Computational Creativity

Music Generation with AI

Computational creativity

The philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative. [3]

State of the art

- Algorithmic composition
- Computer Aided Music Generation
- **Automated stand-alone music generation <- focus of this presentation**

Algorithmic composition

- Use an algorithm to generate music
- Tools like Nyquist (Lisp based programming language)
- Creates modern computer music
- Earliest known recording made by Turing (1951)
  - “Not very interested in programming the computer to play conventional pieces of music”

Nyquist

```
set pitch-cycle = make-cycle(list(c4, c6,
nil))
set vel-cycle = make-cycle({75 100 125})
set dur-cycle = make-cycle({0.3 0.5 0.7})
exec score-gen(save: quote(item-streams),
 score-len: 10,
 pitch: next(pitch-cycle),
 vel: next(vel-cycle),
 ioi: next(dur-cycle))
```

```
SAL> exec score-print(item-streams)
((0 0 (SCORE-BEGIN-END 0 NIL))
(0 0.3 (NOTE vel: 75 pitch: 60))
(0.3 0.5 (NOTE vel: 100 pitch: 84))
(0.8 0.7 (NOTE vel: 125 pitch: NIL))
(1.5 0.3 (NOTE vel: 75 pitch: 60))
(1.8 0.5 (NOTE vel: 100 pitch: 84))
(2.3 0.7 (NOTE vel: 125 pitch: NIL))
(3 0.3 (NOTE vel: 75 pitch: 60))
(3.3 0.5 (NOTE vel: 100 pitch: 84))
(3.8 0.7 (NOTE vel: 125 pitch: NIL))
(4.5 0.3 (NOTE vel: 75 pitch: 60)) )
```

(note-offset duration (NOTE vel: volume pitch: note-pitch))

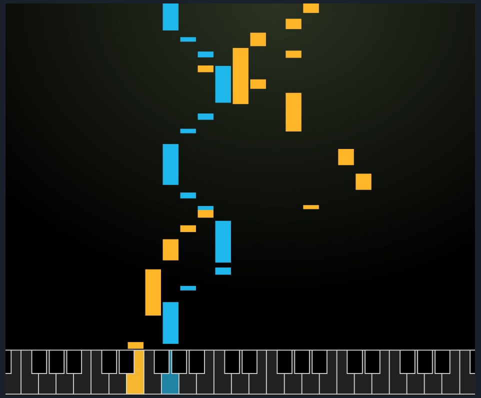
Computer aided music generation

- Computer generates music starting from an initial manually composed input
- Can harmonize melodies
- Can generate melody on top of chords

Very good results available

Magenta

- Research project from Google
- Based on Tensorflow
- Multiple extensions (most based on RNN)
- Examples:
    - AI Duet
    - Melody autocompletion



FlowMachines

- Uses Markov chains to identify patterns [7]
- Generates possible melodies and chords
- Human judgement (selecting the good parts)
- Extremely good results (first AI generated pop album)
- But 90% human work, 10% AI

Markov chains

- Popular technique in music generation
- “Predecessor of deep learning”
- Satisfy markov property
- Challenge: balance markov order: Too low → No musical phrases
- Too high → plagiarism

$$Pr(X_n = x_n \mid X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \ldots, X_1 = x_1)$$
$$= Pr(X_n = x_n \mid X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \ldots, X_{n-m} = x_{n-m}) \; for \; n > m$$

Automated stand-alone music generation
- Idea

- Computer ideally learns patterns from dataset
- Computer generates entire piece of music from scratch
- Can compose in the style of certain composers or genres by choosing datasets

Database -> MIDI

- MIDI = Musical Instrument Digital Interface
- Binary format for specifying music events
- Communication protocol
- Useful for us: NoteOn / NoteOff (pitch) (velocity)
- Use libraries to read midi

Midi → Music21

```
<music21.note.Note B> 72.0
<music21.chord.Chord E3 A3> 72.0
<music21.note.Note A> 72.5
<music21.chord.Chord E3 A3> 72.5
<music21.note.Note E> 73.0
<music21.chord.Chord E3 A3> 73.0
<music21.chord.Chord E3 A3> 73.5
<music21.note.Note E-> 74.0
```

Music21 -> Integer streams

- MIDI format does not perform well in neural networks
- Transform into single integer stream

```
<music21.note.Note B> 72.0
<music21.chord.Chord E3 A3> 72.0
<music21.note.Note A> 72.5
<music21.chord.Chord E3 A3> 72.5
<music21.note.Note E> 73.0
<music21.chord.Chord E3 A3> 73.0
<music21.chord.Chord E3 A3> 73.5
<music21.note.Note E-> 74.0
```

→ 1 2 3 2 4 2 2 5

Understanding music

“Generating long pieces of music is a challenging problem, as music contains structure at multiple timescales, from milisecond timings to motifs to phrases to repetition of entire sections.” [10]

- Markov chains → troubles with markov order
- Recurrent neural nets → troubles with vanishing or exploding gradients
- LSTM’s → Works great for long-term dependencies

Explored Methods

- Generative Adversarial Networks (GAN)
- Restricted Boltzmann Machines
- LSTM based approach

Database - augmentation

Common techniques are:

- all songs in same key
- augment songs to different keys

  => learn patterns and relative note positions instead of notes

- all songs in same tempo

Generative Adversarial Network (GAN)

- Two different roles:
  - Generator

    ⇒ Generate data that looks like train data

  - Discriminator

    ⇒ Recognize train data

- Result: not very impressive [9]

Generative Adversarial Network (GAN)

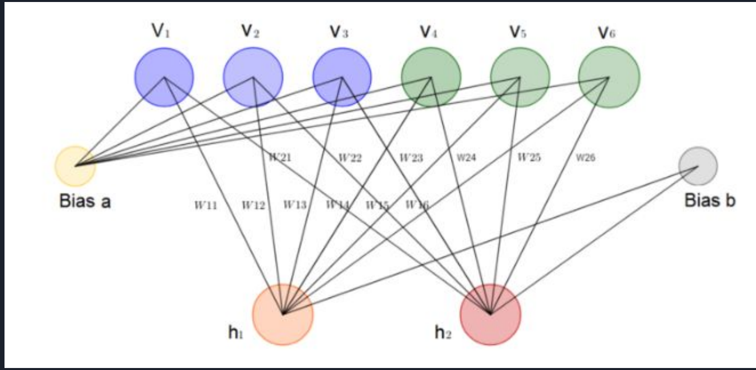
- Result: not very impressive [9]

⇒ Training is very unstable

⇒ Hard for discriminator to define ‘good’ music

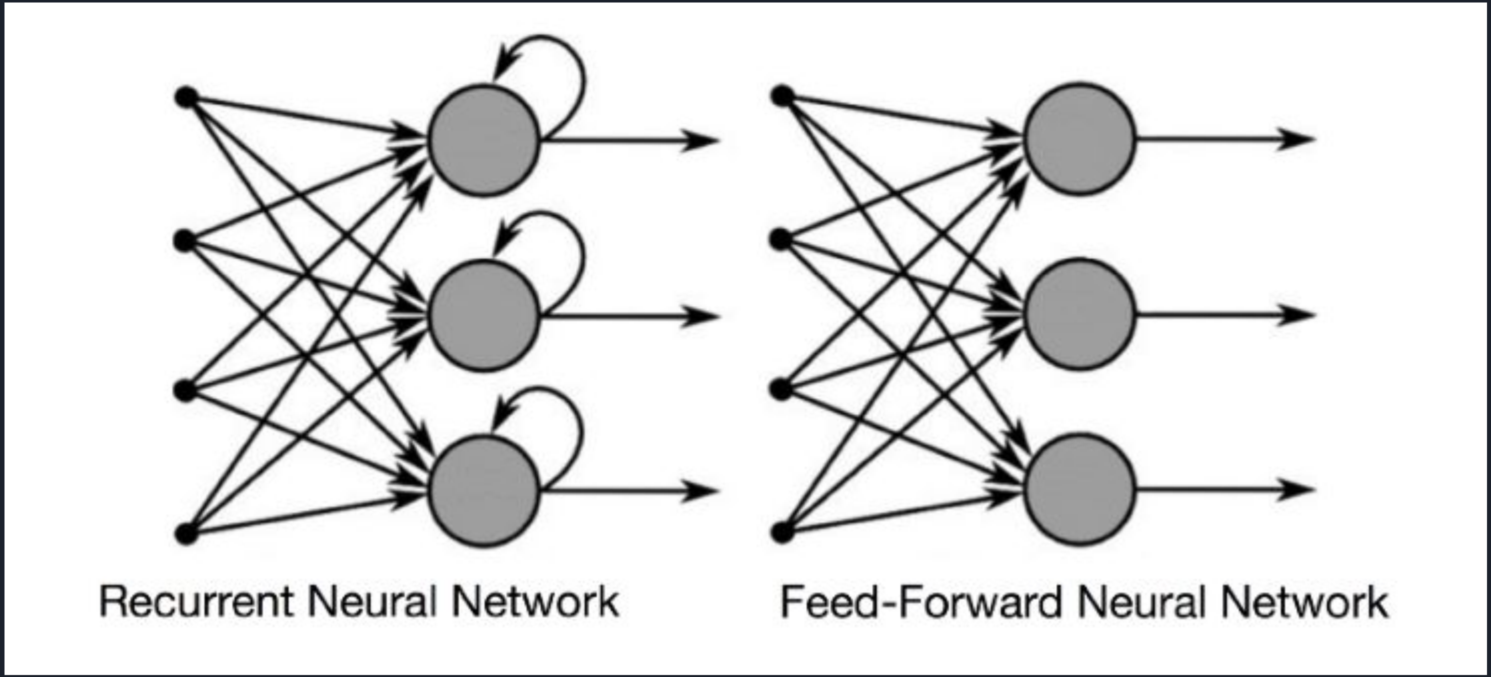
Restricted Boltzmann Machines

- Two layered Neural Net
  - Input layer
  - Hidden Layer



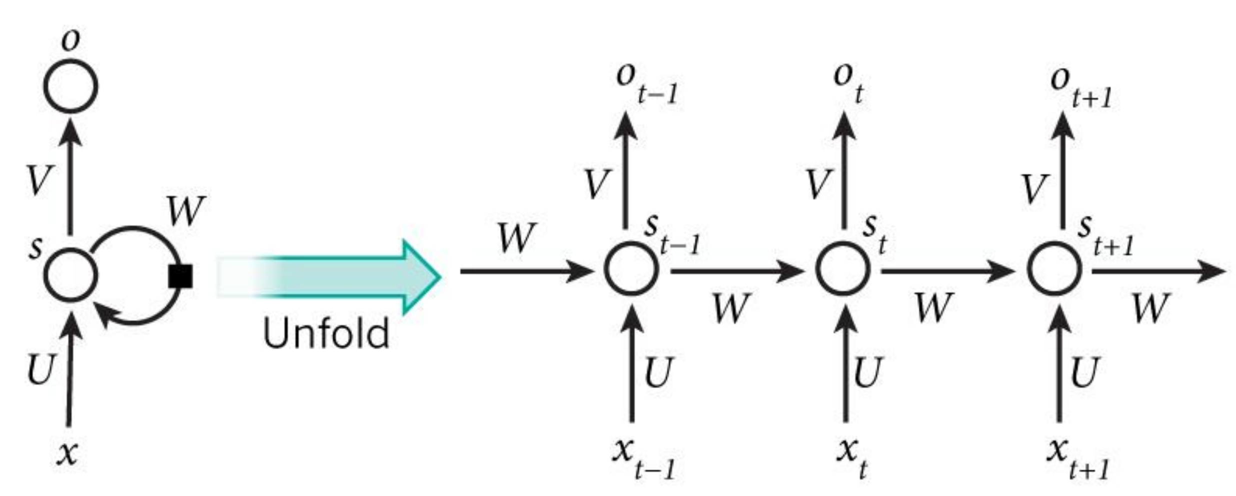
- Hidden layer represents underlying latent factors
- Prediction on 1 training item by updating it’s input values to “output values”
- Trained through Gibbs sampling

Recurrent neural networks



Recurrent neural network

Recurrent neural networks are feedforward networks that incorporate an internal state (memory) to process sequences of inputs.



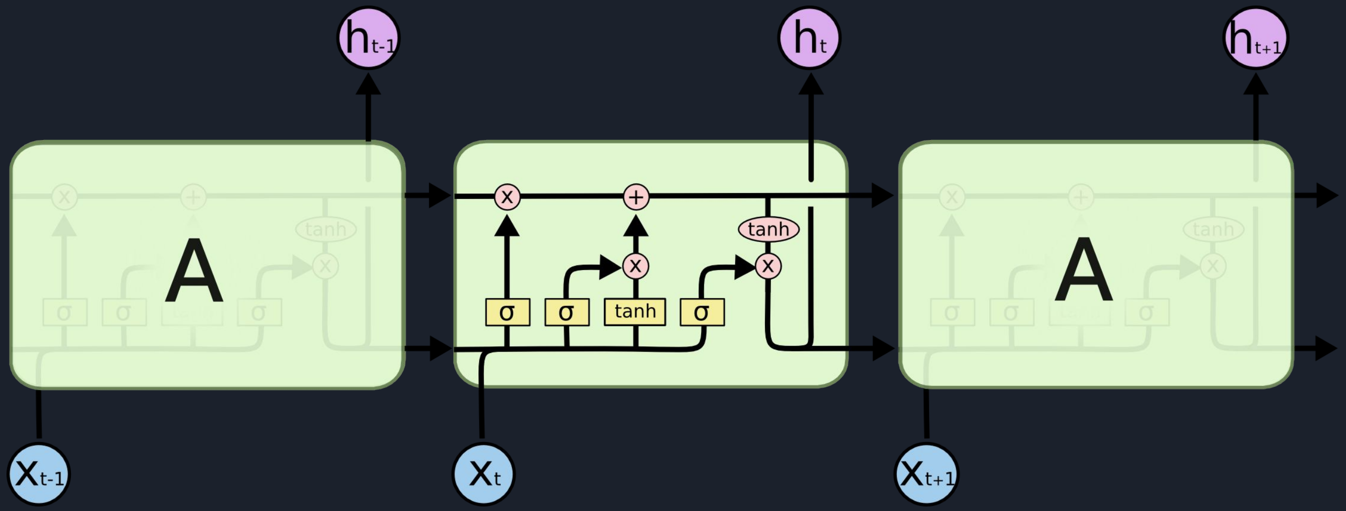
Vanishing or exploding gradient problem

- Gradient = derivative of activation function
- Depending on activation function, derivative can be small or large
- Training recurrent network through backpropagation
  - gradient can become vanishingly small → worst case: net stops training
  - gradient can become huge → very large update steps → unstable training
- Very deep feedforward networks (and thus RNN) suffer from this

Long Short Term Memory Networks (LSTM)

- Avoids the vanishing gradient problem by deciding what information needs to be kept
- How does it work?
  - Gets input from previous prediction
  - Also keeps explicit Cell State

Long Short Term Memory Networks (LSTM)



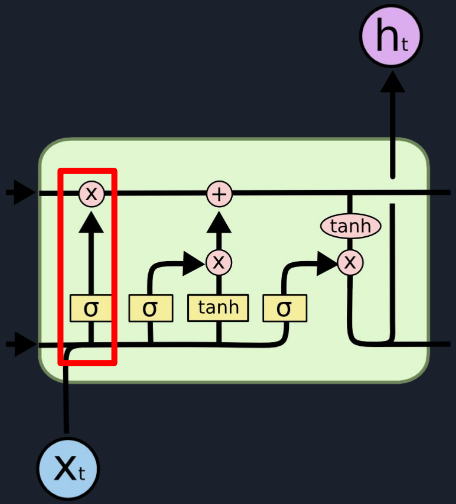
Forget gate

- Chooses what to forget from cell state

  ⇒ Allows to remember values indefinitely

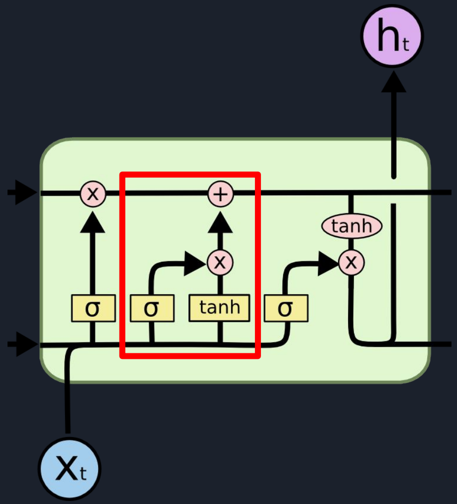
- Sigmoid function

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$



Input Gate

- Updates state based on input
  - Sigmoid decides what to update
  - Tanh activation function



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$C'_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Output gate

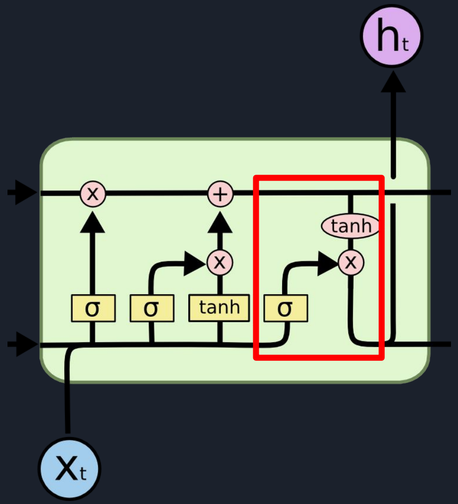
- Combine:
  - Cell state
  - Input

⇒ Predict output

$$C_t = f_t * C_{t-1} + i_t * C'_t$$

$$o_t = \sigma(W_o \,.\, [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$



Results and creative part

Classical Piano Composer

- Ignores rhythms -> converts everything to eight notes
    - major disadvantage
    - but trains faster
- Vocabulary = unique notes and chords
- LSTM based neural network with integer stream input

Loss function and optimizer

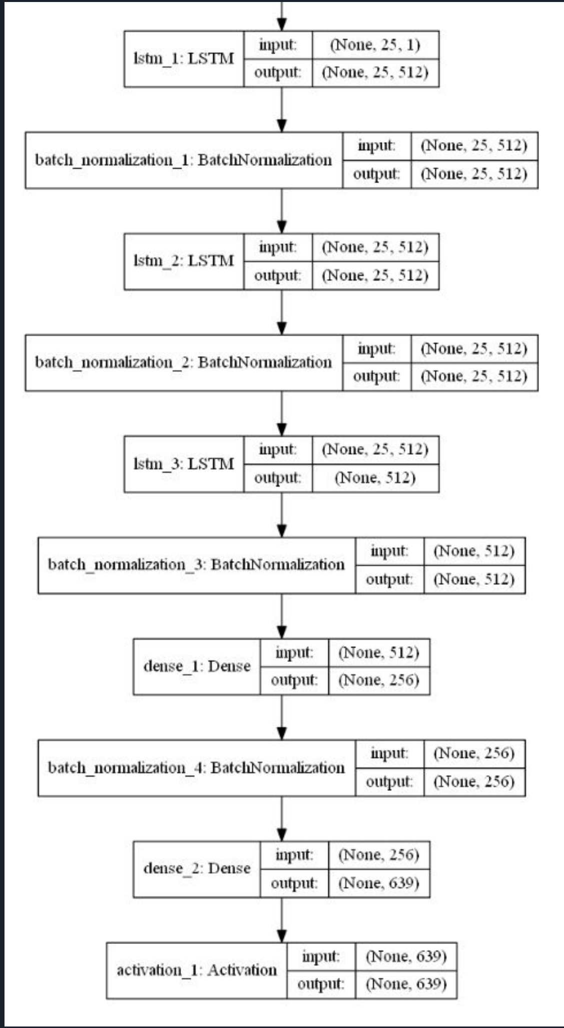
Loss function: categorical cross entropy

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{1}_{y_i\in C_c}\log p_{model}[y_i \in C_c]$$

Optimizer: RMSProp

Own changes

- Changes done to original neural network for better performance
  - Introduction of BatchNormalization instead of Dropout
  - Adam as optimizer instead of RMSProp
  - Increased batch size to 512


lstm_1: LSTM
input: (None, 25, 1)
output: (None, 25, 512)
batch_normalization_1: BatchNormalization
input: (None, 25, 512)
output: (None, 25, 512)
lstm_2: LSTM
input: (None, 25, 512)
output: (None, 25, 512)
batch_normalization_2: BatchNormalization
input: (None, 25, 512)
output: (None, 25, 512)
lstm_3: LSTM
input: (None, 25, 512)
output: (None, 512)
batch_normalization_3: BatchNormalization
input: (None, 512)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 256)
batch_normalization_4: BatchNormalization
input: (None, 256)
output: (None, 256)
dense_2: Dense
input: (None, 256)
output: (None, 639)
activation_1: Activation
input: (None, 639)
output: (None, 639)

Creative part: Experiments

- Effect of structure in music
- Effect of sequence length
- Effect of data quality

Datasets

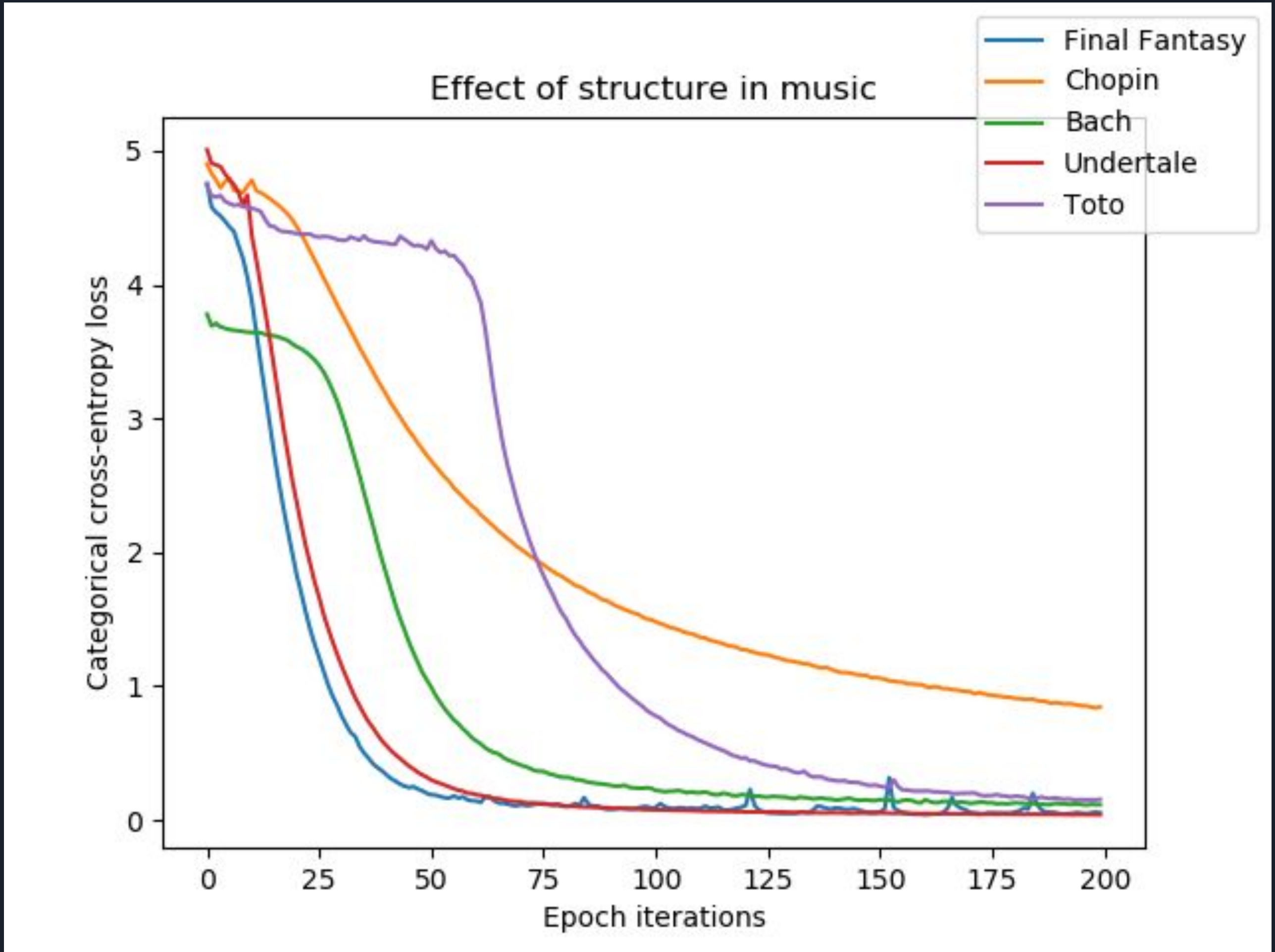
Comparison of different style databases and results

- Bach
- Chopin
- Undertale soundtrack
- Final Fantasy soundtrack
- Toto (with added drum track)

Sounds and tempo are manually picked for best results

Hypothesis

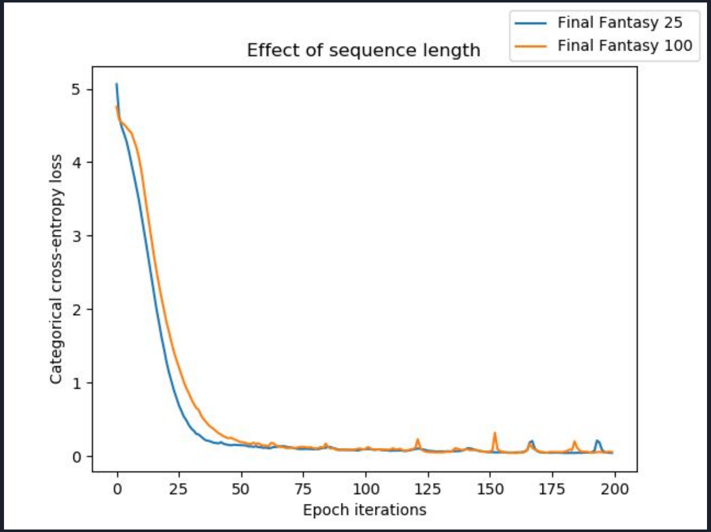
- Structured music is easier to understand and generalize
  - Bach → ‘mathematical’ music
  - 8 Bit gaming music → very repetitive and easy rhythms
    - Final Fantasy
    - Undertale
- Unstructured music is more complex to learn
  - Chopin → Romantic composer
  - Toto → Famous pop rock band

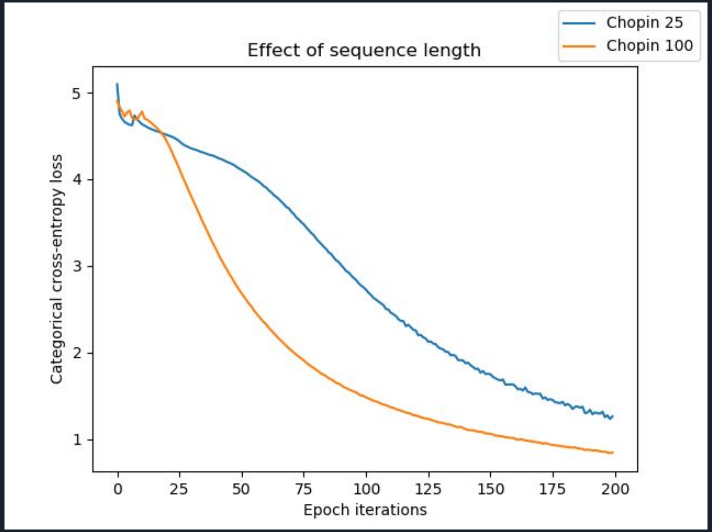


Effect of structure in music
Final Fantasy
Chopin
Bach
Undertale
Toto
Categorical cross-entropy loss
0
1
2
3
4
5
0
25
50
75
100
125
150
175
200
Epoch iterations

Effect of sequence length

- Longer patterns are easier to generalize + give better result
- Final fantasy is so structured that sequence length does not matter





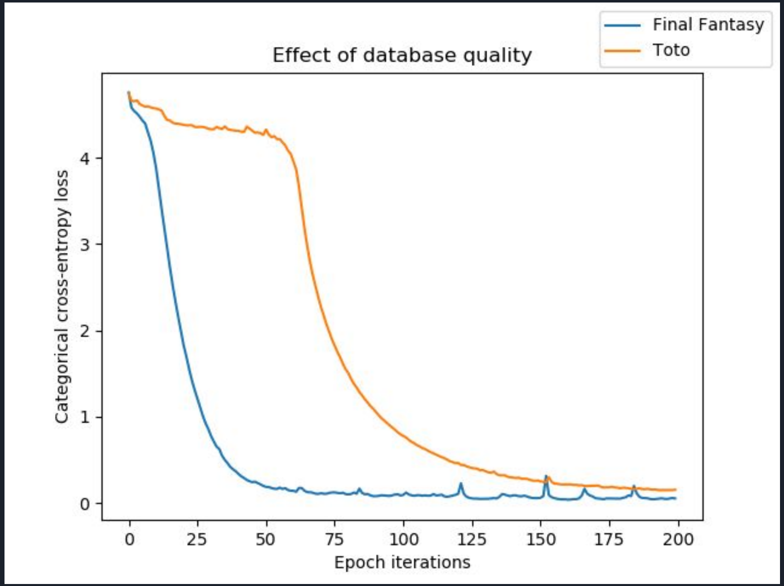
Sequence length: human evaluation

- ⇒ Notice with 100 sequence length:
  - coherence between bars
  - use of longer musical sentences
- ⇒ Notice with 25 sequence length:
  - Individual bars are correct
  - No coherence at all

Effect of data quality

- Toto dataset very noisy (multiple tracks)
  - ⇒ learns different instrument per song
  - difficult to generalize

⇒ Worse convergence



Creative part: Trying something new

- Natural Language Processing
  - Transform unique word tokens into integers
  - Convert these integers into word embeddings (can be pretrained)
  - Embedding == representation of meaning and context

Creative part: Trying something new

- Analogous:
    - Notes and chords converted into integers
    - But nobody uses embedding layers (no pretrained vectors available)
    - Our idea: train an embedding layer in the hope of discovering context between notes
    - Result: did not work, network failed to learn anything (due to random initialization of embeddings and too small database for learning contexts)

References

1. https://github.com/Skuldur/Classical-Piano-Composer
2. https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5
3. Colton, Simon, and Geraint A. Wiggins. "Computational creativity: The final frontier?." Ecai. Vol. 12. 2012.
4. Eck, Douglas, and Juergen Schmidhuber. "A first look at music composition using lstm recurrent neural networks." Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale 103 (2002).
5. Hilscher, Moritz, and Novin Shahroudi. "Music Generation from MIDI datasets."
6. Simoni, Mary, and Roger B. Dannenberg. Algorithmic Composition: A Guide to Composing Music with Nyquist. University of Michigan Press, 2013.
7. Briot, Jean-Pierre, Gaëtan Hadjeres, and François Pachet. "Deep learning techniques for music generation-a survey." arXiv preprint arXiv:1709.01620 (2017).
8. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
9. https://medium.com/cindicator/music-generation-with-neural-networks-gan-of-the-week-b66d01e28200
10. https://magenta.tensorflow.org/music-transformer
11. Boulanger-Lewandowski, Nicolas, Yoshua Bengio, and Pascal Vincent. "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription." arXiv preprint arXiv:1206.6392 (2012).
12. Mogren, Olof. "C-RNN-GAN: Continuous recurrent neural networks with adversarial training." arXiv preprint arXiv:1611.09904 (2016).
13. APA

Questions?