C-RNN-GAN: Continuous recurrent neural networks with adversarial training

Author: Olof Mogren
Presented By: Prakhar Pandey
Table of Content

● Discriminative vs Generative models: GANs

● Music Generation
  ○ Music Representation
  ○ Previous Work
  ○ Problem with Discrete representation

● Model by Mogren
  ○ Music Representation
  ○ Advantage of this Representation
  ○ Model
  ○ Experimental Setup
  ○ Training
  ○ C-RNN-GAN-3
  ○ Results

● Criticism
**Discriminative vs Generative**

**Generative Models:**

Generative models learn the intrinsic distribution function of the input data $p(x)$ (or $p(x,y)$ if there are multiple targets/classes in the dataset), allowing them to generate both synthetic inputs $x'$ and outputs/targets $y'$, typically given some hidden parameters.

**Discriminative Models:**

Discriminative models learn how to model the conditional probability distribution function (p.d.f) $p(y|x)$ instead

GANs

Training set

Random noise

Generator

Fake image

Discriminator

Real
Fake

~Ref:[https://skymind.ai/wiki/generative-adversarial-network-gan]
GANs: Generator vs Discriminator

Generator:

\[ L_G = \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)}))) \]

Discriminator:

\[ L_D = \frac{1}{m} \sum_{i=1}^{m} \left[ - \log D(x^{(i)}) - \log(1 - D(G(z^{(i)}))) \right] \]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D )</td>
<td>Discriminator function</td>
</tr>
<tr>
<td>( G )</td>
<td>Generator function</td>
</tr>
<tr>
<td>( z )</td>
<td>Noise or Random vector</td>
</tr>
<tr>
<td>( x )</td>
<td>Real sample</td>
</tr>
</tbody>
</table>
Music Generation
Symbolic music representations comprise any kind of score representation with an explicit encoding of notes or other musical events.

MIDI (Musical Instrument Digital Interface): MIDI messages encode information for each note event such as the note onset, note offset, and intensity (represented as "velocity" in MIDI terminology).

Example Fields:
- The **MIDI note number** is an integer between 0 and 127 that encodes the note's pitch. Most importantly, C4 (middle C) has MIDI note number 60, and A4 (concert A440) has MIDI note number 69.
- The **key velocity** is an integer between 0 and 127 which controls the intensity of the sound.

~Ref: [https://musicinformationretrieval.com/symbolic_representations.html]
Previous Work

- RNNs to model music. But they all used symbolic or discrete representation.

- Work using RNNs for music generation includes:
  - [Eck and Schmidhuber, 2002], modelling blues songs with 25 discrete tone values.
  - [Nicolas Boulanger-Lewandowski, 2012], combining the RNN with restricted Boltzmann machines, representing 88 distinct tones.
  - [Yu et al. 2016] trained an RNN with adversarial training, applying policy gradient methods to cope with the discrete nature of the symbolic representation they employed.
So what’s the problem with discrete representation?
So what’s the problem with discrete representation?

Hint: Backpropagation

Sampling process from Generator is non-differentiable
Model by Mogren:
Music Representation

Their work represents tones using real valued continuous quadruplets of frequency, length, intensity, and timing:

\[[\text{tone length}, \text{frequency}, \text{intensity}, \text{time}]\]

**Tone length**: How long the tone lasts

**Frequency**: Pitch

**Intensity**: Loudness or amplitude

**Time**: time spent since the previous tone
What is the advantage of this representation?
What is the advantage of this representation?

Modelling the data in this way allows the network to represent polyphonous chords (with zero time between two tones)
Model

- Generator: Uni-directional RNN
- Discriminator: Bidirectional RNN
- Fully connected layer after D share weights across time steps
- One sigmoid output per cell is then averaged to the final decision for the sequence.
Experimental Setup

**Generator and Discriminator:**
- The LSTM network in both G and D with depth 2
- Each LSTM cell has 350 internal (hidden) units.

**Baseline:**
- RNN similar to our generator, but trained entirely to predict the next tone event at each point in the recurrence.

**Dataset:**
- Classical Music collected in MIDI format converted into the format discussed.
- 3697 midi files from 160 different composers of classical music
Training

General:

- Used Mini-batch Stochastic Gradient
- Used L2-regularization on both Generator’s and Discriminator’s weights

Pre-training:

- The generator was pre-trained for 6 epochs with a squared error loss for predicting the next event in the training sequence
  - **Curriculum Learning:**
    - Begin pre-training with short sequences, eventually pre-train the model with increasingly long sequences
    - This helps the model first learn relations between points that are closer in time and then learn longer dependencies
Training

**Freezing:**
- D can become too strong, resulting in a gradient that cannot be used to improve G or vice-versa
- This effect is particularly clear when the network is initialized without pretraining
- Freezing means stopping the updates of one network (D or G) whenever its training loss is less than 70% of the training loss of other network (G or D).

**Feature Matching:**
- This approach encourages greater variance in generated music
- Avoid overfitting to the current discriminator
- Generator’s objective is instead to produce an internal representation at some level in the discriminator that matches that of real data
- Author chooses the representations R from the last layer before the final logistic classification layer in D

\[
\hat{L}_G = \frac{1}{m} \sum_{i=1}^{m} (R(x^{(i)}) - R(G(z^{(i)})))^2
\]
C-RNN-GAN-3

To evaluate the effect on polyphony by changing the model, author also experimented with having up to three tones represented as output from each LSTM cell in G (with corresponding modifications to D). Each tone is then represented with its own quadruplet of values as described above.
Results

Author introduces some metrics to measure the quality of the results example:

- **Polyphony**, measuring how often (at least) two tones are played simultaneously
- **Scale consistency** were computed by counting the fraction of tones that were part of a standard scale
- **Repetitions** of short subsequences were counted, giving a score on how much recurrence there is in a sample.
- **Tone span** is the number of half-tone steps between the lowest and the highest tone in a sample.
(a) **C-RNN-GAN** using feature matching.

(b) **Baseline** with maximum likelihood training.

(c) **C-RNN-GAN-3**, using feature matching and three tone outputs per LSTM cell.

(d) The same statistics for a selection of real music in the dataset.
Results

http://mogren.one/publications/2016/c-rnn-gan/
Critiques

- Easy to understand and simple idea which can be extended to other continuous datasets eg: (Cleaning Noisy audio files)

- The focus in the result section seems to be on producing variance in the generated music while that is one of the important factors but it shouldn’t be the only one. It is unclear as to how once can evaluate a generated music.

- They didn’t use different continuous representation for music they stuck with the quadruplet representation maybe introducing new fields or removing few fields might have helped. (eg: tempo, time-offsets, etc)
References

