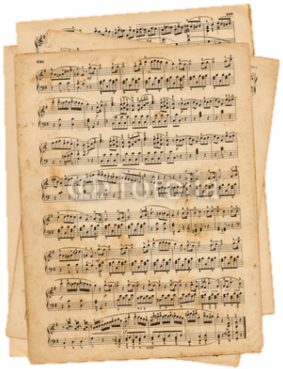
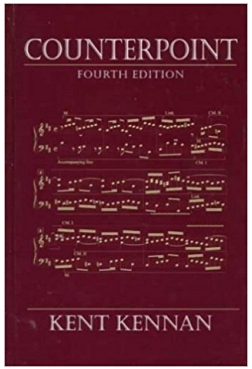
Introducing BachProp | a RNN for music composition

Florian Colombo

Music composition | How humans compose?





Music composition | Automated approaches

Grammars

Markov Chains

Evolutionary algorithms

Cellular automata

Artificial Neural Networks

Music composition | Markov constraints

Markov Chains

Cellular automata

Artificial Neural Networks

Flow Machines

by Sony CSL

Music composition | Artificial Neural Networks

Grammars

Markov Chains

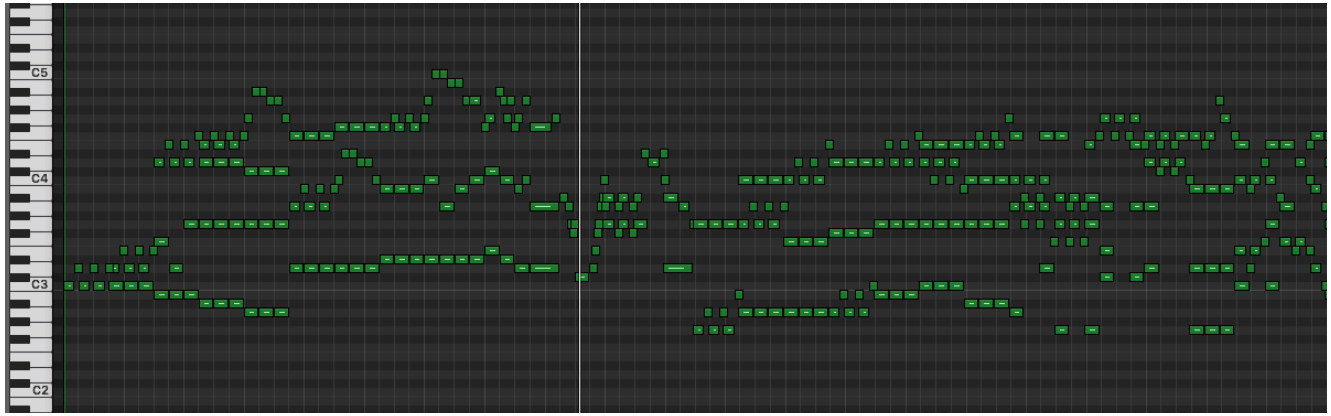
Evolutionary algorithms

Artificial Neural Networks

BachProp

Music is a complex temporal sequence

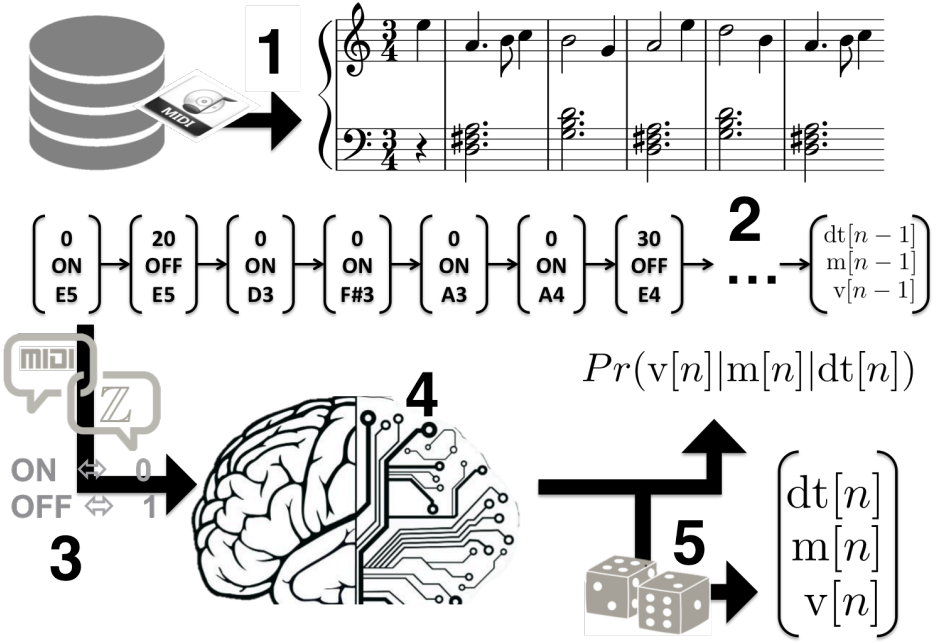
- Many temporal relationships governed by rules (stylistic, general, personal, …)



- BachProp is a probabilistic model that aims at capturing the temporal relationship between notes in specified music corpora

Outline

- Introduction
- Datasets
- From MIDI to BachProp
- Model architecture
- Training
- Generation
- Evaluation
- Discussion



Datasets | What's out there?

- Irish Folk
  - 45'000 Monophonic repetitive music on well defined scales
- Bach
  - 655 pieces from harmonized chorales, sonatas, concertos, …
  - Complex music with counterpoint (melodies that develop in parallel)
- Nottingham
  - 1037 Folk song (monophonic melody on simple chords)
- String quartets
  - 470 movements from Haydn, Mozart, Beethoven, Brahms, Dvorak and Schubert string quartets
- PianoMuscinelli
  - Collection of 25'000 piano music gathered on internet.
  - Heterogeneous





From MIDI to BachProp | MIDI

- MIDI is a standard notation for (symbolic) music
- MIDI is event-based

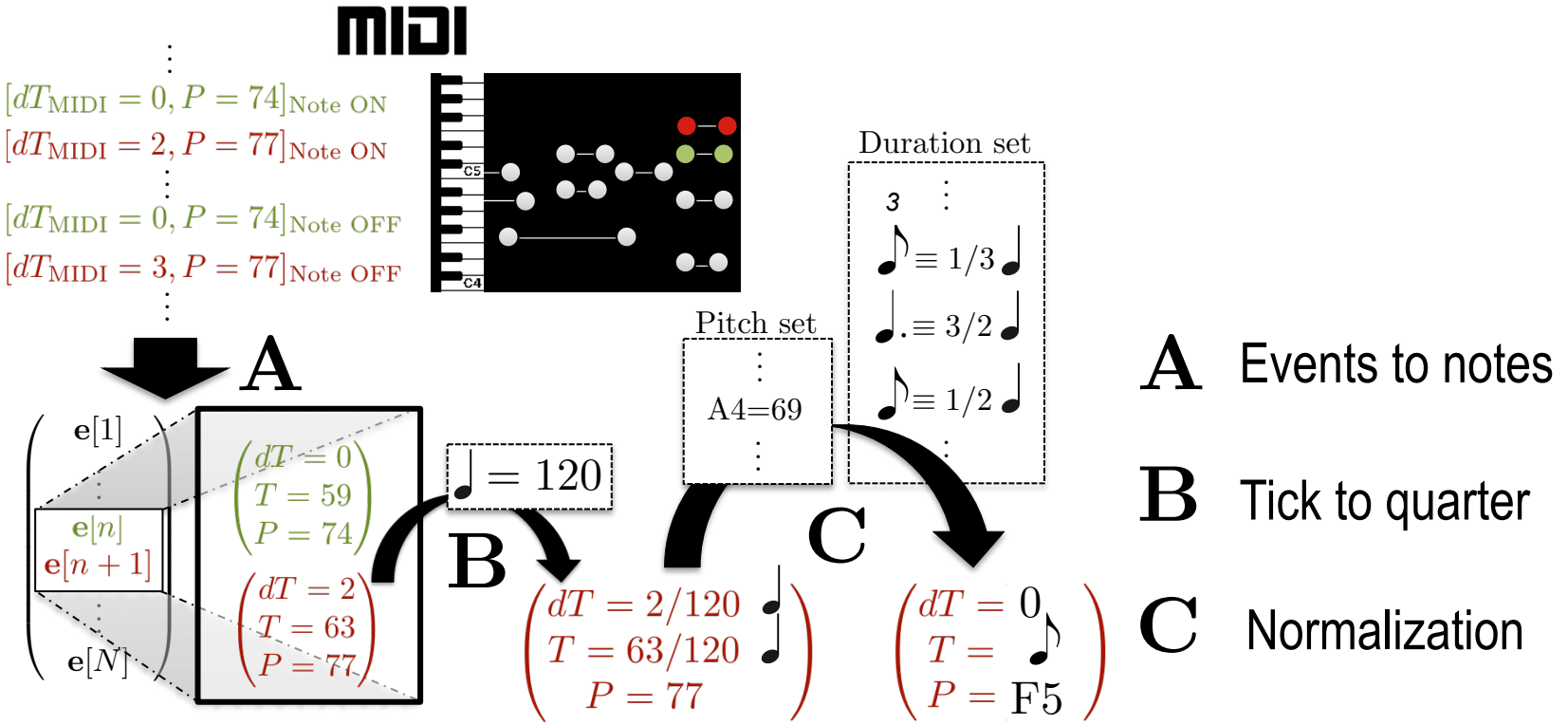
  {‘dt’: 120, ‘m’: ‘Note ON’, ‘value’: 55, ‘velocity’: 90)

  {‘dt’: 120, ‘m’: ‘Note OFF’, ‘value’: 55, ‘velocity’: 0)

  = {‘dt’: 120, ‘m’: ‘Note ON’, ‘value’: 55, ‘velocity’: 0)
- MIDI Metaparameter
  - Pulse Per Quarter note PPQ defines the duration of a quarter note.
  - Beats Per Minute PPM defines the tempo.

```
midi.Track(\
 [midi.ProgramChangeEvent(tick=0, channel=0, data=[0]),
  midi.NoteOnEvent(tick=0, channel=0, data=[60, 90]),
  midi.NoteOnEvent(tick=0, channel=0, data=[79, 90]),
  midi.NoteOnEvent(tick=0, channel=0, data=[84, 90]),
  midi.NoteOnEvent(tick=0, channel=0, data=[88, 90]),
  midi.NoteOffEvent(tick=120, channel=0, data=[60, 0]),
  midi.NoteOnEvent(tick=0, channel=0, data=[72, 90]),
  midi.NoteOffEvent(tick=120, channel=0, data=[72, 0]),
  midi.NoteOffEvent(tick=0, channel=0, data=[84, 0]),
  midi.NoteOffEvent(tick=0, channel=0, data=[88, 0]),
  midi.NoteOnEvent(tick=0, channel=0, data=[71, 90]),
  midi.NoteOnEvent(tick=0, channel=0, data=[86, 90]),
  midi.NoteOnEvent(tick=0, channel=0, data=[91, 90]),
  midi.NoteOffEvent(tick=120, channel=0, data=[71, 0]),
  midi.NoteOffEvent(tick=0, channel=0, data=[86, 0]),
  midi.NoteOffEvent(tick=0, channel=0, data=[91, 0]),
  midi.NoteOnEvent(tick=0, channel=0, data=[72, 90]),
  midi.NoteOnEvent(tick=0, channel=0, data=[84, 90]),
  midi.NoteOnEvent(tick=0, channel=0, data=[88, 90]),
  midi.NoteOffEvent(tick=120, channel=0, data=[72, 0]),
  midi.NoteOffEvent(tick=0, channel=0, data=[88, 0]),
  midi.NoteOnEvent(tick=0, channel=0, data=[69, 90]),
```

MIDI | Representation

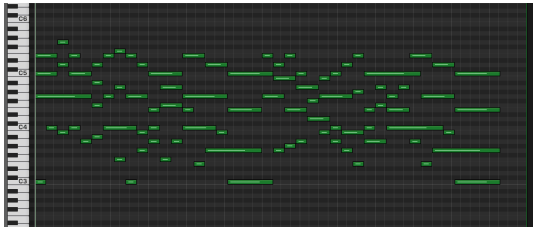


A Events to notes

B Tick to quarter

C Normalization

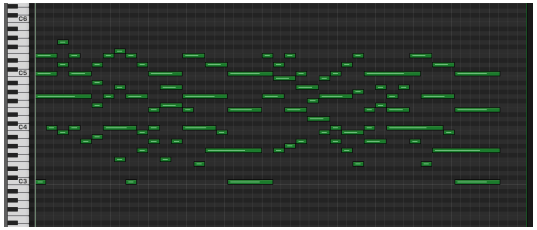
Representation | Events to notes



```
midi.NoteOnEvent(tick=0, channel=0, data=[64, 90])
midi.NoteOnEvent(tick=0, channel=0, data=[67, 90])
midi.NoteOnEvent(tick=0, channel=0, data=[71, 90])
midi.NoteOnEvent(tick=0, channel=0, data=[76, 90])
midi.NoteOffEvent(tick=120, channel=0, data=[67, 0])
midi.NoteOnEvent(tick=0, channel=0, data=[79, 90])
midi.NoteOffEvent(tick=240, channel=0, data=[64, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[71, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[79, 0])
```

n	dT	T	P
0	0	360	64
1	0	120	67
2	0	360	71
3	0	480	76
4	120	240	79

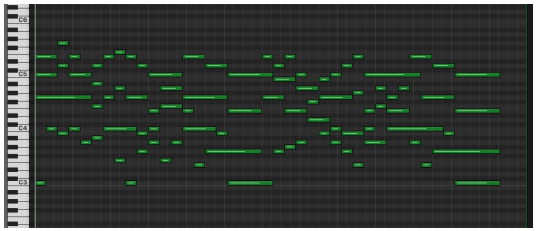
Representation | Ticks and PPQ heterogeneity



```
midi.NoteOnEvent(tick=0, channel=0, data=[64, 90])
midi.NoteOnEvent(tick=1, channel=0, data=[67, 90])
midi.NoteOnEvent(tick=0, channel=0, data=[71, 90])
midi.NoteOnEvent(tick=2, channel=0, data=[76, 90])
midi.NoteOffEvent(tick=120, channel=0, data=[67, 0])
midi.NoteOnEvent(tick=0, channel=0, data=[79, 90])
midi.NoteOffEvent(tick=241, channel=0, data=[64, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[71, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[79, 0])
```

n	dT	T	P
0	0	350	64
1	1	121	67
2	0	364	71
3	2	481	76
4	122	238	79

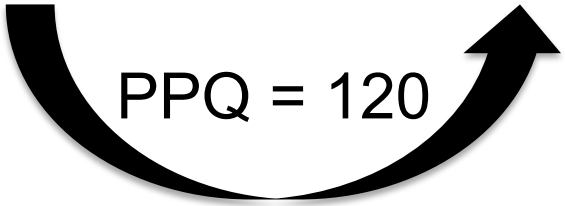
Representation | Ticks to quarter note lengths



```
midi.NoteOnEvent(tick=0, channel=0, data=[64, 90])
midi.NoteOnEvent(tick=1, channel=0, data=[67, 90])
midi.NoteOnEvent(tick=0, channel=0, data=[71, 90])
midi.NoteOnEvent(tick=2, channel=0, data=[76, 90])
midi.NoteOffEvent(tick=120, channel=0, data=[67, 0])
midi.NoteOnEvent(tick=0, channel=0, data=[79, 90])
midi.NoteOffEvent(tick=241, channel=0, data=[64, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[71, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[79, 0])
```

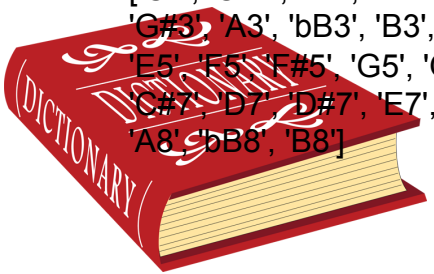
n	dT	T	P
0	0	350	64
1	1	121	67
2	0	364	71
3	2	481	76
4	122	238	79

n	dT	T	P
0	0	2.984	64
1	0.0001	1.001	67
2	0	3.002	71
3	0.0018	4.001	76
4	1.0001	1.978	79

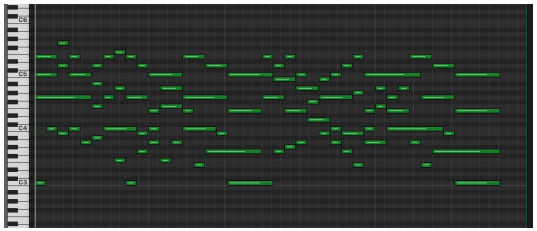


Dictionaries | Tokenization

duration
[0.0, 0.0417, 0.0833, 0.125, 0.1667, 0.1875, 0.25, 0.333, 0.375, 0.5, 0.666, 0.75, 1.0, 1.333, 1.5, 2.0, 2.666, 3.0, 4.0, 5.333, 6.0, 8.0, 12.0, 16.0, 24.0]
duration_text
['0', '32)3', '16)3', '32', '8)3', '32.', '16', '4)3', '16.', '8', '2)3', '8.', '4', '1)3', '4.', '2', '.5)3', '2.', '1', '.25)3', '1.', '.5', '.5.', '.25', '.25.']
pitch
[24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107]
pitch_text
['C2', 'C#2', 'D2', 'D#2', 'E2', 'F2', 'F#2', 'G2', 'G#2', 'A2', 'bB2', 'B2', 'C3', 'C#3', 'D3', 'D#3', 'E3', 'F3', 'F#3', 'G3', 'G#3', 'A3', 'bB3', 'B3', 'C4', 'C#4', 'D4', 'D#4', 'E4', 'F4', 'F#4', 'G4', 'G#4', 'A4', 'bB4', 'B4', 'C5', 'C#5', 'D5', 'D#5', 'E5', 'F5', 'F#5', 'G5', 'G#5', 'A5', 'bB5', 'B5', 'C6', 'C#6', 'D6', 'D#6', 'E6', 'F6', 'F#6', 'G6', 'G#6', 'A6', 'bB6', 'B6', 'C7', 'C#7', 'D7', 'D#7', 'E7', 'F7', 'F#7', 'G7', 'G#7', 'A7', 'bB7', 'B7', 'C8', 'C#8', 'D8', 'D#8', 'E8', 'F8', 'F#8', 'G8', 'G#8', 'A8', 'bB8', 'B8']



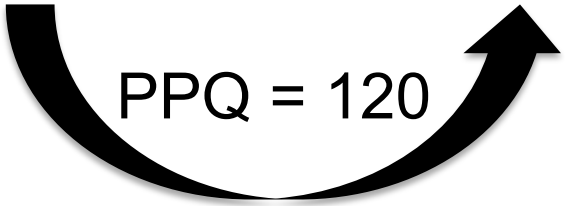
Representation | Tokenization and normalization



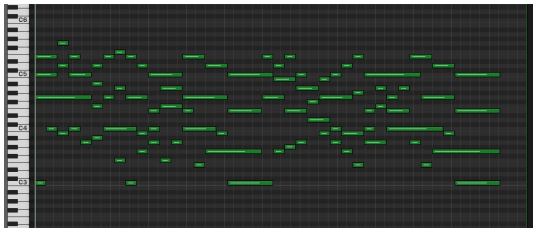
```
midi.NoteOnEvent(tick=0, channel=0, data=[64, 90])
midi.NoteOnEvent(tick=1, channel=0, data=[67, 90])
midi.NoteOnEvent(tick=0, channel=0, data=[71, 90])
midi.NoteOnEvent(tick=2, channel=0, data=[76, 90])
midi.NoteOffEvent(tick=120, channel=0, data=[67, 0])
midi.NoteOnEvent(tick=0, channel=0, data=[79, 90])
midi.NoteOffEvent(tick=241, channel=0, data=[64, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[71, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[79, 0])
```

n	dT	T	P
0	0	350	64
1	1	121	67
2	0	364	71
3	2	481	76
4	122	238	79

n	dT	T	P
0	0	2.984	64
1	0.0001	1.001	67
2	0	3.002	71
3	0.0018	4.001	76
4	1.0001	1.978	79



Representation | Tokenization and normalization

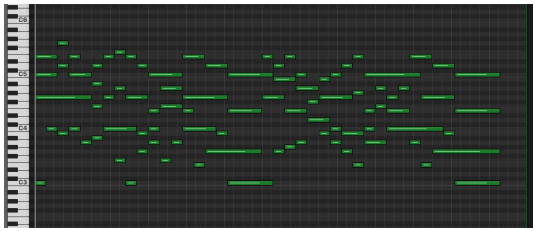


```
midi.NoteOnEvent(tick=0, channel=0, data=[64, 90])
midi.NoteOnEvent(tick=1, channel=0, data=[67, 90])
midi.NoteOnEvent(tick=0, channel=0, data=[71, 90])
midi.NoteOnEvent(tick=2, channel=0, data=[76, 90])
midi.NoteOffEvent(tick=120, channel=0, data=[67, 0])
midi.NoteOnEvent(tick=0, channel=0, data=[79, 90])
midi.NoteOffEvent(tick=241, channel=0, data=[64, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[71, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[79, 0])
```

n	dT	T	P
0	0	350	64
1	1	121	67
2	0	364	71
3	2	481	76
4	122	238	79

n	dT	T	P
0	0	3	64
1	0	1	67
2	0	3	71
3	0	4	76
4	1	2	79

Representation | Integer



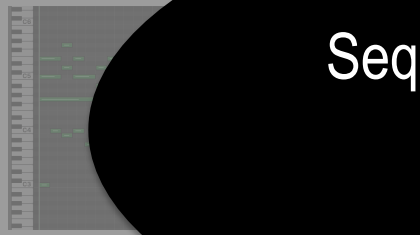
```
midi.NoteOnEvent(tick=0, channel=0, data=[64, 90])
midi.NoteOnEvent(tick=1, channel=0, data=[67, 90])
midi.NoteOnEvent(tick=0, channel=0, data=[71, 90])
midi.NoteOnEvent(tick=2, channel=0, data=[76, 90])
midi.NoteOffEvent(tick=120, channel=0, data=[67, 0])
midi.NoteOnEvent(tick=0, channel=0, data=[79, 90])
midi.NoteOffEvent(tick=241, channel=0, data=[64, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[71, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[79, 0])
```

n	dT	T	P
0	0	350	64
1	1	121	67
2	0	364	71
3	2	481	76
4	122	238	79

n	dT	T	P
0	0	6	21
1	0	4	24
2	0	6	28
3	0	8	33
4	4	3	36

PPQ = 120

Constrain rhythm to finite set of note duration



Sequence of integers representing a MIDI song
{dT[1:N], T[1:N], P[1:N]}

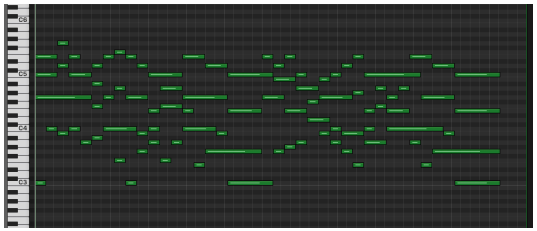
```
midi.NoteOnEvent(tick=0, channel=0, data=[76, 90])
midi.NoteOffEvent(tick=120, channel=0, data=[67, 0])
midi.NoteOnEvent(tick=0, channel=0, data=[79, 90])
midi.NoteOffEvent(tick=240, channel=0, data=[64, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[71, 0])
midi.NoteOffEvent(tick=0, channel=0, data=[79, 0])
```

n	dT	T	P
0	0	6	21
1	0	4	24
2	0	6	28
3	0	8	33
4	4	3	36

4 122 238 79

PPQ = 120

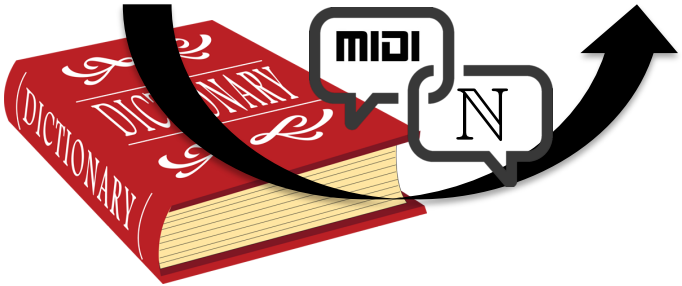
Representation | Irish folk dataset for miniproject 2



```
midi.NoteOnEvent(tick=1, channel=0, data=[64, 90])
midi.NoteOffEvent(tick=239, channel=0, data=[64, 0])
midi.NoteOnEvent(tick=1, channel=0, data=[67, 90])
midi.NoteOffEvent(tick=119, channel=0, data=[67, 0])
midi.NoteOnEvent(tick=1, channel=0, data=[71, 90])
midi.NoteOffEvent(tick=119, channel=0, data=[71, 0])
midi.NoteOnEvent(tick=1, channel=0, data=[79, 90])
midi.NoteOffEvent(tick=119, channel=0, data=[79, 0])
```

n	dT	T	P
0	1	239	64
1	1	119	67
2	1	119	71
3	1	119	79

n	T	P
0	3	21
1	2	24
2	2	28
3	2	36



From MIDI to BachProp | Summary

- Gather a big (homogenous) chorpus of MIDI file
- Construct the dictionaries
  - Constrain rhythm on a finite set of possible duration (21)
  - Constrain pitch on the range from lowest to highest note in the dataset
- For each song in the dataset
  - Extract from MIDI file, the sequence of events defined as (dT[n], T[n], P[n])
    - Convert to type 0 (single track)
    - Order chords
    - Use PPQ to translate tick durations to quarter note lengths
    - Map the duration to the dictionary entries
  - Use the dictionaries to map durations and pitches to unique integers
    - The resulting sequences are in the typical form of ANNs input data (before one-hot encoding)

Modeling note sequence with RNNs | BachProp

$$Pr(\mathbf{note}_s[1:N_s]) = Pr(\mathbf{note}_s[1]) \prod_{n=1}^{N_s-1} Pr(\mathbf{note}_s[n+1]|\mathbf{note}_s[1:n])$$

- RNNs approximate the transition probability as

$$\mathbf{y}[n] = \Phi(\mathbf{note}[n], \mathbf{H}[n]) \approx Pr(\mathbf{note}[n+1]|\mathbf{note}[1:n])$$

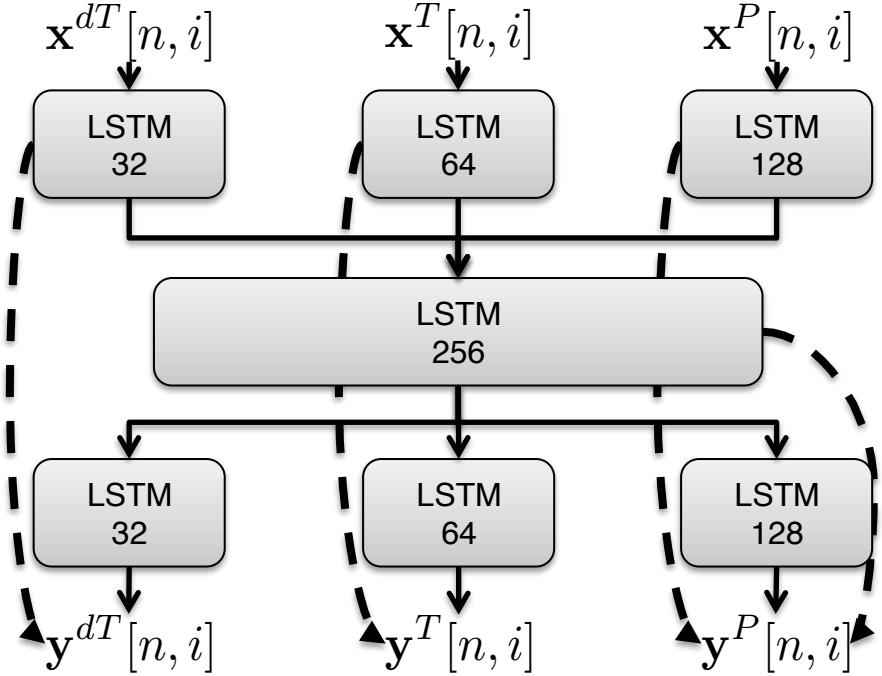
- The space of possible events as a combination of (dT, T, P) is huge (~1e5)

$$Pr(dT[n], T[n], P[n]||\mathbf{note}[1:n]) =$$
$$Pr(dT[n+1]|\mathbf{note}[1:n])\times$$
$$Pr(T[n+1]|\mathbf{note}[1:n], dT[n+1])\times$$
$$Pr(P[n+1]|\mathbf{note}[1:n], T[n+1], dT[n+1])$$

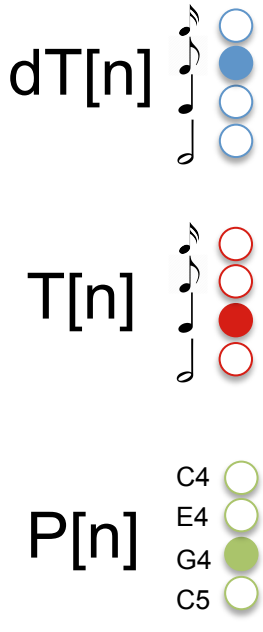
dT	T	P
0	6	21
0	4	24
0	6	28
0	8	33
4	3	36

Network architecture

- 3 LSTM layers
  - 1st and 3rd are feature-dependent
  - 2nd is shared across features
- All feature-dependent layers are connected to their feature's output layer
- The shared layer is connected to the pitch output layer

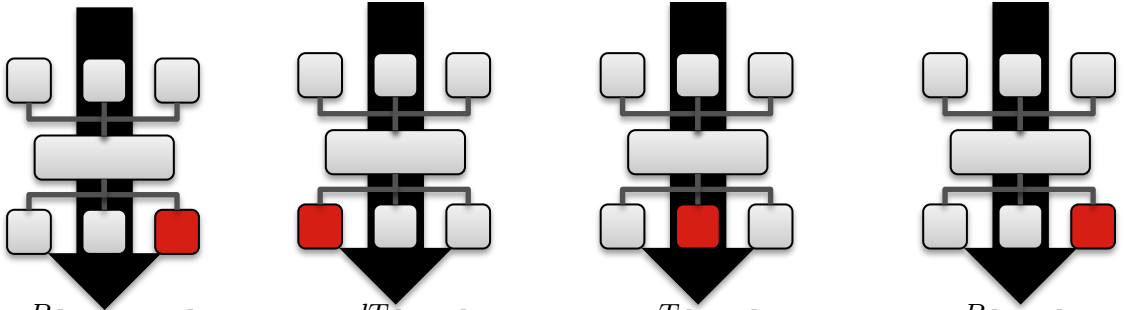


Conditional architecture | Unrolling features in time



$\mathbf{X}[n-1,3]$ $\mathbf{X}[n,1]$ $\mathbf{X}[n,2]$ $\mathbf{X}[n,3]$

$\begin{pmatrix} dT[n] \\ T[n] \\ P[n-1] \end{pmatrix}$ $\begin{pmatrix} dT[n] \\ T[n] \\ P[n] \end{pmatrix}$ $\begin{pmatrix} dT[n+1] \\ T[n] \\ P[n] \end{pmatrix}$ $\begin{pmatrix} dT[n+1] \\ T[n+1] \\ P[n] \end{pmatrix}$



$\mathbf{y}^P[n-1,3]$ $\mathbf{y}^{dT}[n,1]$ $\mathbf{y}^T[n,2]$ $\mathbf{y}^P[n,3]$

$Pr(P[n])$ $Pr(dT[n+1])$ $Pr(T[n+1])$ $Pr(P[n+1])$

Training

Modeling MIDI with RNNs | Miniproject 2

$$Pr(\mathbf{note}_s[1:N_s]) = Pr(\mathbf{note}_s[1]) \prod_{n=1}^{N_s-1} Pr(\mathbf{note}_s[n+1]|\mathbf{note}_s[1:n])$$

- RNNs approximate the transition probability as

$$\mathbf{y}[n] = \Phi(\mathbf{note}[n], \mathbf{H}[n]) \approx Pr(\mathbf{note}[n+1]|\mathbf{note}[1:n])$$

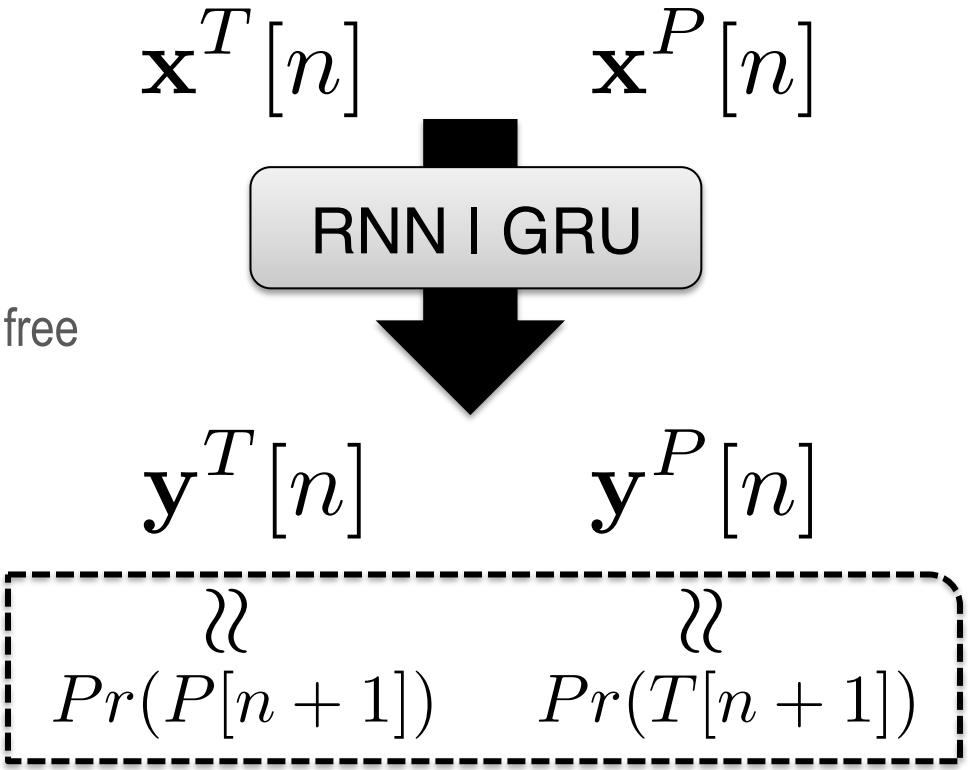
- The space of possible events as a combination of (T, P) is big (~1e3)

$$Pr(T[n], P[n]||\mathbf{note}[1:n]) =$$
$$Pr(T[n+1]|\mathbf{note}[1:n]) \times$$
$$Pr(P[n+1]|\mathbf{note}[1:n])$$

T	P
6	21
4	24
6	28
8	33
3	36

Network architecture | Miniproject 2

- Compare RNN vs GRU
- Hidden layer(s) architecture is free
  - Minimum 128 hidden units



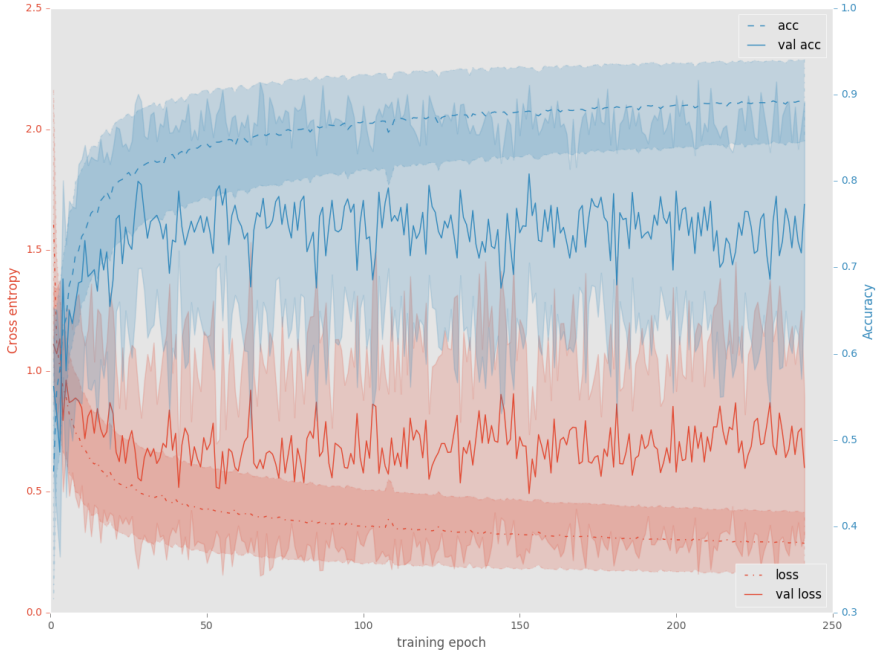
Deep learning techniques

- Multiple layers
  - Having 3 layers of LSTM cells allow for more complexity and broader timescale spectrum yielding better predictive performances
- Shortcut connections
  - Connections between a layer and the output allow the connected layer to learn representations directly (and indirectly) related to the desired output.
- Data augmentation
  - Between each training epoch, randomly transpose each song. Constrain on the possible pitch range
- Regularization
  - Dropout to prevent overfitting
- Masking
  - Allow to handle songs with variable lengths

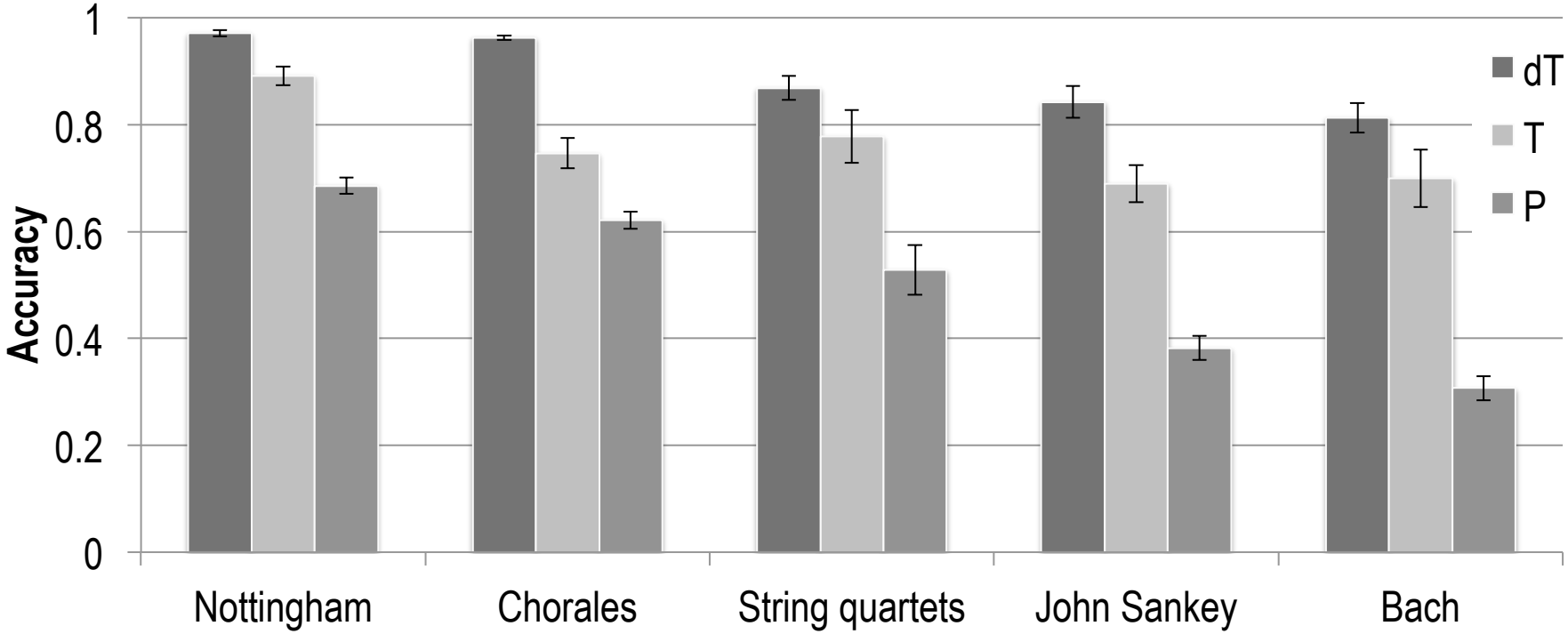
Training the networks

- For epoch and model:
  - Randomly transpose each song in dataset
  - For batch in training data
    - Reset model states
    - S_b → Randomly select batch_size (32) songs
    - X,T → Get the input/target sequences from S_b
    - Cut the X,T sequences in consecutive blocks of N(=50) timesteps
    - For each consecutive block:

      Use BPTT and Adam optimizer to apply the parameters update
  - Evaluate model on validating data
  - Record performance



Trained network performance on validating set



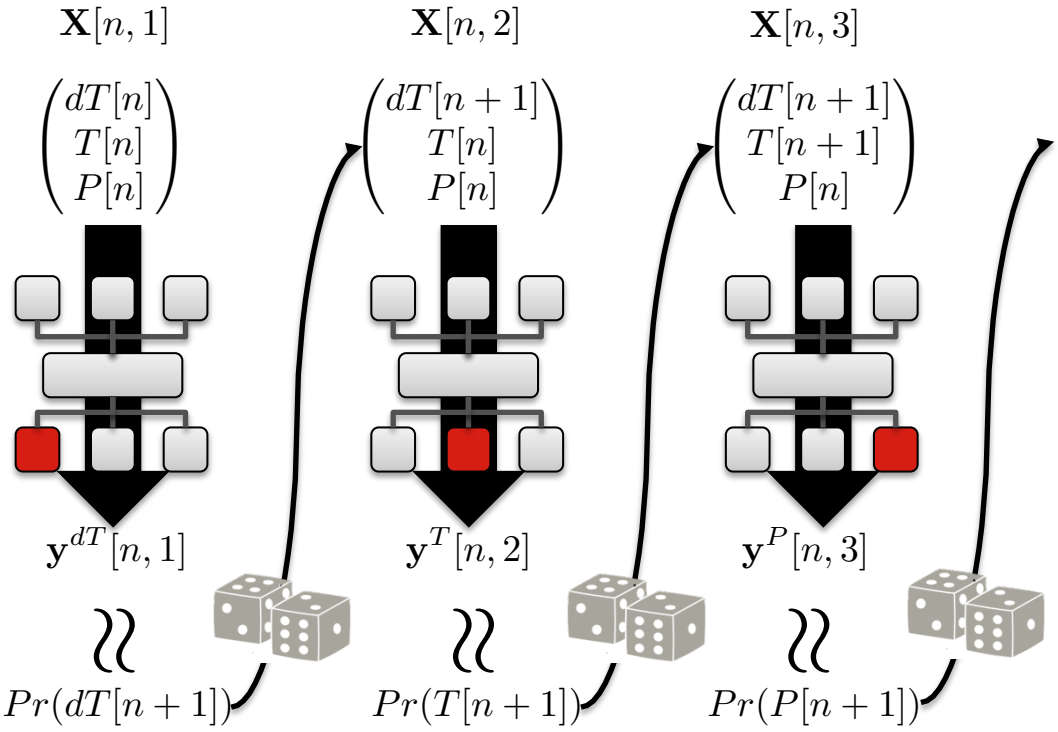
Architecture and training | Summary

- Conditional approach
  - Decrease the input/output dimensions
  - Allow to model the relations between melody and rhythm
- LSTMs ability to operate on multiple timescales allows to reach high performances on many different datasets
- dT > T > P
  - dT[n+1] can often be deduced from T[n] (especially true for monophonic melodies)
  - Rhythm is a more stable feature of music
  - Future works should be focused on improving pitch predictions
- Music with more complex structure is harder to learn
- Heterogeneous datasets are harder to learn

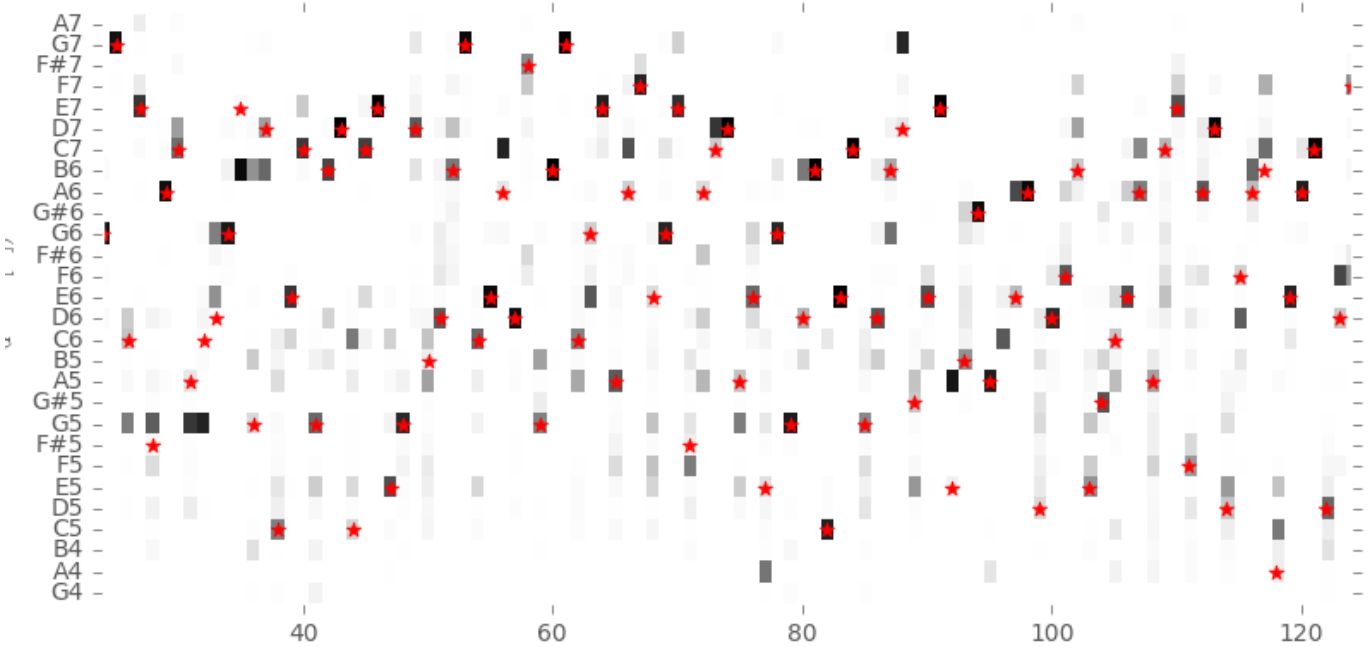
Generating with bachProp

For timestep n in generating song S:

- Sample dT[n+1] from Pr(dT[n+1]) = $Y^{dT}[n,1]$
- Sample T[n+1] from Pr(T[n+1]) = $Y^{T}[n,2]$
- Sample P[n+1] from Pr(P[n+1]) = $Y^{P}[n,3]$
- Add note e[n+1] = (dT[n+1], T[n+1], P[n+1]) to S



Trained model | Pitch prediction and generation



← Prediction | Generation →


Maestro?

Thanks for your attention

Evaluation | AI-generated music challenge