Using Convolutional Networks (with Attention) for Orders-of-Magnitude Speedup of DTW-Based Sequence Retrieval

Colin Raffel
Spotify Machine Learning Seminar
September 11, 2015
Music Information Retrieval Pipeline
The Million Song Dataset

Thierry Bertin-Mahieux et al. “The million song dataset”
Audio Available from 7digital

Schindler et al. “Facilitating Comprehensive Benchmarking Experiments on the Million Song Dataset”
Ground Truth?
import pretty_midi
# Load MIDI file into PrettyMIDI object
midi_data = pretty_midi.PrettyMIDI('midi_file.mid')
# Get a beat-synchronous piano roll
piano_roll = midi_data.get_piano_roll(times=midi_data.get_beats())
# Get tempo changes and their times
times, tempi = midi_data.get_tempo_change_times()
# Synthesize the resulting MIDI data at 22 kHz using fluidsynth
audio_data = midi_data.fluidsynth(fs=22050)
Matching

<table>
<thead>
<tr>
<th>artist</th>
<th>Tori Amos</th>
</tr>
</thead>
<tbody>
<tr>
<td>release</td>
<td>LIVE AT MONTREUX</td>
</tr>
<tr>
<td>title</td>
<td>'Smells Like Teen Spirit'</td>
</tr>
<tr>
<td>id</td>
<td>TRKUYPW128F92E1FC0</td>
</tr>
<tr>
<td>duration</td>
<td>216.4502</td>
</tr>
<tr>
<td>sample_rate</td>
<td>22050</td>
</tr>
<tr>
<td>audio_md5</td>
<td>'8'</td>
</tr>
<tr>
<td>7digitalid</td>
<td>5764727</td>
</tr>
<tr>
<td>year</td>
<td>1992</td>
</tr>
</tbody>
</table>
Matching by Metadata Won’t Work

J/Jerseygi.mid
V/VARIA180.MID
Carpenters/WeveOnly.mid
2009 MIDI/handy_man1-D105.mid
G/Garotos Modernos - Bailanta De Fronteira.mid
Various Artists/REWINDNAS.MID
GoldenEarring/Twilight_Zone.mid
Sure.Polyphone.Midi/Poly 2268.mid
d/danza3.mid
100%sure.polyphone.midi/Fresh.mid
rogers_kenny/medley.mid
2009 MIDI/looking_out_my_backdoor3-Bb192.mid
Aligning
DTW: Natural, and Too Slow

Hu et al., “Polyphonic Audio Matching and Alignment for Music Retrieval”
Similarity-Preserving Hashing
Similarity-Preserving Hashing
distance\([m, n]\) = \text{bits}_\text{set}[x[m] \oplus y[n]]
Training Data: Find Good Metadata

J/Jerseygi.mid
V/VARIA180.MID
Carpenters/WeveOnly.mid
2009 MIDI/handy_man1-D105.mid
G/Garotos Modernos - Bailanta De Fronteira.mid
Various Artists/REWINDNAS.MID
GoldenEarring/Twilight_Zone.mid
Sure.Polyphone.Midi/Poly_2268.mid

↓

Mc Broom, Amanda/The Rose.mid
Men At Work/Down Under.mid
Beach Boys, The/Barbara Ann.mid
Star Wars/Cantina.mid
T L C/CREEP.MID
Beatles/help.mid
Idol, Billy/White Wedding.mid
Training Data: Cleaning Metadata

Mc Broom, Amanda/The Rose.mid
Men At Work/Down Under.mid
Beach Boys, The/Barbara Ann.mid
Star Wars/Cantina.mid
T L C/CREEP.MID
Beatles/help.mid
Idol, Billy/White Wedding.mid

Amanda McBroom/The Rose.mid
Men At Work/Down Under.mid
The Beach Boys/Barbara Ann.mid
TLC/Creep.mid
The Beatles/Help!.mid
Billy Idol/White Wedding.mid

25,000 → 17,000 (10,000)
Training Data: Text Matching

Amanda McBroom/The Rose.mid
Men At Work/Down Under.mid
The Beach Boys/Barbara Ann.mid
TLC/Creep.mid
The Beatles/Help!.mid
Billy Idol/White Wedding.mid

men_at_work/Brazil/07-Down_Under.mp3,
TRLMFJ024KJ42K215E
TRFBTK0128F426441E
tlc/Crazy_Sexy_Cool/02-Creep.mp3
The Beatles - Help!.mp3

17,000 (9,000) → 26,000 (5,000)
Training Data: Alignment

1. Compute beat-synchronized CQTs of audio and synthesized MIDI
Training Data: Alignment

1. Compute beat-synchronized CQTs of audio and synthesized MIDI
2. Compute a pairwise distance matrix of CQTs
Training Data: Alignment

1. Compute beat-synchronized CQTs of audio and synthesized MIDI
2. Compute a pairwise distance matrix of CQTs
3. Use DTW to find lowest-cost path through the distance matrix
1. Compute beat-synchronized CQTs of audio and synthesized MIDI
2. Compute a pairwise distance matrix of CQTs
3. Use DTW to find lowest-cost path through the distance matrix
4. Allow subsequence matching, with some tolerance
Training Data: Alignment

1. Compute beat-synchronized CQTs of audio and synthesized MIDI
2. Compute a pairwise distance matrix of CQTs
3. Use DTW to find lowest-cost path through the distance matrix
4. Allow subsequence matching, with some tolerance
5. Use an additive penalty (e.g. median distance)
Training Data: Alignment

1. Compute beat-synchronized CQTs of audio and synthesized MIDI
2. Compute a pairwise distance matrix of CQTs
3. Use DTW to find lowest-cost path through the distance matrix
4. Allow subsequence matching, with some tolerance
5. Use an additive penalty (e.g. median distance)
6. Compute the total distance between aligned frames
Training Data: Alignment

1. Compute beat-synchronized CQTs of audio and synthesized MIDI
2. Compute a pairwise distance matrix of CQTs
3. Use DTW to find lowest-cost path through the distance matrix
4. Allow subsequence matching, with some tolerance
5. Use an additive penalty (e.g. median distance)
6. Compute the total distance between aligned frames
7. Normalize by path length and mean of path submatrix
Training Data: Alignment

1. Compute beat-synchronized CQTs of audio and synthesized MIDI
2. Compute a pairwise distance matrix of CQTs
3. Use DTW to find lowest-cost path through the distance matrix
4. Allow subsequence matching, with some tolerance
5. Use an additive penalty (e.g. median distance)
6. Compute the total distance between aligned frames
7. Normalize by path length and mean of path submatrix

![AUC: 0.9859](image-url)
import librosa
# We could also obtain audio data from pretty_midi's fluidsynth method
audio, fs = librosa.load('audio_file.mp3')
# Compute a CQT with 48 notes, 12 bins per octave, starting from C3
cqt = np.abs(librosa.cqt(audio, fmin=librosa.midi_to_hz(36), n_bins=48))
# Compute onset envelope from CQT (for speed)
onset_envelope = librosa.onset.onset_strength(S=cqt, aggregate=np.median)
# Perform beat tracking using CQT-based onset strength
bpm, beats = librosa.beat.beat_track(onset_envelope=onset_envelope)
# Synchronize the CQT to the beats
sync_cqt = librosa.feature.sync(cqt, beats)
# Compute log amplitude
sync_cqt = librosa.logamplitude(sync_cqt, ref_power=sync_cqt.max())
# L2 normalize
sync_gram = librosa.util.normalize(sync_cqt, norm=2.)
Loss function
Training details

- Successful alignments split by song 50%/25%/25% train/development/test

- 2.3M train examples, 10% used as validation set

- Negative examples chosen at random

- Data passed to network as batches of randomly sampled length-100 subsequences

- Early-stopping using validation set cost

- Optimization using RMSProp

- No other regularization needed

- Hyperparameters chosen using Whetlab (RIP)

- Objective: Bhattacharyya distance of positive/negative examples distance distributions
Training details

- Successful alignments split by song 50%/25%/25% train/development/test
- \( \approx 2.3 \text{M train examples, 10\% used as validation set} \)
Training details

- Successful alignments split by song 50%/25%/25% train/development/test
- ≈ 2.3M train examples, 10% used as validation set
- Negative examples chosen at random

Optimization using RMSProp

No other regularization needed

Hyperparameters chosen using Whetlab (RIP)

Objective: Bhattacharyya distance of positive/negative examples distance distributions
Training details

- Successful alignments split by song 50%/25%/25% train/development/test
- \( \approx 2.3 \text{M train examples, 10\% used as validation set} \)
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
Training details

- Successful alignments split by song 50%/25%/25% train/development/test
- \( \approx 2.3 \text{M} \) train examples, 10% used as validation set
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost

Optimization using RMSProp
No other regularization needed
Hyperparameters chosen using Whetlab (RIP)
Objective: Bhattacharyya distance of positive/negative examples distance distributions
Training details

- Successful alignments split by song 50%/25%/25% train/development/test
- \( \approx 2.3 \text{M} \) train examples, 10% used as validation set
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost
- Optimization using RMSProp
Training details

- Successful alignments split by song 50%/25%/25% train/development/test
- \( \approx \) 2.3M train examples, 10% used as validation set
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost
- Optimization using RMSProp
- No other regularization needed
Training details

- Successful alignments split by song 50%/25%/25% train/development/test
- ≈ 2.3M train examples, 10% used as validation set
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost
- Optimization using RMSProp
- No other regularization needed
- Hyperparameters chosen using Whetlab (RIP)
Training details

- Successful alignments split by song 50%/25%/25% train/development/test
- $\approx 2.3$M train examples, 10% used as validation set
- Negative examples chosen at random
- Data passed to network as batches of randomly sampled length-100 subsequences
- Early-stopping using validation set cost
- Optimization using RMSProp
- No other regularization needed
- Hyperparameters chosen using Whetlab (RIP)
- Objective: Bhattacharyya distance of positive/negative examples distance distributions
Network Structure

- Two different networks with the same structure used for audio and MIDI sequences
Network Structure

- Two different networks with the same structure used for audio and MIDI sequences
- Two convolutional layers: 5x12 and 3x3
Network Structure

- Two different networks with the same structure used for audio and MIDI sequences
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
Network Structure

- Two different networks with the same structure used for audio and MIDI sequences
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
Network Structure

- Two different networks with the same structure used for audio and MIDI sequences
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output

16-bit hashes created by thresholding output

Weight matrices initialized using He's method, $\{fan\}_{in}$

Bias vectors all initialized to zero

Network made out of lasagne
Network Structure

- Two different networks with the same structure used for audio and MIDI sequences
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output
- 16-bit hashes created by thresholding output

Weight matrices initialized using He's method, a

Bias vectors all initialized to zero

Network made out of lasagne
Network Structure

- Two different networks with the same structure used for audio and MIDI sequences
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output
- 16-bit hashes created by thresholding output
- Weight matrices initialized using He’s method, $\sqrt{2/fan\_in}$
Network Structure

- Two different networks with the same structure used for audio and MIDI sequences
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output
- 16-bit hashes created by thresholding output
- Weight matrices initialized using He’s method, $\sqrt{2/fan_{in}}$
- Bias vectors all initialized to zero
Network Structure

- Two different networks with the same structure used for audio and MIDI sequences
- Two convolutional layers: 5x12 and 3x3
- Two max-pooling layers, both 2x2
- Two dense layers with 2048 units each
- ReLUs throughout, with tanh on the output
- 16-bit hashes created by thresholding output
- Weight matrices initialized using He’s method, $\sqrt{2/fan\_in}$
- Bias vectors all initialized to zero
- Network made out of lasagne
Validation Distance Distribution
Example Sequence
First Layer Filters
Test: MIDI-to-MSD Matching

![Graph showing comparison between Hash Sequence DTW and Baseline methods. The x-axis represents rank, and the y-axis represents percentage below a threshold. The graph shows a significant improvement with Hash Sequence DTW over the Baseline method.]
Can We Do Better?

- Discard 99% of the MSD in 1% of the time
Can We Do Better?

- Discard 99% of the MSD in 1% of the time
- Would nevertheless take weeks
Can We Do Better?

- Discard 99% of the MSD in 1% of the time
- Would nevertheless take weeks
- Most errors caused by Beat Tracking
Can We Do Better?

- Discard 99% of the MSD in 1% of the time
- Would nevertheless take weeks
- Most errors caused by beat tracking
- System requires aligned training data
Can We Do Better?

- Discard 99% of the MSD in 1% of the time
- Would nevertheless take weeks
- Most errors caused by beat tracking
- System requires aligned training data
- Still relies on DTW
Can We Do Better?

- Discard 99% of the MSD in 1% of the time
- Would nevertheless take weeks
- Most errors caused by beat tracking
- System requires aligned training data
- Still relies on DTW
- Recurrent networks for more context?
Can We Do Better?

- Discard 99% of the MSD in 1% of the time
- Would nevertheless take weeks
- Most errors caused by beat tracking
- System requires aligned training data
- Still relies on DTW
- Recurrent networks for more context?
Sequence Embedding
Sequence Embedding
\[ \alpha = \text{softmax}(wx + b) \]

\[ w \in \mathbb{R}^{n_{\text{features}}}, \quad b \in \mathbb{R}, \quad \alpha \in \mathbb{R}^{n_{\text{steps}}} \]
Other Differences

- Batches of entire (cropped) sequences
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not aligned
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not aligned
- No beat-synchronization
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not aligned
- No beat-synchronization
- Higher “correct alignment” threshold
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not aligned
- No beat-synchronization
- Higher “correct alignment” threshold
- Re-tune hyperparameters with simple_spearmint
Other Differences

- Batches of entire (cropped) sequences
- Sequences are not aligned
- No beat-synchronization
- Higher “correct alignment” threshold
- Re-tune hyperparameters with `simple_spearmint`
- Output is now $[-1, 1]^{128}$
Validation Distance Distribution
Example Embeddings
Embedding Distance Matrix
MIDI-to-MSD Matching

![Graph showing MIDI-to-MSD matching results](image-url)
Related Work
Thanks!

http://github.com/craffel/midi-dataset
http://github.com/craffel/sequence-embedding
http://github.com/bmcfee/librosa
http://github.com/Lasagne/Lasagne
http://github.com/craffel/pretty-midi
http://github.com/craffel/simple_spearmint

craffel@gmail.com