Transformer VQ-VAE for Unsupervised Unit Discovery and Speech Synthesis: ZeroSpeech 2020 Challenge

Andros Tjandra[1], Sakriani Sakti[1,2], Satoshi Nakamura[1,2]

[1]Nara Institute of Science and Technology, Japan
[2]RIKEN, Center for Advanced Intelligence Project AIP, Japan

{andros.tjandra.ai6, ssakti, s-nakamura}@is.naist.jp

Abstract

In this paper, we report our submitted system for the ZeroSpeech 2020 challenge on Track 2019. The main theme in this challenge is to build a speech synthesizer without any textual information or phonetic labels. In order to tackle those challenges, we build a system that must address two major components such as 1) given speech audio, extract subword units in an unsupervised way and 2) re-synthesize the audio from novel speakers. The system also needs to balance the codebook performance between the ABX error rate and the bitrate compression rate. Our main contribution here is we proposed Transformer-based VQ-VAE for unsupervised unit discovery and Transformer-based inverter for the speech synthesis given the extracted codebook. Additionally, we also explored several regularization methods to improve performance even further.

Index Terms: unsupervised unit discovery, zero-speech, clustering, speech synthesize, deep learning

1. Introduction

Many speech technologies such as automatic speech recognition (ASR) and speech synthesis (TTS) has been used widely around the world. However, most such systems only cover rich-resource languages. For low-resource languages, using such technologies remains limited since most of those systems require many labeled datasets to achieve good performance. Furthermore, some languages have more extreme limitations, including no written form or textual interpretation for their context. In the ZeroSpeech challenge [1], we addressed the latter problem by only learning the language elements directly from untranscribed speech.

ZeroSpeech 2020 has three challenges: 2017 Track 1, 2017 Track 2, and 2019. In this paper, we focus on the ZeroSpeech 2019 Track, where the task is called as TTS without Text. This challenge has two major components that need to be addressed: 1) given speech audio, extract subword-like units that contain only context information with an unsupervised learning method and 2) re-synthesize the speech into a different speaker's voice.

To simultaneously solve these challenges, our strategy is to develop a primary model that disentangles the speech signals into two major factors: context and speaker information. After the disentanglement process, a secondary model predicts the target speech representation given the context extracted by the primary model. Following our success in a previous challenge [2] in ZeroSpeech 2019 and several prior publications on quantization-based approaches [3–6], we used a vector quantized variational autoencoder (VQ-VAE) as the primary model and a codebook inverter as the secondary model. To improve our result, we introduced Transformer module [7] to capture the long-term information from the sequential data inside the

VQ-VAE. We also explored several regularization methods to improve the robustness and codebook discrimination scores. Based on our experiment, all these combined methods significantly improved the ABX error rate.

2. Self-Attention and Transformer Module

The Transformer is a variant of deep learning modules that consist of several non-linear projections and a self-attention mechanism [7]. Unlike recurrent neural networks (RNNs) such as a simple RNN or long-short term memory (LSTM) [8] modules, a Transformer module doesn't have any recurrent connection between the previous and current time-steps. However, a Transformer module utilizes self-attention modules to model the dependency across different time-steps.

Given input sequence $X = [x_1, x_2, ..., x_S] \in \mathbb{R}^{S \times d_{in}}$ where S denotes the input length and d_{in} is the input dimension, a Transformer module produces hidden representation $Z = [z_1, z_2, ..., z_S] \in \mathbb{R}^{S \times d_{in}}$. Figure 1 shows the complete process inside a Transformer module.

Figure 1: *A Transformer module*
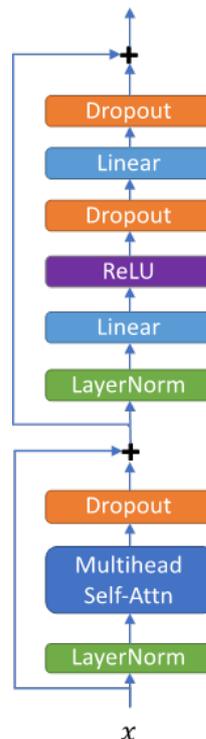
Input sequence X is normalized by a layer-norm and processed by a multi-head self-attention module. Self-attention module inputs are denoted as Q (query), K (key), and V (value) where $\{Q, K, V\} \in \mathbb{R}^{S \times d_k}$. To calculate the output from a self-attention block, we apply the following formula:

$$\texttt{Attention}(Q, K, V) = \texttt{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (1)$$

Since a Transformer module employed multi-head self-attention instead of single head self-attention, input Q, K, V needed to be varied to generate non-identical representation for each head. Therefore, for each head $p \in \{1..P\}$, we projected input X with different matrices W_p^Q, W_p^K, W_p^V and combined the output of many single head self-attentions by concatenations and a linear projection:

$$\texttt{Multi-Head}(Q, K, V) = \texttt{Concat}(h_1, ..., h_P)W^O \quad (2)$$

$$\forall p \in \{1..P\}, h_p = \texttt{Attention}(QW_p^Q, KW_p^K, VW_p^V) \quad (3)$$

where input projection matrices $W_p^Q \in \mathbb{R}^{d_{mdl} \times d_k}, W_p^K \in \mathbb{R}^{d_{mdl} \times d_k}, W_p^V \in \mathbb{R}^{d_{mdl} \times d_k}$, output projection matrix $W^O \in \mathbb{R}^{Pd_k \times d_{mdl}}$ and dimension of $d_k = \frac{d_{mdl}}{P}$. After we got the output from the self-attention, we applied layer normalization [9], two linear projections, and a rectified linear unit (ReLU) activation function.

3. Unsupervised Subword Discovery

Normally, a speech utterance can be factored into several bits of latent information, including context, speaker's speaking style, background noise, emotions, etc. Here, we assume that the speech only contains two factors: context and the speaker's speaking style. In this case, the context denotes the unit that captures the speech information itself in the discretized form, which resembles phonemes or subwords. Therefore, to capture the context without any supervision, we used a generative model called a vector quantized variational autoencoder (VQ-VAE) [10] to extract the discrete symbols. There are some differences between a VQ-VAE with a normal autoencoder [11] and a normal variational autoencoder (VAE) [12] itself. The VQ-VAE encoder maps the input features into a finite set of discrete latent variables, and the standard autoencoder or VAE encoder maps input features into continuous latent variables. Therefore, a VQ-VAE encoder has stricter constraints due to the limited number of codebooks, which enforces the latent variable compression explicitly via quantization. On the other hand, standard VAE encoder has a one-to-one mapping between the input and latent variables. Due to the nature of the unsupervised subword discovery task, a VQ-VAE is more suitable for the subword discovery task compared to a normal autoencoder or VAE. Compared to the machine speech chain [13–16] idea, the encoder is expected to capture only subword-like units can be seen as pseudo-ASR and conditional decoder with speaker embedding can be seen as multi-speaker TTS.

We shows the VQ-VAE model in Fig. 2. Here, we define $\mathbf{E} = [e_1, .., e_K] \in \mathbb{R}^{K \times D_e}$ as a collection of codebook vectors and $\mathbf{V} = [v_1, .., v_L] \in \mathbb{R}^{L \times D_v}$ as a collection of speaker embedding vectors. At the encoding step, input x denotes such speech features as MFCC (Mel-frequency cepstral coefficients) or Mel-spectrogram and auxiliary input $s \in \{1, .., L\}$ denotes the speaker ID from speech feature x. In Fig. 3, we show the
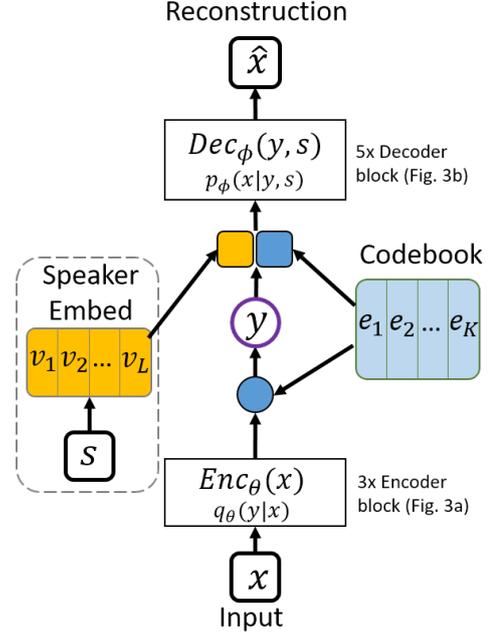

Figure 2: *VQ-VAE for unsupervised unit discovery consists of several parts: encoder $Enc_\theta^{VQ}(x) = q_\theta(y|x)$, decoder $Dec_\phi^{VQ}(y, s) = p_\phi(x|y, s)$, codebooks $\mathbf{E} = [e_1, .., e_K]$, and speaker embedding $\mathbf{V} = [v_1, .., v_L]$.*
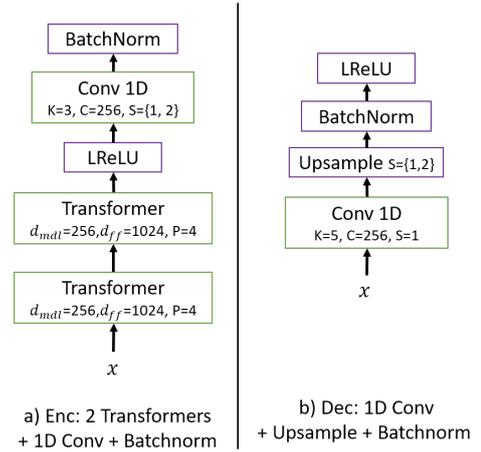

Figure 3: *Building block inside VQ-VAE encoder and decoder: a) Encoder consisted of 2x Transformer layer and 1D convolution with stride to downsample input sequence length; b) Decoder consisted of 1D convolution, followed by up-sampling to recover original input sequence shape.*

details for the building block inside the encoder and decoder modules. Encoder $q_\theta(y|x)$ outputs a discrete latent variable $y \in \{1, ..K\}$. To transform an input in continuous variable into a discrete variable, the encoder first produces intermediate continuous representation $z \in \mathbb{R}^{D_e}$. Later on, we scan through all codebooks to find which codebook has a minimum distance between z and a vector in $\mathbf{E}$. We define this process by the following equations:

$$q_\theta(y = c|x) = \begin{cases} 1 & \text{if } c = \text{argmin}_i \, \text{Dist}(z, e_i) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$e_c = \mathbb{E}_{q_\theta(y|x)}[\mathbf{E}] \quad (5)$$

$$= \sum_{i=1}^{K} q_\theta(y = i|x) \, e_i. \quad (6)$$

where $\text{Dist}(\cdot, \cdot) : \mathbb{R}^{D_e} \times \mathbb{R}^{D_e} \to \mathbb{R}$ is a function that calculates the distance between two vectors. In this paper, we define $\text{Dist}(a, b) = \|a - b\|_2$ as the L2-norm distance.

After the closest codebook index $c \in \{1, .., K\}$ is found, we substitute latent variable z with nearest codebook vector e_c. Later, decoder $p_\phi(x|y, s)$ uses codebook vector e_c and speaker embedding v_s to reconstruct input feature $\hat{x}$.

3.1. VQ-VAE objective

Given input reconstruction across all time-step $\forall t \in [1..T]$, $\hat{X} = [\hat{x}_1, .., \hat{x}_T]$ and closest codebook index $[c_1, .., c_T]$, we calculate the VQ-VAE objective:

$$\mathcal{L}_{VQ} = \sum_{t=1}^{T} - \log p_\phi(x_t|y_t, s) + \gamma \|z_t - \text{sg}(e_{c_t})\|_2^2, , \quad (7)$$

where function $\text{sg}(\cdot)$ stops the gradient:

$$x = sg(x); \qquad \frac{\partial \, \text{sg}(x)}{\partial \, x} = 0. \quad (8)$$

The first term is a negative log-likelihood to measure the reconstruction loss between original input x_t and reconstruction $\hat{x}_t$ to optimize encoder parameters θ and decoder parameters ϕ. The second term minimizes the distance between intermediate representation z_t and nearest codebook e_{c_t}, but the gradient is only back-propagated into encoder parameters θ as commitment loss. The impact from commitment loss controlled with a hyper-parameter γ (we use $\gamma = 0.25$). To update the codebook vectors, we use an exponential moving average (EMA) [17]. EMA updates the codebook $\mathbf{E}$ independently regardless of the optimizer's type and update rules, therefore the model is more robust against different optimizer's choice and hyper-parameters (e.g., learning rate, momentum) and also avoids posterior collapse problem [18].

3.2. Model regularization

3.2.1. Temporal smoothing

Since our input datasets are sequential data, we introduced temporal smoothing between the encoded hidden vectors between two consecutive time-steps:

$$\mathcal{L}_{reg} = \sum_{i=1}^{T-1} \|z_t - z_{t+1}\|_2^2. \quad (9)$$

The final loss is defined:

$$\mathcal{L} = \mathcal{L}_{VQ} + \lambda \mathcal{L}_{reg}, \quad (10)$$

where λ denotes the coefficient for the regularization term.

3.2.2. Temporal jitter

Temporal jitter regularization [3] is used to prevent the latent vector co-adaptation and to reduce the model sensitivity near the unit boundary. In the practice, we could apply the temporal jitter by:

$$j_t \sim Categorical(p, p, 1 - 2 * p) \in \{1, 2, 3\} \quad (11)$$

$$\hat{c}_t = \begin{cases} c_{t-1}, & \text{if } j_t = 1 \text{ and } t > 1 \\ c_{t+1}, & \text{if } j_t = 2 \text{ and } t < T \\ c_t, & \text{else} \end{cases} \quad (12)$$

$$e_t = \mathbf{E}[\hat{c}_t] \quad (13)$$

where p is the jitter probability, c_t is the closest codebook index at time-t and $\hat{c}_t$ is the new assigned codebook index after the jitter operation.

4. Codebook Inverter

A codebook inverter model is used to generate the speech representation given the predicted codebook from our Transformer VQ-VAE. The input is $[\mathbf{E}[c_1], .., \mathbf{E}[c_T]]$, and the output is the following speech representation sequence (here we use linear magnitude spectrogram): $\mathbf{X}^R = [\mathbf{X}_1^R, .., \mathbf{X}_S^R]$.

In Fig. 4, we show our codebook inverter architecture that consists of two multiscale 1D convolutions and three Transformer layers with additional sinusoidal position encoding to help the model distinguish the duplicated codebook positions. Depends on the VQ-VAE encoder architecture, the predicted
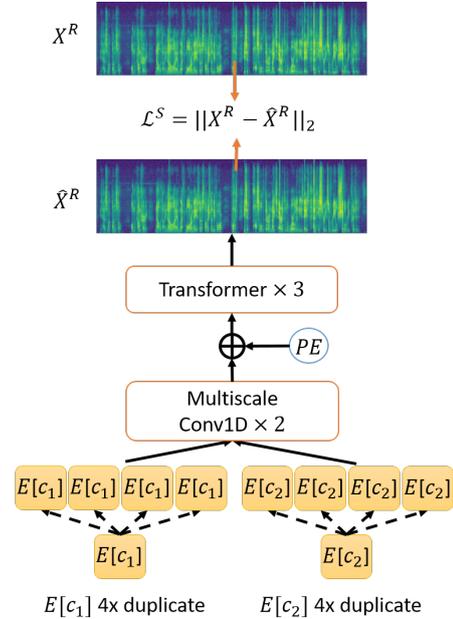

Figure 4: *Codebook inverter: given codebook sequence* $[\mathbf{E}[y_1], .., \mathbf{E}[y_T]]$, *we predict corresponding linear magnitude spectrogram* $\hat{\mathbf{X}}^R = [x_1^R, ..x_S^R]$. *If lengths between S and T are different, we consecutively duplicate each codebook by r-times.*

codebook sequence length T could be shorter than S because the VQ-VAE encoder $q_\theta(y|x)$ have several convolutional layers with stride larger than one. Therefore, to re-align the codebook sequence representation with the speech representation, we duplicate each codebook occurrences $\forall t \in \{1..T\}\, \mathbf{E}[c_t]$ into r copies side-by-side where $r = S/T$. To train a codebook inverter, we set the objective function:

$$\mathcal{L}_{INV} = \|\mathbf{X}^R - \hat{\mathbf{X}}^R\|_2 \qquad (14)$$

to minimize the L2-norm between predicted spectrogram $\hat{\mathbf{X}}^R = \text{Inv}_\rho([\mathbf{E}[c_1], ..., \mathbf{E}[c_T]])$ and groundtruth spectrogram $\mathbf{X}^R$. We defined Inv_ρ as the inverter parameterized by ρ. During the inference step, Griffin-Lim [19] is used to reconstruct the phase from the spectrogram and applied an inverse short-term Fourier transform (STFT) to invert it into a speech waveform.

5. Experiment

In this section, we describe our pipeline, including feature pre-processing, model settings, and hyperparameters.

5.1. Experimental Set-up

There are two datasets for two languages, English data for the development dataset, and a surprising Austronesian language for the test dataset. Each language dataset contains subset datasets: (1) a voice dataset for speech synthesis, (2) a unit discovery dataset, (3) an optional parallel dataset from the target voice to another speaker's voice, and (4) a test dataset. The source corpora for the surprise language are described here [20, 21], and further details can be found [1]. We only used (1) and (2) for training the VQ-VAE codebook and inverter and (4) for evaluation.

For the speech input, we experimented with two different feature representations, such as Mel-spectrogram (with 80 dimensions, 25-ms window size, 10-ms time-steps) and MFCC (with 39 dimensions, 13 main $+\Delta + \Delta^2$, 25-ms window size, 10-ms time-steps). Both MFCC and Mel-spectrogram are generated by the Librosa package [22]. All models are implemented with PyTorch library [23]. We use Adam [12] optimizer with learning rate $1e-4$ for training both models.

5.2. Experimental Results

In this subsection, we report our experiment based on different scenarios and hyperparameters with the English dataset. First, we use different features as the input for VQ-VAE in Table 1. We compared the effect of using log Mel-spectrogram with 80 dimensions versus MFCC$+\Delta + \Delta^2$ with 39 dimensions.

Table 1: *ABX and Bitrate result between MFCC and log Mel-spectrogram features on Transformer VQ-VAE (TrfVQVAE with $K = 128$, stride $4\times$)*

Model	ABX	Bitrate
TrfVQVAE with log-Mel	33.79	171.05
TrfVQVAE with MFCC	21.91	170.42

Based on the result on Table 1, we determined that MFCC provides much better results in terms of ABX error rate. From this point, we use MFCC as the VQ-VAE input features. In Table 2, we compared our proposed model with our submission from previous challenge [2] and we also explored different codebook sizes to determine the trade-off between the ABX error rate and bitrate.

Table 2: *ABX and Bitrate result between our last year submission and new proposed Transformer VQ-VAE.*

Model	ABX	Bitrate
Conv VQVAE (stride $4\times$, K=256) [2]	24.17	184.32
TrfVQVAE (stride $4\times$, K=64)	22.72	141.82
TrfVQVAE (stride $4\times$, K=128)	21.91	170.42
TrfVQVAE (stride $4\times$, K=256)	21.94	194.69
TrfVQVAE (stride $4\times$, K=512)	21.6	217.47

Based on the result on Table 2, by using Transformer VQ-VAE, we outperformed our previous best submission by -2.2 ABX error rate and with a similar bitrate. Later on, by using small codebook sizes $K = 64$, we reduce the bitrate by -30 points compared to $K = 128$, but we sacrifice the +0.8 ABX error rate. We explored different regularization methods to improve transformer VQ-VAEs performance in Table 3. In Table 3, we show the result from Transformer VQ-VAE (stride $4\times$, K=128) with different λ smoothing coefficient, different jitter probability and the combination between both regularization methods. We found out that by combining both temporal smoothing and jittering, we get -1.77 ABX error rate improvement.

Table 3: *ABX and Bitrate result between different temporal smoothing coefficient λ, jitter probability p, and their combination for Transformer VQ-VAE regularization. (Notes: ★ denotes our submitted system. After the submission, we continue the experiment and we found out another hyperparameters combination brings even lower ABX than our submitted system.)*

Model	ABX	Bitrate
TrfVQVAE (stride $4\times$, K=128)	21.91	170.42
+ temp smooth ($\lambda = 1e-2$)	21.88	169.02
+ temp smooth ($\lambda = 5e-3$)	21.67	169.2
+ temp smooth ($\lambda = 1e-3$)	21.75	169.56
+ temp jitter ($p = 0.05$)	21.57	166.19
+ temp jitter ($p = 0.075$)	21.70	164.08
+ temp smooth ($\lambda = 5e-3$) ★ + temp jitter ($p = 0.05$)	20.71	171.99
+ temp smooth ($\lambda = 1e-3$) + temp jitter ($p = 0.05$)	20.14	167.02

6. Conclusions

We described our approach for the ZeroSpeech 2020 challenge on Track 2019. For the unsupervised unit discovery task, we proposed a new architecture: Transformer VQ-VAE to capture the context of the speech into a sequence of discrete latent variables. Additionally, we also use the Transformer block inside our codebook inverter architecture. Compared to our last year's submission, replacing convolutional layers with Transformer gives a significant improvement up to -2.2 ABX error rate. To improve the result further, we investigated several regularization methods. In the end, combining both temporal smoothing and jittering improved the Transformer VQ-VAE performance up to -1.77 ABX error rate compared to the un-regularized model.

7. Acknowledgements

8. References

[1] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao Kam, L. Miskic, C. Dugrain, L. Ondel, A. Black, L. Besacier, S. Sakti, and E. Dupoux, "The Zero Resource Speech Challenge 2019: TTS without T," *Proc. Interspeech 2019*, pp. 1088–1092, 09 2019.

[2] 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," in *Proc. Interspeech 2019*, 2019, pp. 1118–1122. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-3232

[3] J. Chorowski, R. J. Weiss, S. Bengio, and A. v. d. Oord, "Unsupervised speech representation learning using Wavenet autoencoders," *arXiv preprint arXiv:1901.08810*, 2019.

[4] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *arXiv preprint arXiv:1910.05453*, 2019.

[5] R. Eloff, A. Nortje, B. van Niekerk, A. Govender, L. Nortje, A. Pretorius, E. Van Biljon, E. van der Westhuizen, L. van Staden, and H. Kamper, "Unsupervised acoustic unit discovery for speech synthesis using discrete latent-variable neural networks," *arXiv preprint arXiv:1904.07556*, 2019.

[6] A. Tjandra, S. Sakti, and S. Nakamura, "Speech-to-speech translation between untranscribed unknown languages," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 593–600.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

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

[9] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

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

[11] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

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

[13] A. Tjandra, S. Sakti, and S. Nakamura, "Listening while speaking: Speech chain by deep learning," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 301–308.

[14] ——, "Machine speech chain with one-shot speaker adaptation," *Proc. Interspeech 2018*, pp. 887–891, 2018.

[15] A. Tjandra, S. Sakti, and S. Nakamura, "End-to-end feedback loss in speech chain framework via straight-through estimator," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 6281–6285.

[16] ——, "Machine speech chain," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 976–989, 2020.

[17] L. Kaiser, A. Roy, A. Vaswani, N. J. Parmar, S. Bengio, J. Uszkoreit, and N. Shazeer, "Fast decoding in sequence models using discrete latent variables," *ICML*, 2018. [Online]. Available: https://arxiv.org/abs/1803.03382

[18] A. Roy, A. Vaswani, A. Neelakantan, and N. Parmar, "Theory and experiments on vector quantized autoencoders," *CoRR*, vol. abs/1805.11063, 2018. [Online]. Available: http://arxiv.org/abs/1805.11063

[19] 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.

[20] S. Sakti, R. Maia, S. Sakai, T. Shimizu, and S. Nakamura, "Development of HMM-based Indonesian speech synthesis," *Proc. Oriental COCOSDA*, pp. 215–219, 2008.

[21] S. Sakti, E. Kelana, H. Riza, S. Sakai, K. Markov, and S. Nakamura, "Development of Indonesian large vocabulary continuous speech recognition system within A-STAR project," in *Proceedings of the Workshop on Technologies and Corpora for Asia-Pacific Speech Translation (TCAST)*, 2008. [Online]. Available: https://www.aclweb.org/anthology/I08-8004

[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," 2015, pp. 18–25.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.