Unsupervised acoustic unit discovery for speech synthesis using discrete latent-variable neural networks

Interspeech 2019, Graz, Austria

Ryan Eloff, André Nortje, Benjamin van Niekerk, Avashna Govender, Leanne Nortje, Arnu Pretorius, Elan van Biljon, Ewald van der Westhuizen, Lisa van Staden, Herman Kamper

Stellenbosch University, South Africa & University of Edinburgh, UK

https://github.com/kamperh/suzerospeech2019
Advances in speech recognition

- Addiction to text: 2000 hours transcribed speech audio; ∼350M/560M words text [Xiong et al., TASLP'17]
- Sometimes not possible, e.g., for unwritten languages
- Very different from the way human infants learn language
Advances in speech recognition

- **Addiction to text**: 2000 hours transcribed speech audio; 
  ~350M/560M words text [Xiong et al., TASLP’17]

---

1 / 14
Advances in speech recognition

- **Addiction to text**: 2000 hours transcribed speech audio; \(\sim 350M/560M\) words text [Xiong et al., TASLP’17]

- Sometimes not possible, e.g., for unwritten languages
Advances in speech recognition

- **Addiction to text**: 2000 hours transcribed speech audio; \( \sim 350M/560M \) words text [Xiong et al., TASLP’17]
- Sometimes not possible, e.g., for unwritten languages
- Very different from the way human infants learn language
Zero-Resource Speech Challenges (ZRSC)
Zero-Resource Speech Challenges (ZRSC)
ZRSC 2019: Text-to-speech without text

Waveform generator

Target voice

‘the dog ate the ball’
ZRSC 2019: Text-to-speech without text

Waveform generator

Acoustic model

Target voice

the dog ate the ball
What do we get for training?
What do we get for training?

No labels
What do we get for training?

No labels :)

Figure adapted from: http://zerospeech.com/2019
What do we get for training?

No labels :)
Approach: Compress, decode and synthesise
Approach: Compress, decode and synthesise

Symbol-to-speech module

Compression model

Encoder
Discretise
FFTNet
Decoder
z_1:N
h_1:N
x_1:T
\hat{y}_{1:T}

Filterbanks

Vocoder

Training speaker

Embed

MFCCs

Waveform
Approach: Compress, decode and synthesise

- Waveform
- FFTNet
- Vocoder
- Filterbanks
- Decoder
- Embed
- Target speaker
- MFCCs
- Speech synthesis
- Compression model
- Symbol-to-speech module
- Speaker ID
Approach: Compress, decode and synthesise

Encoder
Discretise
FFTNet
Decoder
z_{1:N}
h_{1:N}
x_{1:T}
\hat{y}_{1:T}
MFCCs
Filterbanks
Waveform
Vocoder
Symbol-to-speech module
Speaker ID
Embed
Approach: Compress, decode and synthesise

Encoder

Discretise

FFTNet

Decoder

z_{1:N}

h_{1:N}

x_{1:T}

\hat{y}_{1:T}

MFCCs

Filterbanks

Vocoder

Symbol-to-speech module

Speaker ID

Embed

Compression model

Waveform

Training speaker

Target speaker

Speaker ID

Embed
Discretisation methods

- Straight-through estimation (STE) binarisation:

- Categorical variational autoencoder (CatVAE):

- Vector-quantised variational autoencoder (VQ-VAE):
Neural network architectures

- Encoder: Convolutional layers, each layer with a stride of 2
- Decoder: Transposed convolutions mirroring encoder
- Waveform generation: FFTNet autoregressive vocoder
- Also experimented with WaveNet: Sometimes gave noisy output
- Bitrate: Set by number of symbols $K$ and number of striding layers
Evaluation

Human evaluation metrics:

- Mean opinion score (MOS)
- Character error rate (CER)
- Similarity to the target speaker’s voice
Evaluation

Human evaluation metrics:

• Mean opinion score (MOS)
• Character error rate (CER)
• Similarity to the target speaker’s voice

Objective evaluation metrics:

• ABX discrimination
• Bitrate
Evaluation

Human evaluation metrics:

- Mean opinion score (MOS)
- Character error rate (CER)
- Similarity to the target speaker’s voice

Objective evaluation metrics:

- ABX discrimination
- Bitrate

Two evaluation languages:

- English: Used for development
- Indonesian: Held out “surprise language”
ABX on English with speaker conditioning

![Bar chart showing ABX results for STE, VQ-VAE, and CatVAE models with and without speaker conditioning.](chart.png)
ABX on English for different compression rates

![ABX bar chart]

- **STE**
  - 64: 116
  - 256: 473
  - 512: 79

- **VQ-VAE**
  - 64: 154
  - 256: 644
  - 512: 85

- **CatVAE**
  - 64: 93
  - 256: 188
  - 512: 770

**Comparison:**
- No downsampling
- ×4 downsample
- ×8 downsample
ABX on English for different compression rates

![Bar chart showing ABX results for different compression rates and downsampling factors.](chart.png)
ABX on English for different compression rates

![Bar chart showing ABX performance across different compression rates and downsampling factors for STE, VQ-VAE, and CatVAE models.](chart.png)
ABX on English for different compression rates

The graph compares ABX scores for different compression rates on ST Enhanced (STE), VQ-VAE, and CatVAE models. The scores are indicated for three downsampling levels: no downsampling, ×4 downsample, and ×8 downsample. The ABX (%) is shown for each compression rate across the three models.
## Official evaluation results

<table>
<thead>
<tr>
<th>Model</th>
<th>CER (%)</th>
<th>MOS [1, 5]</th>
<th>Similarity [1, 5]</th>
<th>ABX (%)</th>
<th>Bitrate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>English:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPGMM-Merlin</td>
<td>75</td>
<td>2.50</td>
<td>2.97</td>
<td>35.6</td>
<td>72</td>
</tr>
<tr>
<td>VQ-VAE-x8</td>
<td>75</td>
<td>2.31</td>
<td>2.49</td>
<td>25.1</td>
<td>88</td>
</tr>
<tr>
<td>VQ-VAE-x4</td>
<td>67</td>
<td>2.18</td>
<td>2.51</td>
<td>23.0</td>
<td>173</td>
</tr>
<tr>
<td>Supervised</td>
<td>44</td>
<td>2.77</td>
<td>2.99</td>
<td>29.9</td>
<td>38</td>
</tr>
</tbody>
</table>

| **Indonesian:**        |         |            |                   |         |         |
| DPGMM-Merlin           | 62      | 2.07       | 3.41              | 27.5    | 75      |
| VQ-VAE-x8              | 58      | 1.94       | 1.95              | 17.6    | 69      |
| VQ-VAE-x4              | 60      | 1.96       | 1.76              | 14.5    | 140     |
| Supervised             | 28      | 3.92       | 3.95              | 16.1    | 35      |
## Synthesised examples

<table>
<thead>
<tr>
<th>Model</th>
<th>Input</th>
<th>Synthesised output</th>
<th>Target speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>English:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VQ-VAE-×4</td>
<td>Play</td>
<td>Play</td>
<td>Play</td>
</tr>
<tr>
<td>VQ-VAE-×4-new</td>
<td>Play</td>
<td>Play</td>
<td>Play</td>
</tr>
<tr>
<td>VQ-VAE-×4</td>
<td>Play</td>
<td>Play</td>
<td>Play</td>
</tr>
<tr>
<td>VQ-VAE-×4-new</td>
<td>Play</td>
<td>Play</td>
<td>Play</td>
</tr>
<tr>
<td><strong>Indonesian:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VQ-VAE-×4</td>
<td>Play</td>
<td>Play</td>
<td>Play</td>
</tr>
<tr>
<td>VQ-VAE-×4-new</td>
<td>Play</td>
<td>Play</td>
<td>Play</td>
</tr>
<tr>
<td>VQ-VAE-×4</td>
<td>Play</td>
<td>Play</td>
<td>Play</td>
</tr>
<tr>
<td>VQ-VAE-×4-new</td>
<td>Play</td>
<td>Play</td>
<td>Play</td>
</tr>
</tbody>
</table>
Conclusions

• Speaker conditioning consistently improves performance

• Different discretisation methods are similar (VQ-VAE slightly better)

• Different models difficult to compare because of bitrate

• Future: Does discretisation actually benefit feature learning?
Why do we have ten authors on this paper?

Ryan Eloff
André Nortje
Benjamin van Niekerk
Avashna Govender
Leanne Nortje

Arnu Pretorius
Elan van Biljon
Ewald van der Westhuizen
Lisa van Staden
Herman Kamper
https://github.com/kamperh/suzerospeech2019

(Update coming soon)
Straight-through estimation (STE) binarisation

- STE binarisation:
  \[ z_k = 1 \text{ if } h_k \geq 0 \text{ or } z_k = -1 \text{ otherwise} \]

- For backpropagation we need:
  \[
  \frac{\partial J}{\partial h} = \frac{\partial z}{\partial h} \cdot \frac{\partial J}{\partial z}
  \]

- For single element:
  \[
  \frac{\partial J}{\partial h_k} = \frac{\partial z_k}{\partial h_k} \cdot \frac{\partial J}{\partial z_k}
  \]

- What is \( \frac{\partial z_k}{\partial h_k} \) with \( z_k = \text{threshold}(h_k) \)? Cannot solve directly

- Idea: If \( z_k \approx h_k \) then we could use \( \frac{\partial J}{\partial h_k} \approx \frac{\partial J}{\partial z_k} \)
As an example, let us say $h_k = 0.7$: 

\begin{figure}
\centering
\includegraphics[width=\textwidth]{example}
\end{figure}
Instead of direct thresholding, let us set $z_k = 1$ with probability 0.85 and $z_k = -1$ with probability 0.15:

Estimated mean of $z_k$ over 500 samples: 0.668
Straight-through estimation (STE) binarisation

- So, instead of direct thresholding, we set $z_k = h_k + \epsilon$, where $\epsilon$ is sampled noise:

$$\epsilon = \begin{cases} 
1 - h_k & \text{with probability } \frac{1+h_k}{2} \\
-h_k - 1 & \text{with probability } \frac{1-h_k}{2}
\end{cases}$$

- Since $\epsilon$ is zero-mean, the derivative of the expected value of $z_k$ is:

$$\frac{\partial \mathbb{E}[z_k]}{\partial h_k} = 1$$

- Therefore, gradients are passed unchanged through the thresholding operation:

$$\frac{\partial J}{\partial h} \approx \frac{\partial J}{\partial z}$$