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

Andros Tjandra¹, Sakriani Sakti¹,², Satoshi Nakamura¹,²

Nara Institute of Science and Technology, Japan
RIKEN AIP, Japan

© 2020 Andros Tjandra
Outline

• Background
• Unsupervised subword discovery
  • VQVAE model & objective
  • Self-attention & Transformers
  • Model regularization
• Codebook inverter
• Experiment
• Conclusion
Background

• The ZeroSpeech 2020 - Track 2019 challenge confronts the problem of constructing a speech synthesizer without any text or phonetic labels: TTS without T.

• Two objectives:
  • Find related contexts from the speech and encodes them as efficient as possible (low-bitrate).
  • Using the encoded representation, reconstruct the content back to the speech into another speaker voice.
Vector Quantized Variational Autoencoder (VQVAE)

- VQ-VAE has three main modules:
  - Encoder $q_\theta(z|x)$ read speech features $x \in \mathbb{R}^D$ and output $z \in \{1..K\}$
  - Codebook $E = [e_1, \ldots, e_K] \in \mathbb{R}^{K \times D_e}$
  - Decoder $p_\phi(x|z, s)$ reconstruct the speech features conditioned by codebook $z$ and speaker ID $s$

- Using explicit speaker information for the decoder, encoder and codebook only need to model the speech context without capturing the speaking style (disentangled with speaker).

© 2020 Andros Tjandra
Training Objective and Inference

• The discretization process in the encoder:

\[ q_\theta(z = c | x) = \begin{cases} 
1 & \text{if } c = \text{argmin}_i \| \hat{z} - e_i \|_2 \\
0 & \text{else}
\end{cases} \]

\[ e_c = \sum_{i=1}^{K} q_\theta(z = i | x) e_i. \]

• Training objective:

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

Reconstruction loss \hspace{1cm} \text{Consistency loss}
Self Attention and Transformer

Dot Product Attention

Multihead Self Attention

Transformer module

Image reference: Attention is all you need [Vaswani et al., NIPS 2017]

© 2020 Andros Tjandra
VQVAE Encoder & Decoder blocks

a) Encoder: 2x Transformer layer + 1D convolution (with stride to downsample input) + batchnorm

b) Decoder: 1D Convolution + Upsample operation + batchnorm
Model regularization

• Sometimes, generative model could also suffer from overfitting especially when the amount of data is small.

• We deploy several regularization technique to improve the performance:
  1. Temporal smoothing
  2. Temporal jitter
Temporal Smoothing

• Since the encoder captures sequential data, we introduced temporal smoothing between two consecutive time-steps:

\[ L_{reg} = \sum_{i=1}^{T-1} \left\| z_t - z_{t+1} \right\|^2_2. \]

• Final loss:

\[ L = L_{VQ} + \lambda L_{reg}, \]
Temporal Jitter

• Temporal jitter regularization [1] is used to prevent the latent vector co-adaptation and to reduce the model sensitivity near the unit boundary.

\[ j_t \sim \text{Categorical}(p, p, 1 - 2 \times p) \in \{1, 2, 3\} \]

\[
\hat{c}_t = \begin{cases} 
  c_{t-1}, & \text{if } j_t = 1 \text{ and } t > 1 \\
  c_{t+1}, & \text{if } j_t = 2 \text{ and } t < T \\
  c_t, & \text{else}
\end{cases}
\]

\[ e_t = \mathbb{E}[\hat{c}_t] \]

[1] Unsupervised speech representation learning using Wavenet autoencoders, [Chorowski et al., 2019]
Codebook Inverter

- Codebook inverter predicts the linear spectrogram given the predicted codebook from VQVAE.
- We use Griffin-Lim algorithm to generate the speech waveform.
- Loss: \( L^S = \| X^R - \hat{X}^R \|_2 \)
Experimental Setup

• Log mel-spectrogram (80 dims) vs MFCC (39 dims with delta & delta-delta)

<table>
<thead>
<tr>
<th>Model</th>
<th>ABX</th>
<th>Bitrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrfVQVAE with log-Mel</td>
<td>33.79</td>
<td>171.05</td>
</tr>
<tr>
<td>TrfVQVAE with MFCC</td>
<td>21.91</td>
<td>170.42</td>
</tr>
</tbody>
</table>

• For the rest of experiments, we will use MFCC features.
Conv VQVAE vs Transformer VQVAE

<table>
<thead>
<tr>
<th>Model</th>
<th>ABX</th>
<th>Bitrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv VQVAE (stride $4\times$, $K=256$) [1]</td>
<td>24.17</td>
<td>184.32</td>
</tr>
<tr>
<td>TrfVQVAE (stride $4\times$, $K=64$)</td>
<td>22.72</td>
<td>141.82</td>
</tr>
<tr>
<td>TrfVQVAE (stride $4\times$, $K=128$)</td>
<td>21.91</td>
<td>170.42</td>
</tr>
<tr>
<td>TrfVQVAE (stride $4\times$, $K=256$)</td>
<td>21.94</td>
<td>194.69</td>
</tr>
<tr>
<td>TrfVQVAE (stride $4\times$, $K=512$)</td>
<td>21.6</td>
<td>217.47</td>
</tr>
</tbody>
</table>

## Best model + regularization

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

© 2020 Andros Tjandra
Conclusions

• We achieved significant improvement with Transformer VQ-VAE (-2.2 ABX vs Conv VQ-VAE).

• By adding regularization in the VQVAE encoder and codebook, we get further improvement (up to -1.77 ABX vs un-regularized model.)
The end.