



Unsupervised Acoustic Unit Representation Learning for Voice Conversion using WaveNet Auto-encoders

Mingjie Chen, Thomas Hain

Department of Computer Science, University of Sheffield, Sheffield, UK

{mchen33, t.hain}@sheffield.ac.uk

Abstract

Unsupervised representation learning of speech has been of keen interest in recent years, which is for example evident in the wide interest of the ZeroSpeech challenges. This work presents a new method for learning frame level representations based on WaveNet auto-encoders. Of particular interest in the ZeroSpeech Challenge 2019 were models with discrete latent variable such as the Vector Quantized Variational Auto-Encoder (VQVAE). However these models generate speech with relatively poor quality. In this work we aim to address this with two approaches: first WaveNet is used as the decoder and to generate waveform data directly from the latent representation; second, the low complexity of latent representations is improved with two alternative disentanglement learning methods, namely instance normalization and sliced vector quantization. The method was developed and tested in the context of the recent ZeroSpeech challenge 2020. The system output submitted to the challenge obtained the top position for naturalness (Mean Opinion Score 4.06), top position for intelligibility (Character Error Rate 0.15), and third position for the quality of the representation (ABX test score 12.5). These and further analysis in this paper illustrates that quality of the converted speech and the acoustic units representation can be well balanced.

Index Terms: voice conversion, acoustic unit discovery

1. Introduction

Unsupervised speech representation learning has gained interest of researchers. It has been shown that representation learning benefits downstream speech applications such as: speech recognition [1], speaker verification [2] and voice conversion [3]. In some recent voice conversion systems [3, 4, 5], auto-encoder based models with disentanglement learning objective functions have achieved good performance. The aim of the disentanglement learning in these systems is to remove the speaker information and retain the speech content information.

Most current deep neural network (DNN) based techniques for speech processing tasks such as automatic speech recognition (ASR) [6, 7, 8] and text-to-speech (TTS) [9, 10, 11] rely on annotated resources or expert knowledge. It still remains a challenge to utilize speech processing techniques for languages that lack resources. Zero Resource Speech Challenge series [12, 13, 14] aim to explore speech processing techniques for a 'low-resource' situation. In the ZeroSpeech 2015 challenge and ZeroSpeech 2017 challenge, the key objective is to learn a representation of speech which is robust to speaker variations. The ZeroSpeech 2019 challenge was expanded to a multi-task scenario covering both: acoustic unit discovery and voice conversion. Participants were required to submit the obtained acoustic units representation and the converted speech. The ZeroSpeech 2020 challenge consolidates the ZeroSpeech 2017 challenge and the ZeroSpeech 2019 challenge.

In the ZeroSpeech 2015 & 2017 challenge, the best performance methods were based on Dirichlet Process Gaussian Mixture Model (DPGMM) [15]. As the discriminative acoustic units representation, clustering posterior grams [16, 17] obtained from the DPGMM have been used. However, these posterior grams are still found to contain the speaker information. In order to learn a speaker-invariant representation, Higuchi et al. [18] used DNN bottleneck feature and speaker adversarial training. Besides, vocal tract length normalization (VTLN) [19] and fMLLR for speaker normalization [17] were also used. Feng et al. [20] used a disentanglement learning model [3] combined with the DNN bottleneck feature learning system. The state of the art acoustic unit discovery performance [21] of the ZeroSpeech 2017 challenge was obtained by using VQVAE [22] with a WaveNet [23] decoder and time-jitter regularization.

More recently, in the context of multi-task learning, the ZeroSpeech 2019 challenge was dominated by use of auto-encoder with discrete latent variable such as VQVAE [22]. VQVAE obtains the representation in a discrete latent space and generates the converted speech from a discrete latent space. However, it is notable that VQVAE based systems [24, 25] suffered from poor quality of the reconstructed speech. We hypothesize that the cause of this phenomenon derives from two main reasons: (1) the speech is generated in a two-stage process with independently trained vocoder models (2) the discrete latent space with low complexity might cause the fine-grained acoustic units information to be lost.

This work aims to improve the speech quality and retain the discriminability of the representation. In order to learn a better representation of the fine-grained acoustic units, the main idea is to increase the complexity of the latent representation in a WaveNet [22] auto-encoder model. The contribution can be summarised as following: (1) an auto-encoder with a WaveNet decoder is used to directly generate waveform data; (2) instead of using vector quantization as in VQVAE, we propose to utilize two alternative disentanglement learning methods. More specifically, we propose to use sliced vector quantization module [26], aiming to increase the complexity of the discrete latent space. Furthermore, we propose to add instance normalization [27] layers to the WaveNet auto-encoder model, which has a continuous latent representation.

2. Proposed Methods

In this section, methods submitted to the 2019 part of the ZeroSpeech 2020 challenge are introduced. The 2019 part is a multi-task scenario including two sub-tasks: acoustic unit discovery and voice conversion. In the context of multi-task learning, the challenge requires the participants to develop a representation of acoustic units without supervision. The representation are supposed to be speaker-invariant. Based on the representation, the participants need to generate the converted speech

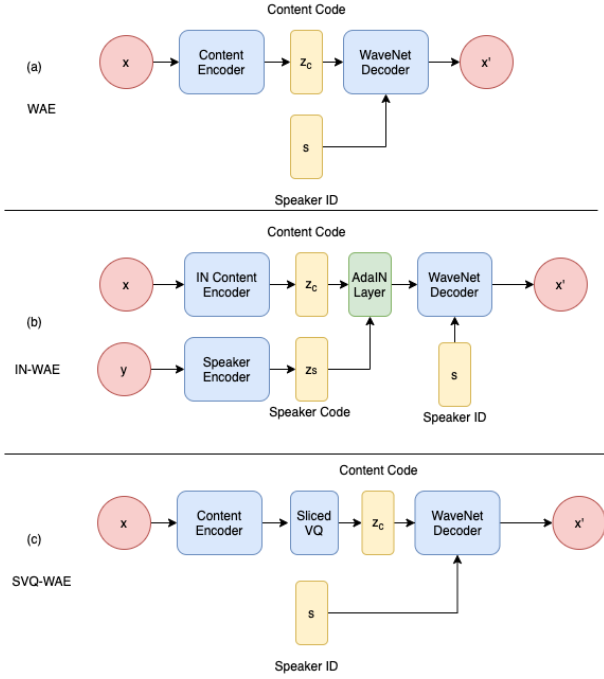


Figure 1: System architectures: (a) WaveNet auto-encoder (WAE), (b) instance normalization WaveNet auto-encoder (IN-WAE), (c) sliced vector quantized WaveNet auto-encoder (SVQ-WAE) x is the input speech, z_c is the content code, z_s is the speaker code, s is the speaker ID input, x' is the reconstructed speech

given the target speaker ID.

Our submission utilizes two different methods which we name as instance normalization WaveNet auto-encoder (IN-WAE) and sliced vector quantized WaveNet auto-encoder (SVQ-WAE) respectively. These are independent submissions in this challenge. IN-WAE is a WaveNet auto-encoder model incorporated with the instance normalization [27] layers. SVQ-WAE is a WaveNet auto-encoder model with a sliced vector quantization [26] bottleneck module.

2.1. WaveNet Auto-encoder

Figure 1(a) illustrates the architecture of the WaveNet auto-encoder model [21]. Let $x \in X$ be the input speech data. The content encoder converts the acoustic unit information into the content code z_c . Then the WaveNet [23] decoder generates speech data conditioning on the speaker ID s and the content code z_c . At training time, s is the source speaker ID, which is associated with the input x . The objective function is to reconstruct the input data x . At conversion time, s is the target speaker ID, and the model produces converted speech data. As for acoustic unit discovery, the content code z_c is regarded as the representation of acoustic unit information.

The content encoder contains six 1D convolutional layers and four residual ReLU [28] layers. The six convolutional layers can be separated to three groups: (1) 2 layers with kernel size 3 and stride 1; (2) 2 down-sampling layer with kernel size 4 and stride 2; (3) 2 layers with kernel size 3 and stride 1. The down-sampling rate of the content encoder is controlled by the stride of the convolutional layer. For example, two convolutional layers with stride 2 enables a 25 Hz frame rate latent code.

The content code z_c and the speaker ID s are the inputs to the WaveNet decoder. The WaveNet model contains 4 up-sampling layers and 20 dilation convolutional layers.

2.2. Instance Normalization WaveNet Auto-encoder

Instance Normalization [27] (IN) and Adaptive Instance Normalization [29] (AdaIN) were proposed for image style transfer. IN normalizes the feature for each sample and each feature channel. Chou et al. [30] has shown that IN can produce disentanglement between content and speaker information. AdaIN is an extension of IN. AdaIN normalizes the feature as IN, then it adapts the feature to the target style given by a style input data.

Let $m \in \mathbb{R}^{B \times C \times T}$ be the output feature map of a convolutional layer in a deep neural network, where B is the batch size, C is the number of channels, T is the length of feature frames. Let $m_{b,c,t}$ represents the element of b th sample, c th channel and t th frame. The mean μ_{bc} and standard deviation σ_{bc} of c th channel and b th sample can be obtained by using the following equations.

$$\mu_{bc} = \frac{1}{T} \sum_{t=1}^T m_{b,c,t} \quad (1)$$

$$\sigma_{bc}^2 = \frac{1}{T} \sum_{t=1}^T (m_{b,c,t} - \mu_{bc})^2 \quad (2)$$

The output of the IN layer is the normalized feature.

$$o_{b,c,t}^{IN} = \frac{m_{b,c,t} - \mu_{bc}}{\sqrt{\sigma_{bc}^2 + \epsilon}} \quad (3)$$

where $o_{b,c,t}^{IN}$ is the normalized feature map, ϵ is a parameter that avoids numerical instability.

AdaIN receives the style input y and adapts the normalized feature map.

$$o_{b,c,t}^{AdaIN} = \sigma(y) o_{b,c,t}^{IN} + \mu(y) \quad (4)$$

where $\sigma(y)$ and $\mu(y)$ are trainable functions. Following previous work [30] adding IN and AdaIN layers to the auto-encoder model, the IN-WAE extends the WAE by adding the IN layer to the content encoder and adding the AdaIN layer ahead of the WaveNet decoder. Incorporating with the IN layer, [30] has shown the content encoder can normalize the global speaker variation while retaining the fine-grained acoustic units information. The IN-WAE utilizes the same content encoder architecture as the WAE. The IN layer is added [30] to the IN content encoder behind every two layers (2,4,6,8,10 th layer).

As shown in Figure 1(b), since the AdaIN layer requires a speaker code z_s , a speaker encoder is used to derive the speaker information from the speech input y . The speaker encoder contains 3 convolutional layers. The global average pooling layer is used as the last layer of the speaker encoder, which compresses the feature into one vector. The output of the speaker encoder is the speaker code z_s , which is the input to the AdaIN layers. As in Equation 4, z_c is adapted according to the speaker code z_s . The WaveNet decoder generates speech data conditioning on the output of the AdaIN layer and the speaker ID.

2.3. Sliced Vector Quantization WaveNet Auto-encoder

VQVAE [22] is a variant of variational auto-encoder (VAE) [31]. VQVAE encodes data to a discrete latent space through Vector Quantization (VQ) bottleneck module. VQVAE consists of three modules: the encoder, the VQ module and the decoder.

The VQ module contains a codebook $M \in \mathbb{R}^{K \times D}$, which is regarded as a collection of D dimensional embeddings, where K is the number of embeddings. m_k is the k th embedding in the codebook. Let the output of the encoder be z_e , which is the input to the VQ module. The output of the VQ module z_q is computed as the nearest neighbour of z_e in the latent embedding space M .

$$z_q = \operatorname{argmin}_{m_k \in M} \|z_e - m_k\|_2 \quad (5)$$

The decoder receives z_q and reconstructs the data. The objective function of VQVAE can be written as following formula.

$$\mathcal{L} = \log p(x|z_q) - \|sg(z_e) - z_q\|_2 - \beta \|z_e - sg(z_q)\|_2 \quad (6)$$

The 2nd and 3rd term in Equation 6 are VQ losses, sg is the stop gradient function where the backward gradient is 0. Since the training speed of the encoder and the decoder are different, sg function is used to learn the encoder and the decoder parameters separately. β is a hyper-parameter that balances two VQ losses.

$$sg(x) = \begin{cases} x & \text{forward} \\ 0 & \text{backward} \end{cases} \quad (7)$$

Since the VQ module utilizes argmin function, which is non-differentiable, the straight-through [32] trick is used for gradient estimation. The straight-through trick maps the gradient from z_q to z_e .

The sliced Vector Quantization (Sliced-VQ) module splits the output of the encoder z_e into N slices.

$$z_e = \operatorname{concat}(z_e^1, \dots, z_e^n, \dots, z_e^N) \quad (8)$$

where $z_e^n \in \mathbb{R}^{D//N}$, concat is the concatenate function which concatenates all the feature slices. The Sliced-VQ operates N parallel sub-VQs on the feature slices $\{z_e^n\}_1^N$. The sub-codebook for each sub-VQs were defined as $\{M^n\}_1^N$ where $M^n \in \mathbb{R}^{K \times (D//N)}$. The output of each sub-VQ module can be computed as:

$$z_q^n = \operatorname{argmin}_{m_k^n} \|z_e^n - m_k^n\|_2 \quad (9)$$

Then the concatenation of all z_q^n forms the final output of the Sliced-VQ module.

$$z_q = \operatorname{concat}(z_q^1, \dots, z_q^n, \dots, z_q^N) \quad (10)$$

As illustrated in Figure 1(c), the content encoder feeds the feature to the Sliced-VQ module, then the output of the Sliced-VQ module is the content code z_c which can be used as the input to the acoustic unit discovery task. The WaveNet decoder receives the content code z_c and speaker ID s , then generates the speech data.

3. Experiment Setup

The following describes the experiment setup for the 2019 part of the ZeroSpeech 2020 challenge. First, the dataset and the evaluation metrics are introduced. Then the implementation details of the submissions are described.

3.1. Dataset

The 2019 part of the ZeroSpeech 2020 challenge corpus contains two languages: English dataset for development and a surprise language [33, 34] dataset for test. For each language, the dataset is split into four parts: the train unit dataset, the train voice dataset, the train parallel voice set and the test set. The

Table 1: Hyper-parameter exploration: latent representation frame rate for the IN-WAE, the VQ-WAE and the SVQ-WAE

Model	50 Hz		25 Hz	
	ABX	Bit-rate	ABX	Bit-rate
IN-WAE	19.13	820.08	20.19	385.75
VQ-WAE	33.31	328.50	31.44	163.89
SVQ-WAE(2 slices)	32.68	587.17	26.24	376.82
SVQ-WAE(4 slices)	31.39	790.68	26.06	377.05

train unit dataset is used for developing acoustic units representations. The English training unit dataset contains 15 hours data for about 100 speakers. The train voice dataset is the training data for target speakers for voice conversion task. The English train voice dataset contains two speakers, 2 hours data per speaker. For surprise language, there is only one target speaker and 1.5 hours data. For the surprise language dataset, the train unit dataset contains 15 hours for 150 speakers and the train voice dataset contains 1 speaker with 1.5 hours data.

In our experiment, the train unit dataset and the train voice dataset are used. The English dataset is used for training and tuning hyper-parameters. The hyper-parameters are kept fixed for training on surprise language dataset. The official evaluation process includes subjective and objective evaluations. The objective evaluation includes two metrics: Machine ABX [35] and bit-rate. Both of two objective evaluation metrics focus on the quality of the acoustic unit representation. The machine ABX measures the discriminability of the representations. The bit-rate measures the compression rate of the representation. The subjective evaluation includes: mean opinion score (MOS), similarity and character error rate (CER). Both the MOS score and the similarity are a scalar in range [1,5]. The MOS score represents the naturalness of the converted speech, while the similarity represents the similarity with the target speaker. The challenge organisers also conduct human evaluation according to the transcriptions and uses CER to measure the intelligibility of the produced speech.

3.2. Implementation

13-dimensional MFCCs with 10 ms step size and 25 ms window size were used as the speech feature. The MFCCs are concatenated with the first and the second derivatives. Mean and variance normalization is conducted. The length of the speech segment is 32 frames (320 ms) and the output waveform length is 5120 samples. The implementation¹ uses the PyTorch [36] toolkit. The Adam [37] optimizer with learning rate 4e-4 was used. One single GTX-1080Ti GPU is used for training. The batch size is 10. The IN-WAE model is trained for 600k steps on the English training dataset. The SVQ-WAE model is trained for 400k steps. The training takes one day for every 100k steps.

4. Results

4.1. Latent Representation Frame Rate

In Table 1, the effect of latent representation frame rate on the discriminability and the compression rate of the acoustic units representation is explored. For the IN-WAE, the 50 Hz model obtains better ABX score (19.13) than the model with 25 Hz frame rate (20.19). However, the model with 50 Hz frame rate

¹code: https://github.com/MingjieChen/wavenet_autoencoders

Table 2: Hyper-parameter Exploration for the SVQ-WAE

#Slices	ABX/Bit-rate		
	k=128	k=512	k=1024
N=1	31.44/163.89	30.91/171.98	28.05/175.84
N=2	26.24/ 376.82	28.80/339.55	27.59/303.69
N=4	26.06 /377.05	26.92/379.52	27.06/379.76

Table 3: Comparing results of test dataset (surprise language)

Model	MOS	CER	Similarity	ABX	Bit-rate
Baseline	2.23	0.67	3.26	27.46	74.55
Topline	3.49	0.33	3.77	16.09	35.2
IN-WAE	4.06	0.15	2.67	12.5	387.83
SVQ-WAE	2.28	0.55	2.5	16.47	384.23

gets higher bit-rate (820.08) than 25 Hz (385.75). The number of the embeddings in the codebook of the SVQ-WAE model is kept as 128. Table 1 compares the VQ-WAE model, the 2 slices SVQ-WAE and the 4 slices SVQ-WAE. Comparing two frame rate options, the trend is that, 25 Hz frame rate obtains better ABX score than 50 Hz. The best ABX score (26.06) is obtained when frame rate is 25 Hz with 4 slices.

For the IN-WAE, the 50 Hz model shows better discriminability than the 25 Hz model, because higher compression rate might causes the fine-grained acoustic units information such as short phones to be lost. As for the SVQ-WAE, the 25 Hz model shows better discriminability than the 50 Hz model. Comparing two proposed methods, the IN-WAE shows better discriminability than the SVQ-WAE.

4.2. Bottleneck Shape in the SVQ-WAE

The hyper-parameters in terms of the shape of the sliced-VQ module in the SVQ-WAE are explored. This part of experiment explores the number of the embeddings k in codebook of the sliced-VQ module and the number of slices N . The frame rate is kept as 25 Hz. In Table 2, the best ABX score (26.06) is when k is 128 and N is 4, and the best bit-rate (163.89) is when N is 1 and k is 128. The trend is that the ABX score gets lower and the bit-rate gets higher as the number of the slices N increasing. It means that the latent space is getting more complex as N increases. And a higher complexity of the latent space benefits the modeling of the acoustic units information. Moreover, a higher N also means a higher bit-rate. It is also notable that increasing the number of embeddings k has a negative effect on both the ABX score and the bit-rate.

4.3. ZeroSpeech Challenge Results

The challenge official baseline method is a combination of a DPGMM [15] system and a Merlin [38] system. The topline method is a combination of an ASR and a TTS systems trained with annotated data.

Table 3 and 4 present the result of our submissions and the provided official systems on developing dataset (English) and test dataset (surprise language) respectively. The total number of submissions was 22, including two official systems. On the surprise language dataset, the IN-WAE obtains the top position for both the naturalness (MOS score 4.06) and the intelligibility (CER 0.15). Meanwhile, it achieves the third position for discriminability of the representation (ABX score 12.5). However, it is obvious that the speaker similarity is worse than the

Table 4: Comparing results of developing dataset (English)

Model	MOS	CER	Similarity	ABX	Bit-rate
Baseline	2.14	0.77	2.98	35/63	71.98
Topline	2.52	0.43	3.1	29.85	37.73
IN-WAE	3.61	0.18	2.57	20.19	385.75
SVQ-WAE	2.88	0.47	2.35	26.06	377.05

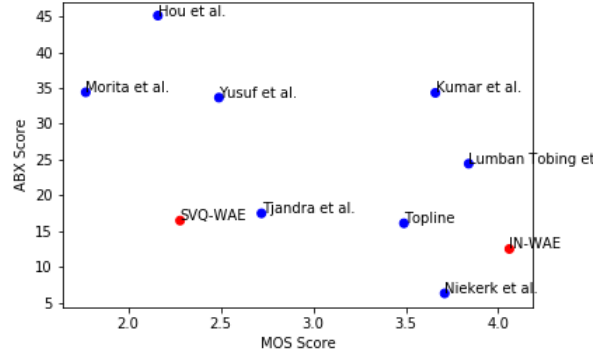


Figure 2: The result comparison of ABX score and MOS score for part of the submissions in ZeroSpeech challenge 2020 test data

official baseline system. The SVQ-WAE obtains the fifth position for ABX score, whereas it does not show an advantage on any other metrics. Comparing the IN-WAE and the SVQ-WAE, the IN-WAE with continuous representation has advantages on both speech quality and representation discriminability. For the English dataset, as shown in Table 4, the IN-WAE achieves the third position for both naturalness (MOS score 3.61) and representation discriminability (ABX score 20.19) and the top position for intelligibility (CER 0.18). The SVQ-WAE achieves the 6th position on ABX score and the 8th on MOS score. Both the IN-WAE and the SVQ-WAE are not showing competitive performance for speaker similarity. The reason might be that the latent representation still contains speaker information with an increased complexity.

4.4. Challenge Result Comparison

We aimed to improve the quality of the speech and meanwhile keep the discriminability of the representation. Figure 2 plots the results of a part of submissions in this challenge. As shown in Figure 2, the IN-WAE obtains a better MOS score while a worse ABX score than [39]. The SVQ-WAE does not obtain good MOS score however it still gets competitive performance on ABX score.

5. Conclusions and Future Work

We proposed to incorporate WaveNet auto-encoder with instance normalization and sliced-VQ respectively. In the ZeroSpeech 2020 challenge, the IN-WAE obtains competitive performance on naturalness, intelligibility and representation discriminability. Meanwhile, the SVQ-WAE obtains competitive representation discriminability. In future work, the methods that can achieve good speaker similarity for voice conversion will be explored. Moreover, the techniques that can accelerate the WaveNet auto-encoder model inference will be investigated.

6. References

- [1] A. Baevski, S. Schneider, and M. Auli, “vq-wav2vec: Self-supervised learning of discrete speech representations,” in *International Conference on Learning Representations*, 2020.
- [2] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, “An Unsupervised Autoregressive Model for Speech Representation Learning,” in *Proc. Interspeech 2019*, 2019, pp. 146–150.
- [3] W.-N. Hsu, Y. Zhang, and J. Glass, “Unsupervised learning of disentangled and interpretable representations from sequential data,” in *Advances in neural information processing systems*, 2017, pp. 1878–1889.
- [4] L. Yingzhen and S. Mandt, “Disentangled sequential autoencoder,” in *ICML*, 2018, pp. 5670–5679.
- [5] J.-c. Chou, C.-c. Yeh, H.-y. Lee, and L.-s. Lee, “Multi-target voice conversion without parallel data by adversarially learning disentangled audio representations,” *Proc. Interspeech 2018*, pp. 501–505, 2018.
- [6] A. Graves, N. Jaitly, and A.-r. Mohamed, “Hybrid speech recognition with deep bidirectional lstm,” in *ASRU 2013*. IEEE, 2013.
- [7] V. Peddinti, D. Povey, and S. Khudanpur, “A time delay neural network architecture for efficient modeling of long temporal contexts,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [8] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, “Purely sequence-trained neural networks for asr based on lattice-free mmi,” in *Interspeech*, 2016, pp. 2751–2755.
- [9] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, “Tacotron: Towards end-to-end speech synthesis,” *Proc. Interspeech 2017*, pp. 4006–4010, 2017.
- [10] 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 *ICASSP 2018*. IEEE, 2018, pp. 4779–4783.
- [11] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, “Deep voice 3: Scaling text-to-speech with convolutional sequence learning,” *arXiv preprint arXiv:1710.07654*, 2017.
- [12] M. Versteegh, R. Thiolliere, T. Schatz, X. N. Cao, X. Anguera, A. Jansen, and E. Dupoux, “The zero resource speech challenge 2015,” in *Sixteenth annual conference of the international speech communication association*, 2015.
- [13] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, “The zero resource speech challenge 2015: Proposed approaches and results,” in *SLTU*, 2016, pp. 67–72.
- [14] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. W. Black *et al.*, “The zero resource speech challenge 2019: Tts without t,” *arXiv preprint arXiv:1904.11469*, 2019.
- [15] J. Chang and J. W. Fisher III, “Parallel sampling of dp mixture models using sub-cluster splits,” in *Proc. NIPS*, 2013, pp. 620–628.
- [16] T. K. Ansari, R. Kumar, S. Singh, S. Ganapathy, and S. Devi, “Unsupervised hmm posteriors for language independent acoustic modeling in zero resource conditions,” in *ASRU 2017*. IEEE, 2017, pp. 762–768.
- [17] M. Heck, S. Sakti, and S. Nakamura, “Feature optimized dpmm clustering for unsupervised subword modeling: A contribution to zerospeech 2017,” in *ASRU 2017*. IEEE, 2017, pp. 740–746.
- [18] Y. Higuchi, N. Tawara, T. Kobayashi, and T. Ogawa, “Speaker Adversarial Training of DPGMM-Based Feature Extractor for Zero-Resource Languages,” in *Proc. Interspeech 2019*, 2019, pp. 266–270.
- [19] H. Chen, C.-C. Leung, L. Xie, B. Ma, and H. Li, “Multilingual bottle-neck feature learning from untranscribed speech,” in *ASRU 2017*. IEEE, 2017, pp. 727–733.
- [20] S. Feng, T. Lee, and Z. Peng, “Combining Adversarial Training and Disentangled Speech Representation for Robust Zero-Resource Subword Modeling,” in *Proc. Interspeech 2019*, 2019, pp. 1093–1097.
- [21] 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.
- [22] A. van den Oord, O. Vinyals *et al.*, “Neural discrete representation learning,” in *Proc. NIPS*, 2017, pp. 6306–6315.
- [23] A. van den 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,” in *9th ISCA Speech Synthesis Workshop*, pp. 125–125.
- [24] 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,” in *Proc. Interspeech 2019*, 2019, pp. 1103–1107.
- [25] 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.
- [26] L. Kaiser, S. Bengio, A. Roy, A. Vaswani, N. Parmar, J. Uszkoreit, and N. Shazeer, “Fast decoding in sequence models using discrete latent variables,” in *Proceedings of ICML*, 2018, pp. 2390–2399.
- [27] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” *arXiv preprint arXiv:1607.08022*, 2016.
- [28] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of ICML*, 2010.
- [29] X. Huang and S. Bengio, “Arbitrary style transfer in real-time with adaptive instance normalization,” in *Proceedings of ICCV*, 2017, pp. 1501–1510.
- [30] J. chieh Chou and H.-Y. Lee, “One-Shot Voice Conversion by Separating Speaker and Content Representations with Instance Normalization,” in *Proc. Interspeech 2019*, 2019, pp. 664–668.
- [31] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [32] Y. Bengio, N. Léonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” *arXiv preprint arXiv:1308.3432*, 2013.
- [33] S. Sakti, R. Maia, S. Sakai, T. Shimizu, and S. Nakamura, “Development of hmm-based indonesian speech synthesis,” in *Proc. Oriental COCODSA*, vol. 1, 2008.
- [34] 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 *TCAST 2008*, 2008.
- [35] T. Schatz, V. Peddinti, F. Bach, A. Jansen, H. Hermansky, and E. Dupoux, “Evaluating speech features with the minimal-pair abx task: Analysis of the classical mfc/plp pipeline,” 2013.
- [36] 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., 2019.
- [37] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [38] Z. Wu, O. Watts, and S. King, “Merlin: An open source neural network speech synthesis system,” in *SSW*, 2016, pp. 202–207.
- [39] B. van Niekerk, L. Nortje, and H. Kamper, “Vector-quantized neural networks for acoustic unit discovery in the ZeroSpeech 2020 challenge,” in *Submitted to Interspeech*, 2020.