Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation

Hao-Wen Dong and Yi-Hsuan Yang
Research Center of IT Innovation, Academia Sinica
Outlines

• Introduction
• Binary Neurons
• Proposed Model
• Data
• Results
• Future Works

Source Code  https://github.com/salu133445/bmusegan
Demo Page  https://salu133445.github.io/bmusegan/
Introduction
Introduction

- **MuseGAN**
  - can only generate **real-valued predictions**
  - require **postprocessing at test time**
    (e.g., hard thresholding or Bernoulli sampling)
Introduction

- Naïve binarization methods can lead to **overly-fragmented notes**

<table>
<thead>
<tr>
<th></th>
<th>raw</th>
<th>Bernoulli sampling</th>
<th>hard thresholding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>![raw image]</td>
<td>![Bernoulli sampling image]</td>
<td>![hard thresholding image]</td>
</tr>
</tbody>
</table>
Introduction

- Real-valued predictions can lead to **training difficulties of the discriminator**

Decision boundaries to learn for the **discriminator**

- **real** samples
- **fake** samples
- **decision boundaries**

- **real values**
- **binary values**
Binary Neurons
Binary Neurons

• Neurons that output binary-valued predictions
• In this work, we consider
  ◦ deterministic binary neurons (DBNs)
    \[
    DBN(x) = \begin{cases} 
      1, & \text{if } \sigma(x) > 0.5 \\
      0, & \text{otherwise}
    \end{cases}
    \]
  ◦ stochastic binary neurons (SBNs)
    \[
    SBN(x) = \begin{cases} 
      1, & \text{if } z < \sigma(x) \\
      0, & \text{otherwise}
    \end{cases}, \quad z \sim U[0, 1]
    \]
Gradient Estimators

• Computing the exact gradients for binary neurons is intractable
• **Straight-through (ST) estimator**
  ◦ treat BNs as *identity functions* in the backward pass
    \[
    \frac{\partial BN(x)}{\partial x} = 1
    \]
• **Sigmoid-adjusted ST estimator**
  ◦ treat BNs as *identity functions multiplied by the derivative of the sigmoid function* in the backward pass
    \[
    \frac{\partial BN(x)}{\partial x} = \sigma(x)
    \]
Proposed Model
Generative Adversarial Networks

**Generator**
Make $G(z)$ indistinguishable from real data for $D$

**Discriminator**
Tell $G(z)$ as fake data from $X$ being real ones

random noise: $z \sim p_z$

fake samples: $G(z)$

real samples: $X \sim p_X$

1/0
Generator

• One single input random vector

• **Shared/private design**
  ◦ Different tracks have their own musical properties (e.g. textures, patterns, techniques)
  ◦ Jointly all tracks follow a common, high-level musical idea
Refiner

- Refine the real-valued outputs of the generator into binary ones
- Composed of a number of *residual units*
Refiner

- Refine the real-valued outputs of the generator into binary ones
- Composed of a number of *residual units*
Discriminator

- **Shared/private design** (similar to the generator)
- Additional **onset/offset stream** and **chroma stream**
Two-stage Training

- First stage — pretrain the **generator** and **discriminator**
- Second stage — train the **refiner** and **discriminator** (with $G$ fixed)
Data Representation

- Multi-track piano-roll
- 8 tracks
  - Drums, Piano, Guitar, Bass, Ensemble, Reed, Synth Lead and Synth Pad

A $4 \times 96 \times 84 \times 8$ tensor
Training Data

- **Lakh Pianoroll Dataset** (LPD)
- **13746 four-bar phrases** from 2291 songs (six for each)
  - Pick only songs in 4/4 time and with an *alternative* tag
Results
### Qualitative Comparison

<table>
<thead>
<tr>
<th></th>
<th>raw</th>
<th>pretrained (+BS)</th>
<th>pretrained (+HT)</th>
<th>proposed (+SBNs)</th>
<th>proposed (+DBNs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>raw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pretrained (+BS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pretrained (+HT)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proposed (+SBNs)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proposed (+DBNs)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Audio Samples

- **proposed (+DBNs)** — fewer overly-fragmented notes; more out-of-scale notes
- **proposed (+SBNs)** — more overly-fragmented notes; lots of artifacts

More samples available on demo page [https://salu133445.github.io/bmusegan/]
Evaluation Metrics

• Qualified note rate (QN)

\[ QN = \frac{\text{# of notes no shorter than 3 time steps (i.e., a 32th note)}}{\text{# of notes}} \]

• Polyphonicity (PP)

\[ PP = \frac{\text{# of time steps where more than two pitches are played}}{\text{# of time steps}} \]

• Tonal distance (TD)
  ◦ measure the distance between two chroma features in a tonal space
Comparisons of Training Strategies

- **Two-stage training** (*proposed*) — [stage 1] pretrain G and D  [stage 2] train R and D
- **Joint training** (*joint*) — [stage 1] pretrain G and D  [stage 2] train G, R and D
- **End-to-end training** (*end-to-end*) — train G, R and D in one stage

<table>
<thead>
<tr>
<th>Training Data</th>
<th>Pretrained BS</th>
<th>Pretrained HT</th>
<th>Proposed SBNs</th>
<th>Proposed DBNs</th>
<th>Joint SBNs</th>
<th>Joint DBNs</th>
<th>End-to-end SBNs</th>
<th>End-to-end DBNs</th>
</tr>
</thead>
<tbody>
<tr>
<td>QN</td>
<td>0.88</td>
<td>0.67</td>
<td>0.42</td>
<td><strong>0.78</strong></td>
<td>0.18</td>
<td>0.55</td>
<td><strong>0.67</strong></td>
<td>0.28</td>
</tr>
<tr>
<td>PP</td>
<td>0.48</td>
<td>0.20</td>
<td>0.26</td>
<td><strong>0.45</strong></td>
<td>0.19</td>
<td>0.19</td>
<td>0.16</td>
<td><strong>0.29</strong></td>
</tr>
<tr>
<td>TD</td>
<td>0.96</td>
<td><strong>0.98</strong></td>
<td><strong>0.99</strong></td>
<td>0.87</td>
<td><strong>0.95</strong></td>
<td>1.00</td>
<td>1.40</td>
<td>1.10</td>
</tr>
</tbody>
</table>

(values closer to that of the training data is better; **underline**: closest; **bold**: top 3 closest)
Comparisons of Training Strategies
Comparisons of Training Strategies

QN

PP
End-to-end Models

• First attempt, to our best knowledge, to **generate such high-dimensional data with binary neurons from scratch**
End-to-end Models

• First attempt, to our best knowledge, to generate such high-dimensional data with binary neurons from scratch.
Effects of the Discriminator Design

- **pretrained** — *shared/private design + offset/onset stream + chroma stream*
- **ablated** — *shared/private design*
- **baseline** — *only one shared discriminator*

<table>
<thead>
<tr>
<th>training data</th>
<th>pretrained</th>
<th>ablated</th>
<th>baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BS</td>
<td>HT</td>
<td>BS</td>
</tr>
<tr>
<td>QN</td>
<td>0.88</td>
<td>0.67</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>0.72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PP</td>
<td>0.48</td>
<td>0.20</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>0.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TD</td>
<td>0.96</td>
<td>0.98</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Future Works
Summary

• A convolutional GAN for **binary-valued multi-track piano-rolls**
  ◦ **CNNs + residual units + binary neurons**
  ◦ **Shared/private design** in both the generator and discriminator (proved effective)
  ◦ **Onset/offset** and **chroma streams** in the discriminator (proved effective)
  ◦ **Two-stage training** (proved effective)

• **Proposed model** with **deterministic binary neurons (DBNs)** features fewer overly-fragmented notes as compared with existing methods.
Future Works

• **Tradeoff**
  ◦ *easy-to-train* generator + *hard-to-train* discriminator
  ◦ *hard-to-train* generator + *easy-to-train* discriminator

• **Longer music**
  ◦ RNNs (LSTMs or GRUs)
  ◦ How to generate high-level/long-term structure?

• **More tracks**
  ◦ Symphony/orchestra compositions
  ◦ (hierarchical) sections → sub-sections → instruments
Thank you for your attention