Encoder

Encoders for raw waveform and F0 (Eqs. 5 and 6) share the same architecture. Each of them has ten down-sampling blocks (Figure 2 (a)). Each block consists of a 1D convolution layer, followed by another 1D convolution layer and a gated activation layer using tanh and sigmoid functions. Outputs of the gated activation are further filtered using 1D convolution. The number of channels is set to 256 for the first convolution layer, and 128 for the second convolution layer. The final 1D convolution is also set to 128 channels. The residual connection combines sparsified block input with block output to retain block

[1] https://github.com/mkotha/WaveRNN

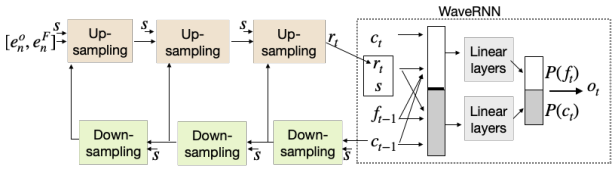

Figure 3: *Decoder framework.* $\boldsymbol{r}_t$ *is output of the last upsampling block.* c_t *and* f_t *represent coarse and fine bits, respectively, of an output waveform* $\boldsymbol{o}_t$ *at time* t*.* $P(c_t)$ *and* $P(f_t)$ *at the outputs are probabilities estimated by the dual softmax layer.* $\boldsymbol{s}$ *is the global condition.*

input information before down sampling. Its diagram is shown in Figure 2 (b). To encourage generalization, we introduce additive and multiplicative Gaussian noise to the waveform and F0 inputs. Without these types of noise, the model tends to be sensitive to small input perturbations, and over-fitting occurs after a certain point in training. The downsampling rate of the entire encoder is approximately $N/T \approx 0.0145$.

4.3. Vector Quantization

In vector quantization of Eq. 7, the encoder output $\boldsymbol{z}_n^o$ at time step n is quantized using the closest vector included in the VQ codebook, as computed by Euclidean distance. The closest code vector $\boldsymbol{e}_n^o$ for each time step n is repeated for the entire sequence to obtain $\boldsymbol{e}_{1:N}^o$. The same operation is conducted for Eq. 8, with separate codebook for F0 to output $\boldsymbol{e}_{1:N}^F$. In our implementation, the waveform codebook consists of 512 code vectors of 128 dimensions. As for the F0 codebook, we varied the number of code vectors from 6 to 512 and found out that ten code vectors are capable enough to capture the F0 trajectory variation. The dimension of the F0 code vector is also 128.

4.4. Decoder

The purpose of the decoder is to reconstruct a speech waveform $\widehat{\boldsymbol{o}}_{1:T}$ using the code vector sequences $\boldsymbol{e}_{1:N}^o$ and $\boldsymbol{e}_{1:N}^F$ together with the global condition $\boldsymbol{s}$. As shown in Figure 3, our decoder of Eq. 9 was implemented using a combination of upsampling blocks, downsampling blocks, and a WaveRNN module.

Up-sampling block: Time scales for the code vectors and waveform are different from each other because the encoder downsamples the code vectors. Therefore we use upsampling blocks so that code vectors ($\boldsymbol{e}_n^o$ and $\boldsymbol{e}_n^F$) are at the same operating rate as the waveform points. Each upsampling block is a GRU layer followed by a transposed convolution network.

Down-sampling block: Since WaveRNN is an auto-regressive (AR) model, it uses the coarse component c_{t-1} of the output waveform $\widehat{\boldsymbol{o}}_{t-1}$ at time $t-1$ as an additional condition for predicting a next waveform point $\widehat{\boldsymbol{o}}_t$ at time t [12]. To do this, the past coarse components are downsampled and combined with the next code vectors. The downsampling block is similar to Figure 2 (b) except for the lack of residual connections. There are three downsampling blocks. The output of each one is connected to the corresponding upsampling block like a U-net [14]. For the AR feedback, teacher-forcing strategy is used for training and predicted waveforms are used for inference.

WaveRNN module: This module takes the output of the last upsampling block $\boldsymbol{r}_t$ and the previous sample $\widehat{\boldsymbol{o}}_{t-1} = (c_{t-1}, f_{t-1})$, and predicts coarse and fine bits at time t via separated softmax layers, not simultaneously but one-by-one. It first predicts coarse bits c_t, then uses c_t to predict fine bits f_t. It consists of one common GRU layer and two separated feedforward layers for coarse and fine bits, respectively. In our framework, the global condition $\boldsymbol{s}$ is inserted as an additional input to all layers in the decoder.

4.5. Global Conditions and Neural Speaker Embedding

We used three types of global conditions, gender, emotion, and speaker, as $\boldsymbol{s}$ of Eq. 9. The gender and emotions are represented using simple one-hot vectors (two and four dimensions, respectively). It is ideal to determine number of emotional classes from data in an unsupervised way [6], but, for simplicity, we used labeled data in this work. The speaker condition is represented using a neural embedding vector obtained from a speaker encoder, which was pre-trained using the Learnable Dictionary Encoding (LDE) and angular softmax [15]. The original dimension of the speaker embedding vector is 512 and we reduced its dimension to 50 using Linear Discriminant Analysis.[2]. For more details, refer to Section 2 of [16] and its implementation[3].

4.6. Expected Down-Stream Tasks

An example of expected down-stream tasks using our extended VQ-VAE is TTS without any manual labels. Since the proposed model allows us to automatically learn segmental and F0-related suprasegemental discrete representations, it is expected that we can build a TTS system without using manually-defined phone sets or pitch accent labels by just predicting the code indices from text. This would be useful for under-resourced dialects and languages. For instance, Japanese dialects have different pitch accent patterns from the standard ones and there are no established methods for annotating them. Some African languages such as Ibibio also use tones [17]. However, the construction and evaluation of down-stream tasks are beyond the scope of this paper. The focus of this paper is to show that the proposed extended VQ-VAE can produce a higher quality of speech thanks to the F0 encoder and its codebook.

5. Experiments

Speech databases: We mainly experimented with Japanese speech databases. We combined the JVS corpus [18], JTES corpus [19], and an in-house corpus introduced in our previous work [20] in order to augment training data. Details of the corpora are as follows: There are a total of 212 speakers (half male and half female), each with around 100 utterances. We used 192 speakers out of the 212 speakers for training and validation. The total number of utterances in the training set is 20,000. The JTES corpus and in-house corpus contain four kinds of emotions: neutral, happy, joy, and angry. The JVS corpus has only neutral speech. We selected 8 speakers and 10 utterances per speaker as the test set. Our test speakers are completely unseen and omitted from both training and validation sets. The testing utterances were common across all speakers. In addition, we conducted a small experiment on a Mandarin dataset to analyze the performance in another tonal language. We used a publicly-available single-speaker Chinese standard Mandarin speech corpus[4]. Around 9,000 utterances were used for training and validation. 80 utterances were used for evaluation.

All speech data was down-sampled to 22.05kHz and precision was converted to 16 bits per sample. The waveform amplitude was normalized to -26 dBov in advance using ITU-T G.191 called "sv56" [21]. The silence segments were also trimmed in advance using Librosa [22]. F0 values were automatically extracted using a CREPE model [23] with a frame shift of 5ms.

[2]To consider the balance of gender, emotion, and speaker, we duplicated the gender embedding by 5 times and the emotion embedding by 10 times. Thus the total dimension of global conditions is 100.

[3]https://github.com/jefflai108/pytorch-kaldi-neural-speaker-embeddings

[4]https://www.data-baker.com/open_source.html

Table 1: *F0 RMSE errors and P563-based estimated MOS scores of each Japanese test speaker and their average. Natural speech MOS score is 4.2 on average. Original VQ-VAE doesn't have F0 encoder but the extended one does.*

Speaker	Gender	VQ-VAE	log F0 RMSE	P563 MOS
Speaker 1	M	Original	0.51	3.8
		Extended	0.30	4.2
Speaker 2	M	Original	0.46	3.6
		Extended	0.20	5.0
Speaker 3	M	Original	0.49	3.8
		Extended	0.27	3.9
Speaker 4	M	Original	0.43	3.7
		Extended	0.24	3.9
Speaker 5	F	Original	0.36	4.0
		Extended	0.25	4.5
Speaker 6	F	Original	0.30	3.7
		Extended	0.23	4.2
Speaker 7	F	Original	0.36	3.0
		Extended	0.27	3.5
Speaker 8	F	Original	0.41	4.1
		Extended	0.27	4.1
Average	M+F	Original	0.42	3.8
		Extended	0.26	4.2

Speaker encoder and speaker embedding: Before training VQ-VAE models, we first trained an LDE-based speaker encoder using a multi-speaker Japanese corpus called ATR-APP[5] to construct a language-dependent speaker encoder. This corpus has thousands of Japanese speakers and we used 135k utterances for training. Using this pre-trained speaker encoder, we extracted speaker embedding vectors from the training set of the above Japanese databases.

VQ-VAE training: Both the original and extended VQ-VAE parameters are optimized using the Adam [24]. Hyperparameters β in Eq. 10 is initialized as 0.001 and linearly increased to 0.01 after 1k steps. γ is set to 10 for the first 10k steps, then until 100k steps it is linearly reduced to 0.1 and remains fixed thereafter. Total number of steps is 1000k and it took 10 days on a TESLA V100 GPU card for each model.

5.1. Objective Evaluation in Japanese

We first computed F0 distortion between input speech and reconstructed speech as an objective evaluation. We also ran the P.563 algorithm [25] for evaluating the reconstruction quality. Results for each test speaker and the average are shown in Table 1. We can see that the proposed extension reduced the F0 distortion for all test speakers as expected. Moreover, we can see that reconstructed speech using the extended VQ-VAE generally has higher P563 scores than that using the original VQ-VAE. The averaged score of the extended VQ-VAE is 4.2 and is the same as that of the input natural speech.

5.2. Subjective Evaluation in Japanese

We conducted an AB preference test and compared the VQ-VAE with and without the F0 encoder subjectively[6]. Subjects are native speakers of Japanese and the number of subjects was 21. Each subject was asked to listen to a total of 20 pairs, and for each pair they were asked to choose the one that sounds better in terms of appropriateness of pitch accents and quality.

Table 2 shows the preference score and 95% confidence intervals for each test speaker, and the average. As we can see from the table, our proposed extension has achieved significantly higher preference scores than the original VQ-VAE for all the test speakers, and this clearly demonstrates that the extended VQ-VAE learned complementary supra-segemental features, which are critical for human perception.

Table 2: *Preference scores and 95% confidence intervals (CI) of each Japanese test speaker and their average.*

Speaker	Original [%]	Extended [%]	95% CI
Speaker 1	8.8	91.2	±3.6
Speaker 2	15.4	84.6	±6.7
Speaker 3	2.4	97.6	±3.4
Speaker 4	6.0	94.0	±6.8
Speaker 5	4.1	95.9	±5.7
Speaker 6	11.3	88.7	±8.8
Speaker 7	7.6	92.4	±7.4
Speaker 8	5.8	94.2	±6.6
Average	7.7	92.3	±2.1

Table 3: *F0 RMSE errors and P563-based estimated MOS scores for Mandarin speaker. Natural speech MOS score is 4.4.*

Speaker	VQ-VAE	log F0 RMSE	P563 based MOS
Chinese	Original	0.22	4.4
	Extended	0.15	4.4

5.3. Extensions to Tonal Languages: Mandarin

We also trained a speaker-dependent VQ-VAE without and with the F0 encoder using the Mandarin database to analyze what happens in another language. Table 3 gives the F0 distortion between input speech and reconstructed speech and P.563-based estimated MOS scores. From this table, we can see that the proposed extension effectively reduced the F0 distortion for the Mandarin speaker, as expected. The RMSE value for the Mandarin speaker is smaller than those of the Japanese speakers, but this is not surprising since the Mandarin model is speaker-dependent whereas the Japanese model is speaker-independent, and all test speakers in Japanese are unseen.

P.563-based MOS of the Mandarin speaker shows a different tendency from the Japanese results in Table 1. The MOS scores of Mandarin reconstructed speech are 4.4 for the VQ-VAE without and with the F0 encoder, which is the same as that of the input speech. This is because Mandarin syllables and a tone index can be coupled and be represented together, unlike the Japanese pitch accents. For learning tones separately from syllables, explicit disentanglement using adversarial loss or mutual information [26] would be needed. This is our next step.

6. Conclusions

This paper proposed an extension of VQ-VAE for learning F0-related suprasegmental information additionally. The extended framework uses a F0 trajectory as additional input and learns discrete representations for F0. We constructed a speaker-independent VQ-VAE model using the Japanese databases and showed that reconstructed speech has lower F0 RMSE for all unseen speakers. A listening test also indicated significantly higher preference scores. An additional experiment using the single-speaker Mandarin database also showed the proposed model reduced F0 distortion in another language. Our future work will include the evaluation of down-stream tasks using learned discrete representations such as TTS and explicit disentanglement of latents for F0 and waveforms.

Acknowledgments Cheng-I is supported by the Merrill Lynch Fellowship, MIT. This work was partially supported by a JST CREST Grant (JPMJCR18A6, VoicePersonae project), Japan, and by MEXT KAKENHI Grants (16H06302, 18H04112, 18KT0051, 19K24373 Japan. The numerical calculations were carried out on the TSUBAME 3.0 supercomputer at the Tokyo Institute of Technology.

[5] https://www.atr-p.com/products/sdb.html

[6] Audio samples are available at: https://nii-yamagishilab.github.io/yi-demo/interspeech-2020/index.html.

7. References

[1] Y.-A. Chung and J. Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," in *Proc. Interspeech 2018*, 2018, pp. 811–815.

[2] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[3] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 12, pp. 2041–2053, 2019.

[4] A. van den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 6306–6315.

[5] S. Ding and R. Gutierrez-Osuna, "Group Latent Embedding for Vector Quantized Variational Autoencoder in Non-Parallel Voice Conversion," in *Proc. Interspeech 2019*, 2019, pp. 724–728.

[6] G. E. Henter, J. Lorenzo-Trueba, X. Wang, and J. Yamagishi, "Deep encoder-decoder models for unsupervised learning of controllable speech synthesis," *arXiv preprint arXiv:1807.11470*, 2018.

[7] X. Wang, S. Takaki, J. Yamagishi, S. King, and K. Tokuda, "A vector quantized variational autoencoder (vq-vae) autoregressive neural f_0 model for statistical parametric speech synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 157–170, 2019.

[8] A. Tjandra, B. Sisman, M. Zhang, S. Sakti, H. Li, and S. Nakamura, "Vqvae unsupervised unit discovery and multi-scale code2spec inverter for zerospeech challenge 2019," *arXiv preprint arXiv:1905.11449*, 2019.

[9] C. Gârbacea, A. van den Oord, Y. Li, F. S. Lim, A. Luebs, O. Vinyals, and T. C. Walters, "Low bit-rate speech coding with vq-vae and a wavenet decoder," in *ICASSP*. IEEE, 2019, pp. 735–739.

[10] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:[TODO]*, 2020.

[11] A. v. d. 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.

[12] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. v. d. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," *arXiv preprint arXiv:1802.08435*, 2018.

[13] K. Qian, Y. Zhang, S. Chang, D. Cox, and M. Hasegawa-Johnson, "Unsupervised speech decomposition via triple information bottleneck," *arXiv preprint arXiv:2004.11284*, 2020.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[15] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *Odyssey 2018, The Speaker and Language Recognition Workshop*, 2018.

[16] E. Cooper, C.-I. Lai, Y. Yasuda, F. Fang, X. Wang, N. Chen, and J. Yamagishi, "Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings," *ICASSP*, pp. 6184–6188, 2020.

[17] M. Ekpenyong, E.-A. Urua, O. Watts, S. King, and J. Yamagishi, "Statistical parametric speech synthesis for ibibio," *Speech Communication*, vol. 56, pp. 243 – 251, 2014.

[18] S. Takamichi, K. Mitsui, Y. Saito, T. Koriyama, N. Tanji, and H. Saruwatari, "JVS corpus: free Japanese multi-speaker voice corpus," *arXiv preprint arXiv:1908.06248*, 2019.

[19] T. Kosaka, Y. Aizawa, M. Kato, and T. Nose, "Acoustic model adaptation for emotional speech recognition using twitter-based emotional speech corpus," in *APSIPA ASC*. IEEE, 2018, pp. 1747–1751.

[20] Y. Zhao, A. Ando, S. Takaki, J. Yamagishi, and S. Kobashikawa, "Does the Lombard Effect Improve Emotional Communication in Noise? — Analysis of Emotional Speech Acted in Noise," in *Proc. Interspeech 2019*, 2019, pp. 3292–3296.

[21] International Telecommunication Union, Recommendation G.191: Software Tools and Audio Coding Standardization, Nov 11 2005.

[22] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[23] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *ICASSP*. IEEE, 2018, pp. 161–165.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] I. Rec, "P. 563: Single-ended method for objective speech quality assessment in narrow-band telephony applications," *International Telecommunication Union, Geneva*, pp. 1–25, 2004.

[26] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*, 2018.