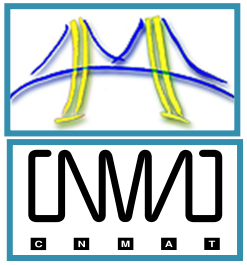
Teaching Computers to Listen to Music

Eric Battenberg http://ericbattenberg.com

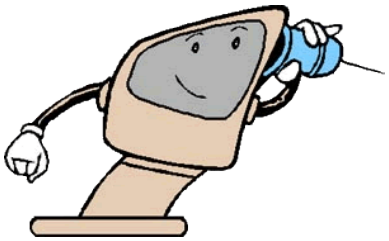
Gracenote, Inc.

Previously:

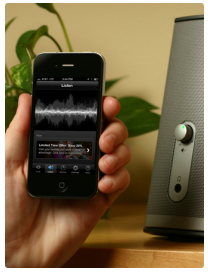
UC Berkeley, Dept. of EECS
Parallel Computing Laboratory
CNMAT (Center for New Music and Audio Technologies)



Machine Listening



- Speech processing
  - Speech processing makes up the vast majority of funded machine listening research.
  - Just as there's more to Computer Vision than OCR, there's more to machine listening than speech recognition!



- Audio content analysis
  - Audio fingerprinting (e.g. Shazam, Gracenote)
  - Audio classification (music, speech, noise, laughter, cheering)
  - Audio event detection (new song, channel change, hotword)
- Content-based Music Information Retrieval (MIR)
  - Today's topic

Getting Computers to "Listen" to Music

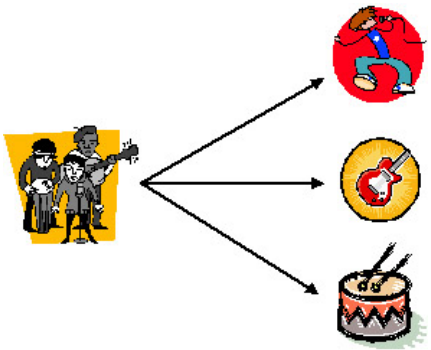
- Not trying to get computers to "listen" for enjoyment.
- More accurate: Analyzing music with computers.
- What kind of information to get out of the analysis?
  - What instruments are playing?
  - What is the mood?
  - How fast/slow is it (tempo)?
  - What does the singer sound like?
  - How can I play this song on the guitar/drums?

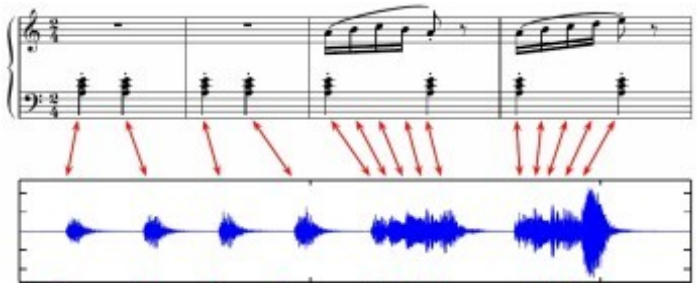
Ben Harper

James Brown

Content-Based Music Information Retrieval

- Many tasks:
  - Genre, mood classification, auto-tagging
  - Beat tracking, tempo detection
  - Music similarity, playlisting, recommendation
    - Heavily aided by collaborative filtering
  - Automatic music transcription
  - Source separation (voice, drums, bass…)
  - Music segmentation (verse, chorus)





Talk Summary

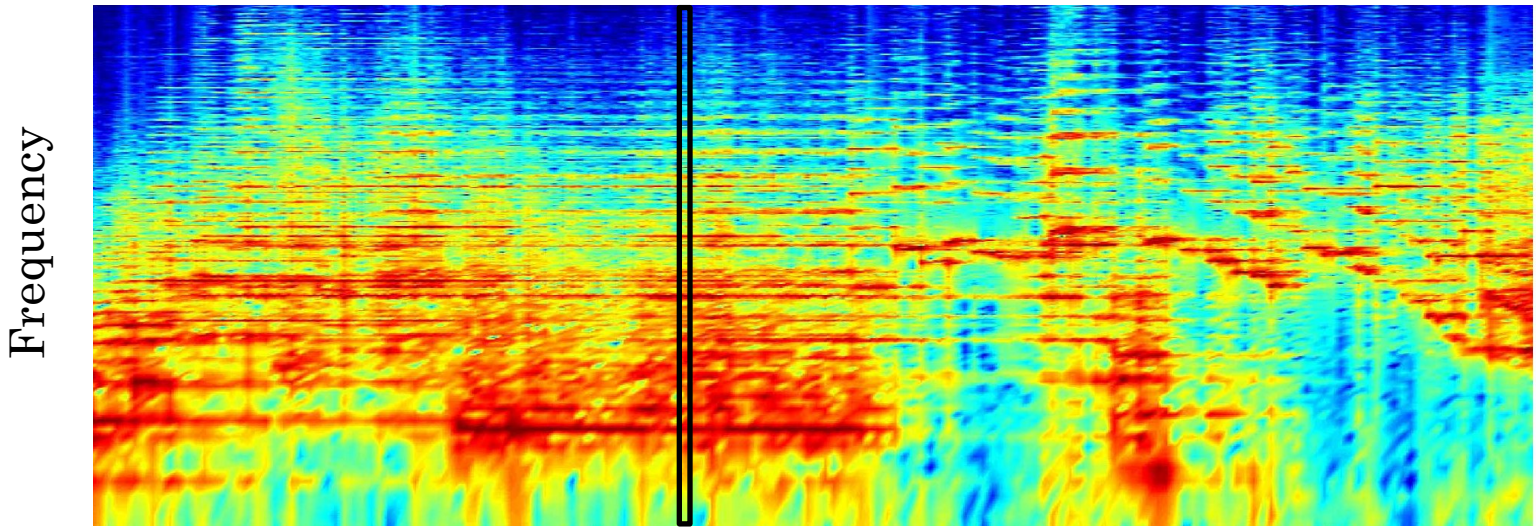
- Introduction to Music Information Retrieval
  - Some common techniques
  - Exciting new research directions
- Live Drum Understanding
  - Drum detection/transcription
  - Drum pattern analysis

Talk Summary

- **Introduction to Music Information Retrieval**
  - Some common techniques
  - Exciting new research directions
- Live Drum Understanding
  - Drum detection/transcription
  - Drum pattern analysis

Quick Lesson: The Spectrogram

- The spectrogram: Very common feature used in audio analysis.
- Time-frequency representation of audio.
- Take FFT of adjacent frames of audio samples, put them in a matrix.
- Each column shows frequency content at a particular time.



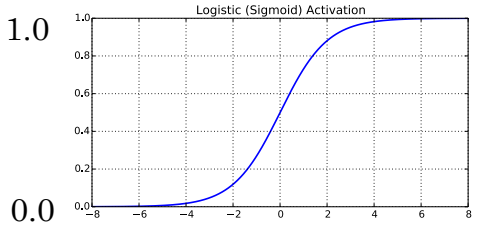
Genre/Mood Classification: Typical Approaches

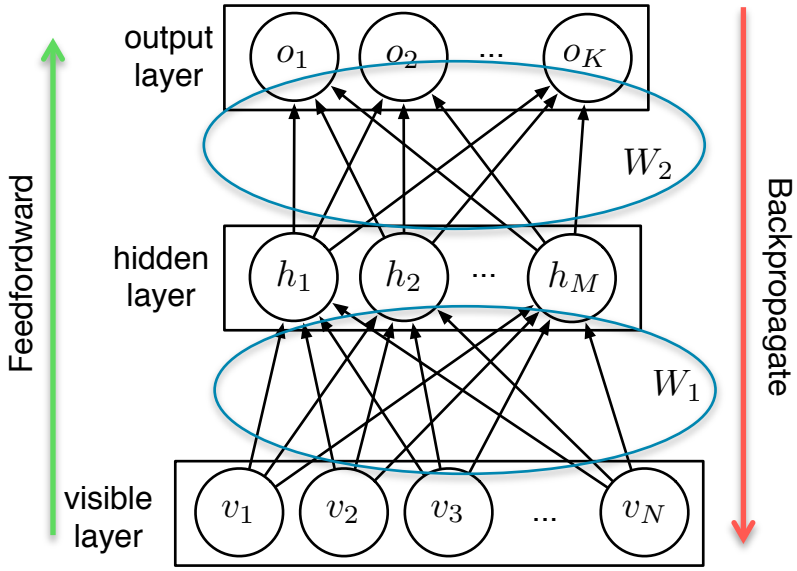
- Typical approach:
  - Extract **a bunch of hand-designed features** describing small windows of the signal (e.g., spectral centroid, kurtosis, harmonicity, percussiveness, MFCCs, 100's more…).
  - **Train a GMM or SVM** to predict genre/mood/tags by either:
    - Summarizing a song using mean/variance of each feature
    - Log-likelihood sum across frames (GMM) or frame-wise voting (SVM)
- Pros:
  - Works fairly well, was state of the art for a while.
  - Well understood models, implementations widely available.
- Cons:
  - Bag-of-frames style approach lacks ability to describe rhythm and temporal dynamics.
  - Getting further improvements requires hand designing more features.

Learning Features: Neural Networks

- Each layer computes a non-linear transformation of the previous layer.
  - Linear transformation (weight matrices)
  - + non-linearity (e.g. sigmoid [σ])
  - $h = \sigma(W_1 v)$, $o = \sigma(W_2 h)$
- Train to minimize output error.
- Each hidden layer can be thought of as a set of features.
- Train using backpropagation.
- Iterative steps:
  - Compute activations
  - Compute output error
  - Backprop. error signal
  - Compute gradients
  - Update all weights.
- Resurgence of neural networks:
  - More compute
  - More data
  - A few new tricks…

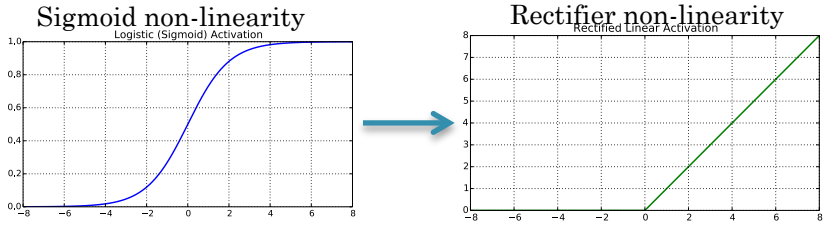
Sigmoid non-linearity

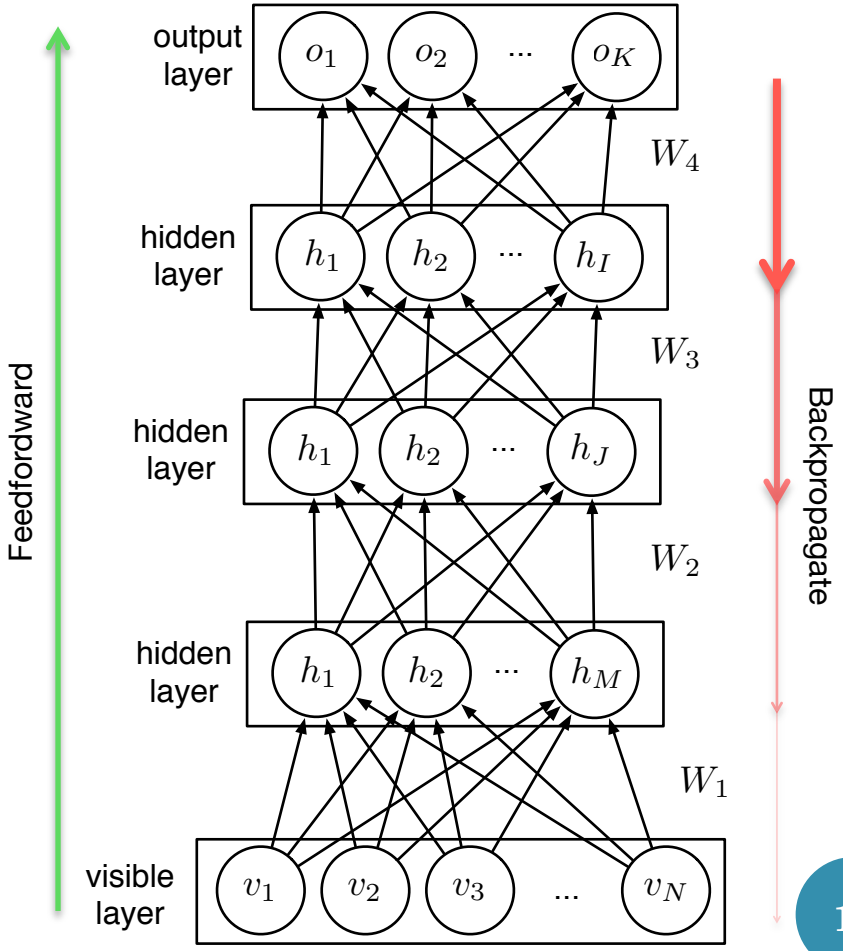




Deep Neural Networks

- Deep Neural Networks
  - Millions to billions of parameters
  - Many layers of "features"
  - Achieving state of the art performance in vision and speech tasks.
- Problem: Vanishing error signal.
  - Weights of lower layers do not change much.
- Solutions:
  - Train for a really long time. ☹
  - Pre-train each hidden layer as an autoencoder. [Hinton, 2006]
  - Rectified Linear Units [Krizhevsky, 2012]

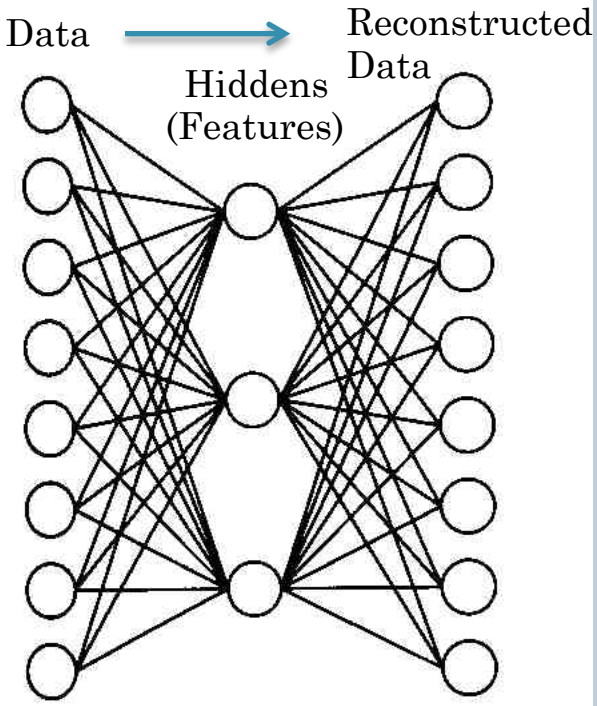




Autoencoders and Unsupervised Feature Learning

- Many ways to learn features in an unsupervised way:
- Autoencoders – train a network to reconstruct the input
  - Restricted Boltzmann Machine (RBM) [Hinton, 2006]
  - Denoising Autoencoders [Vincent, 2008]
  - Sparse Autoencoders
- Clustering – K-Means, mixture models, etc.
- Sparse Coding – learn overcomplete dictionary of features with sparsity constraint

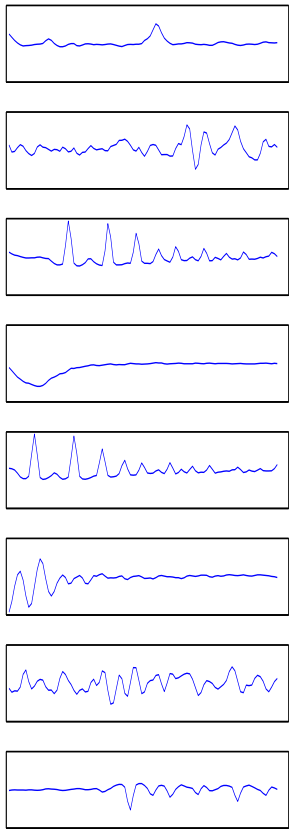
Autoencoder:
Train to reconstruct input



Genre/Mood Classification: Newer Approaches

- Newer approaches to feature extraction:
  - Learn spectral features using Restricted Boltzmann Machines (RBMs) and Deep Neural Networks (DNN) [Hamel, 2010] – good genre performance.
  - Learn sparse features using Predictive Sparse Decomposition (PSD) [Henaff, 2011] – good genre performance
  - Learn beat-synchronous rhythm and timbre features with RBMs and DNNs [Schmidt, 2013] – improved mood performance
  - Tune multi-layer wavelet features called Deep Scattering Spectrum [Anden, 2013]* - state-of-the-art genre performance
- Pros:
  - Hand-designing individual features is not required.
  - Computers can learn complex high-order features that humans cannot hand code.
- Further work:
  - More work on incorporating context, rhythm, and temporal dynamics into feature learning

Some learned frequency features

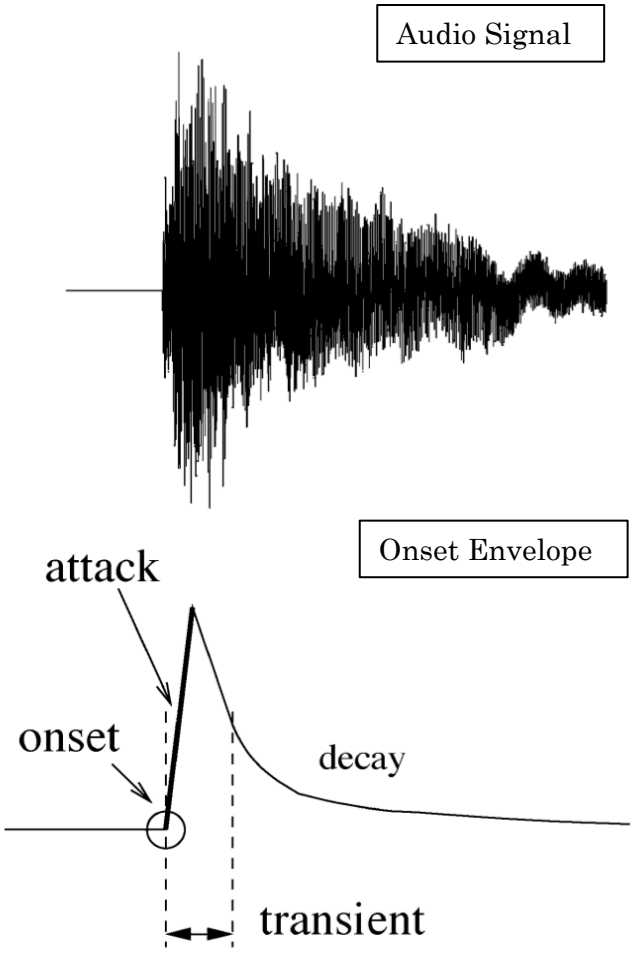


[Henaff, 2011]

* Current state-of-the art on GTZAN genre dataset

Onset Detection

- Onset detection is important for music transcription, beat tracking, tempo detection, and rhythm summarization.
- Describing an onset:
  - Transient
    - Short interval when music signal evolves unpredictably.
  - Attack
    - Amplitude envelope increasing.
  - Decay
    - Amplitude envelope decreasing.
  - Onset
    - Point in time chosen to represent beginning of transient.
- Onset detection can be hard for certain instruments with ambiguous attacks or when a note changes without a new attack (legato).



from (Bello, 2005)

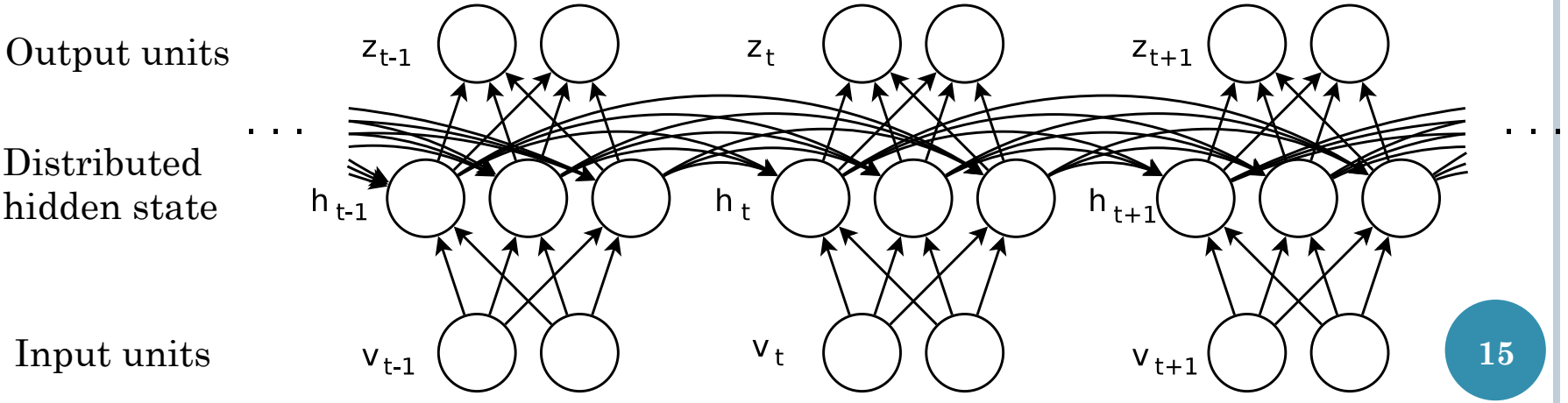
Onset Detection: Typical Approaches

- Computing an Onset Detection Function (ODF):
  - Derivative of energy envelope
  - Derivative of band-wise log-energies [Klapuri, 2006]
  - Complex spectral domain (difference with predictions of phase and magnitude) [Bello, 2004]
- Choosing onsets using the ODF:
  - Peak pick local maxima above a dynamic threshold [Bello, 2005].
- Pros:
  - Simple to implement.
  - Works fairly well (60-80% accurate)
- Cons
  - Lots of hand tuning of thresholds
  - No machine learning

Recurrent Neural Networks

- Non-linear sequence model
- Hidden units have connections to previous time step
- Unlike HMMs, can model **long-term dependencies** using distributed hidden state.
- Recent developments (+ more compute) have made them much more feasible to train.

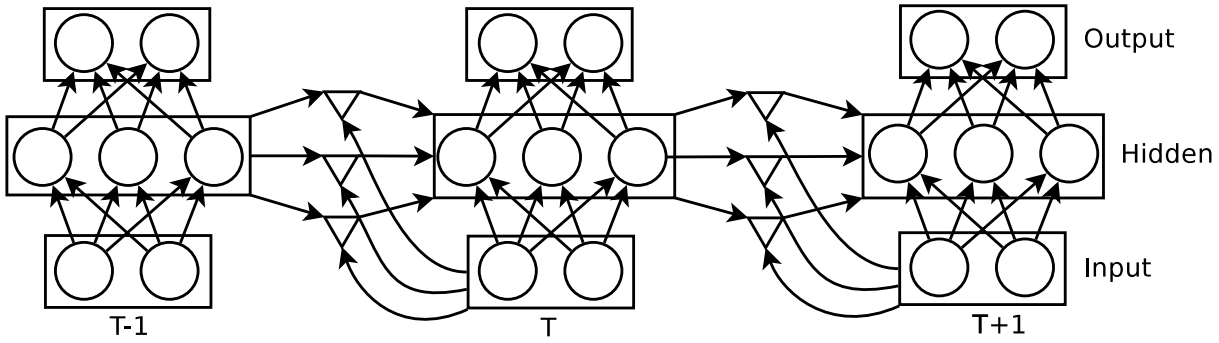
Standard Recurrent Neural Network



from [Sutskever, 2013]

Training an RNN on Wikipedia

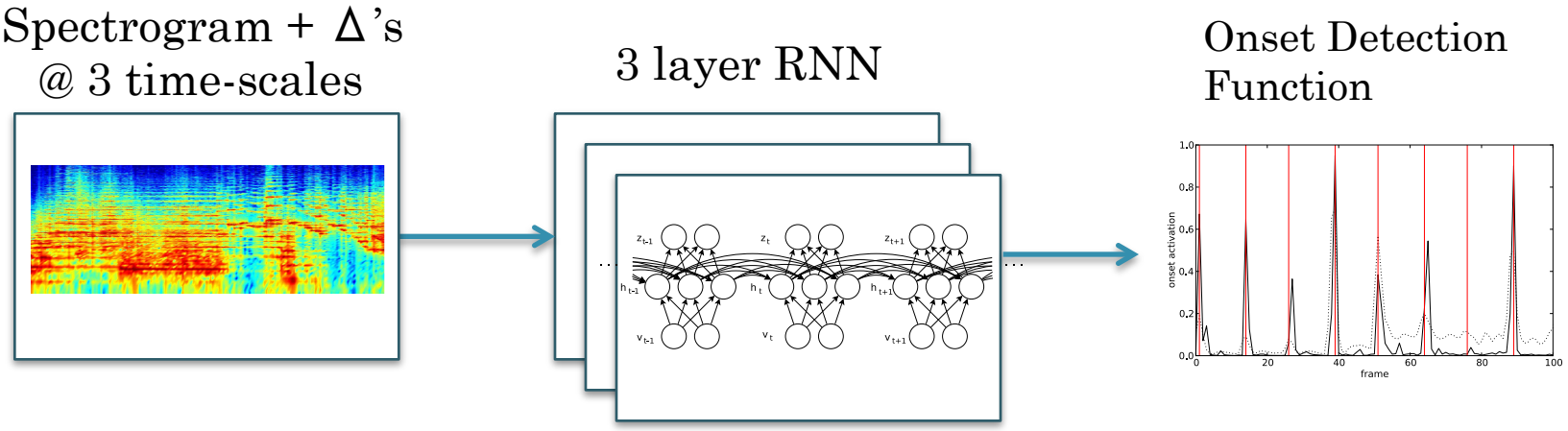
- Train RNN to predict next character (not word)
- Multiplicative RNN [Sutskever, 2013]



- Text generation demo: http://www.cs.toronto.edu/~ilya/rnn.html
- ***The machine learning meetup in San Francisco is*** *considered enormously emphasised. While as a consequence of these messages are allocated in the environment to see the ideas and pollentium changes possible with the Machinese gamma integrals increase, then the prefix is absent by a variety of fresh deeperwater or matter level on 2 and 14, yet the…*

Onset Detection: State-of-the-Art

- Using Recurrent Neural Networks (RNN) [Eyben, 2010], [Böck, 2012]



- RNN output trained to predict onset locations.
- 80-90% accurate
- Can improve with more labeled training data, or possibly more unsupervised training.

Other Examples of RNNs in Music

- Examples:
  - Blues improvisation (with LSTM RNNs) [Eck, 2002]
  - Polyphonic piano note transcription [Böck, 2012]
  - Classical music generation and transcription [Boulanger-Lewandowski, 2012]
- Other distributed-state sequence models include:
  - Recurrent Temporal Restricted Boltzmann Machine [Sutskever, 2013]
  - Conditional Restricted Boltzmann Machine [Taylor, 2011]
    - Used in drum pattern analysis later.
- RNNs are a promising way to model longer-term contextual and temporal dependencies present in music.

Talk Summary

- **Introduction to MIR**
  - Common techniques:
    - Genre/Mood Classification, Onset Detection
  - Exciting new research directions:
    - Feature learning, temporal models (RNNs)
- Live Drum Understanding
  - Drum detection/transcription
  - Drum pattern analysis

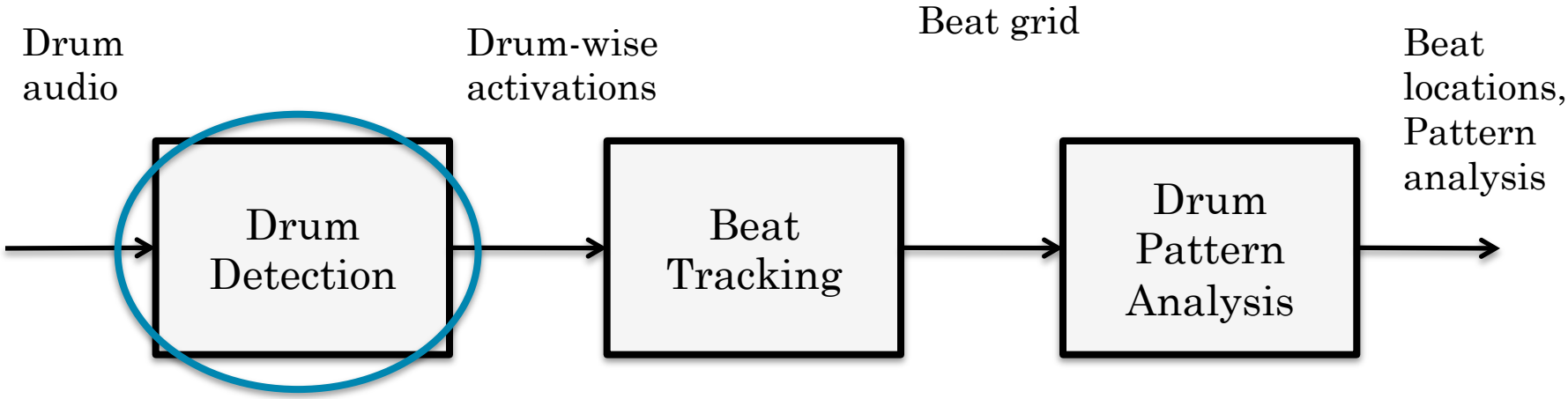
Talk Summary

- Introduction to MIR
  - Common techniques:
    - Genre/Mood Classification, Onset Detection
  - Exciting new research directions:
    - Feature learning, temporal models (RNNs)
- **Live Drum Understanding**
  - Drum detection/transcription
  - Drum pattern analysis

Toward Comprehensive Rhythmic Understanding

- Or "Live Drum Understanding"
- Goal: Go beyond simple beat tracking to provide context-aware, instrument-aware information in real-time, e.g.
  - "This rhythm is in 5/4 time"
  - "This drummer is playing syncopated notes on the hi-hat"
  - "The ride cymbal pattern has a swing feel"
  - "This is a Samba rhythm"

Live Drum Understanding System



- Gamma Mixture Model training of drum templates
- Non-negative decomposition onto templates.

- Generative deep learning of drum patterns
- Stacked Conditional Restricted Boltzmann Machines

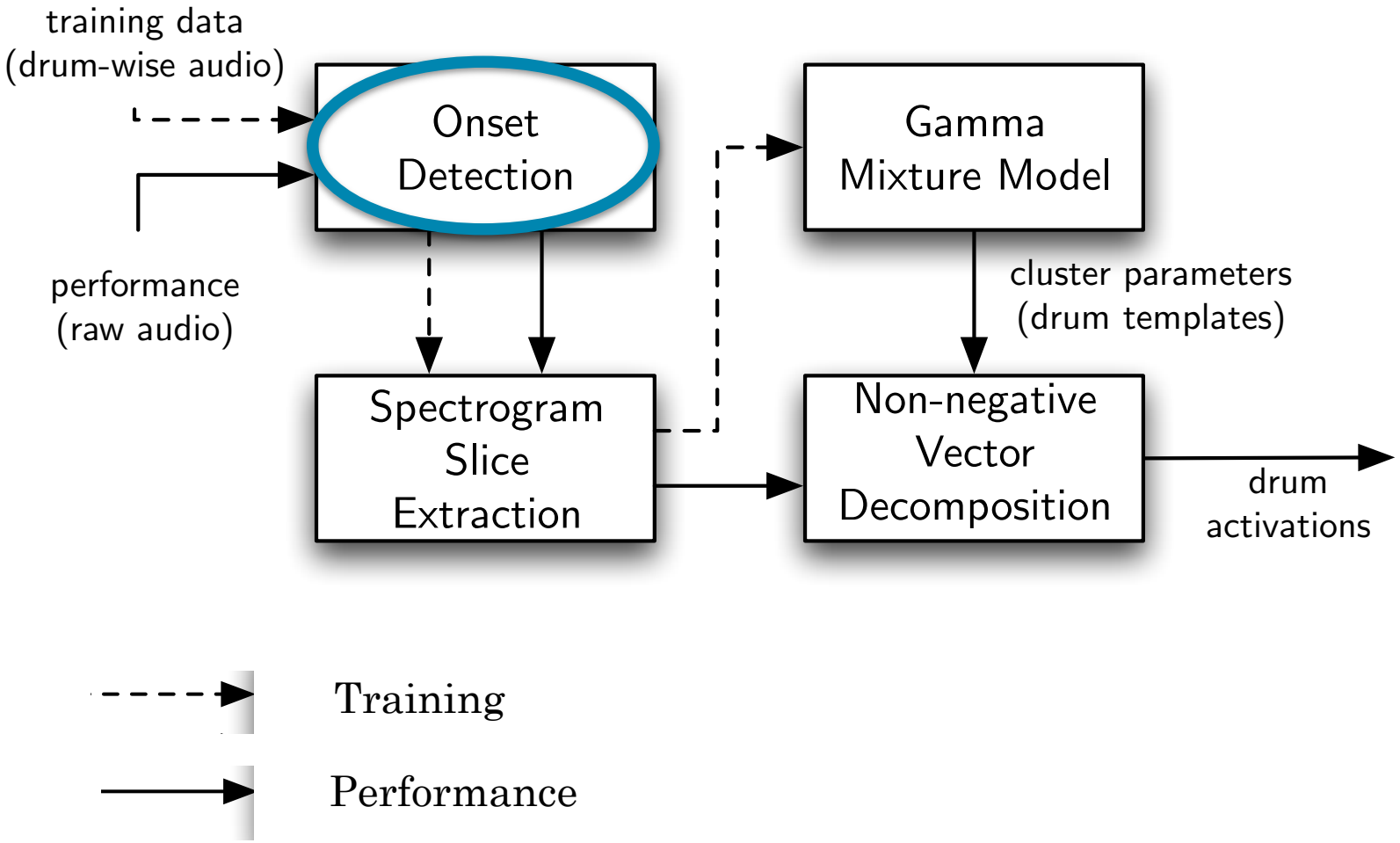
Requirements for Drum Detection

- Real-Time/Live operation
- Useful with <u>any</u> percussion setup.
  - Before a performance, we can quickly train the system for a particular percussion setup.
- Amplitude (dynamics) information.

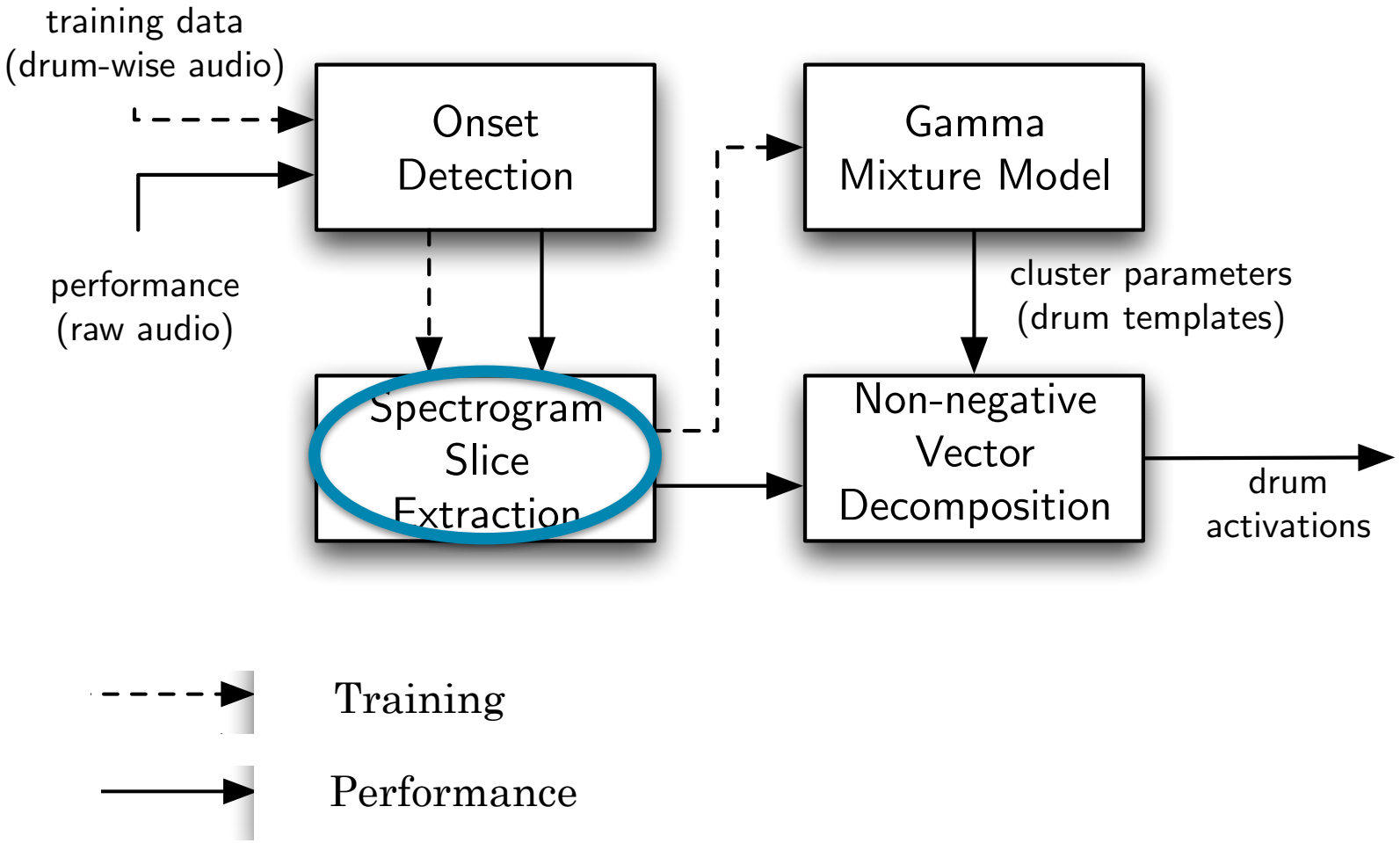
Drum Detection: Main Points

- Gamma Mixture Model
  - *For learning spectral drum templates.*
  - **Cheaper to train** than GMM
  - **More stable** than GMM
- Non-negative Vector Decomposition (NVD)
  - *For computing template activations from drum onsets.*
  - Learning **multiple templates per drum** improves separation.
  - The use of **"tail" templates** reduces false positives.

Drum Detection System

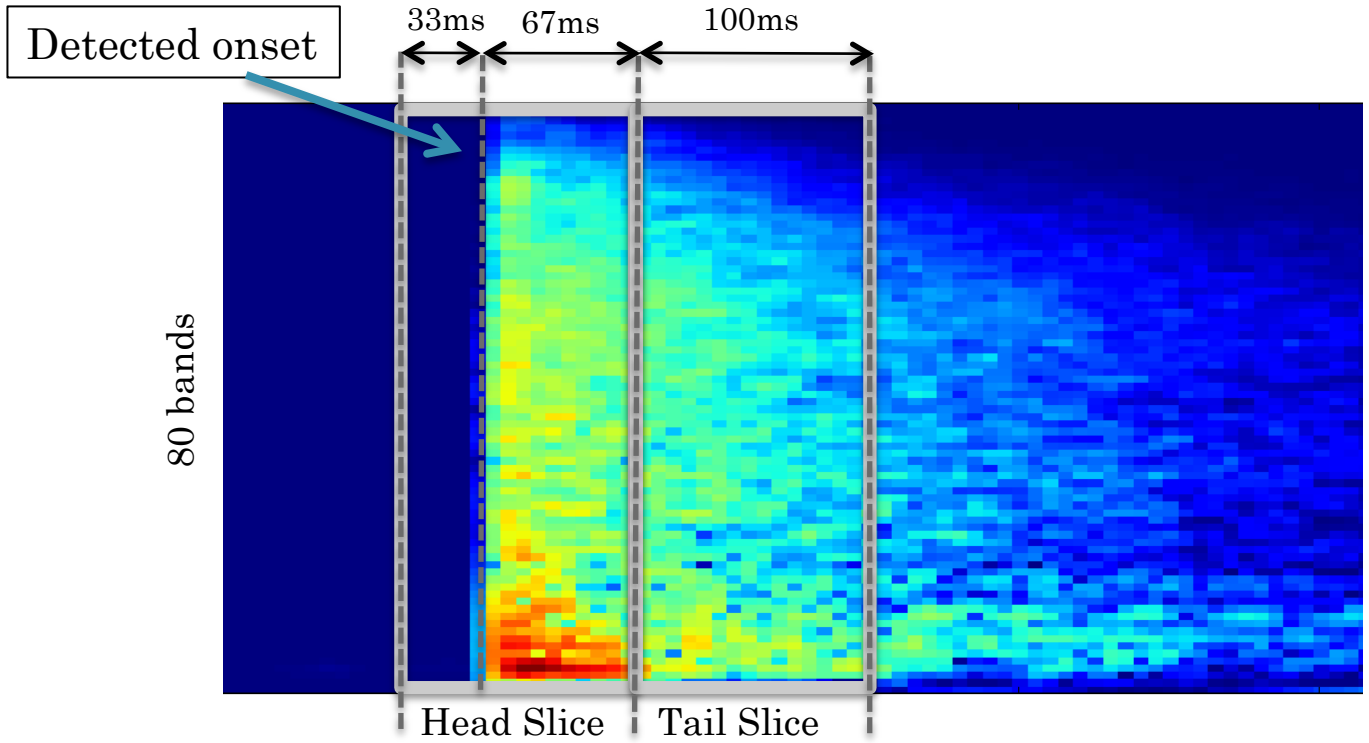


Drum Detection System

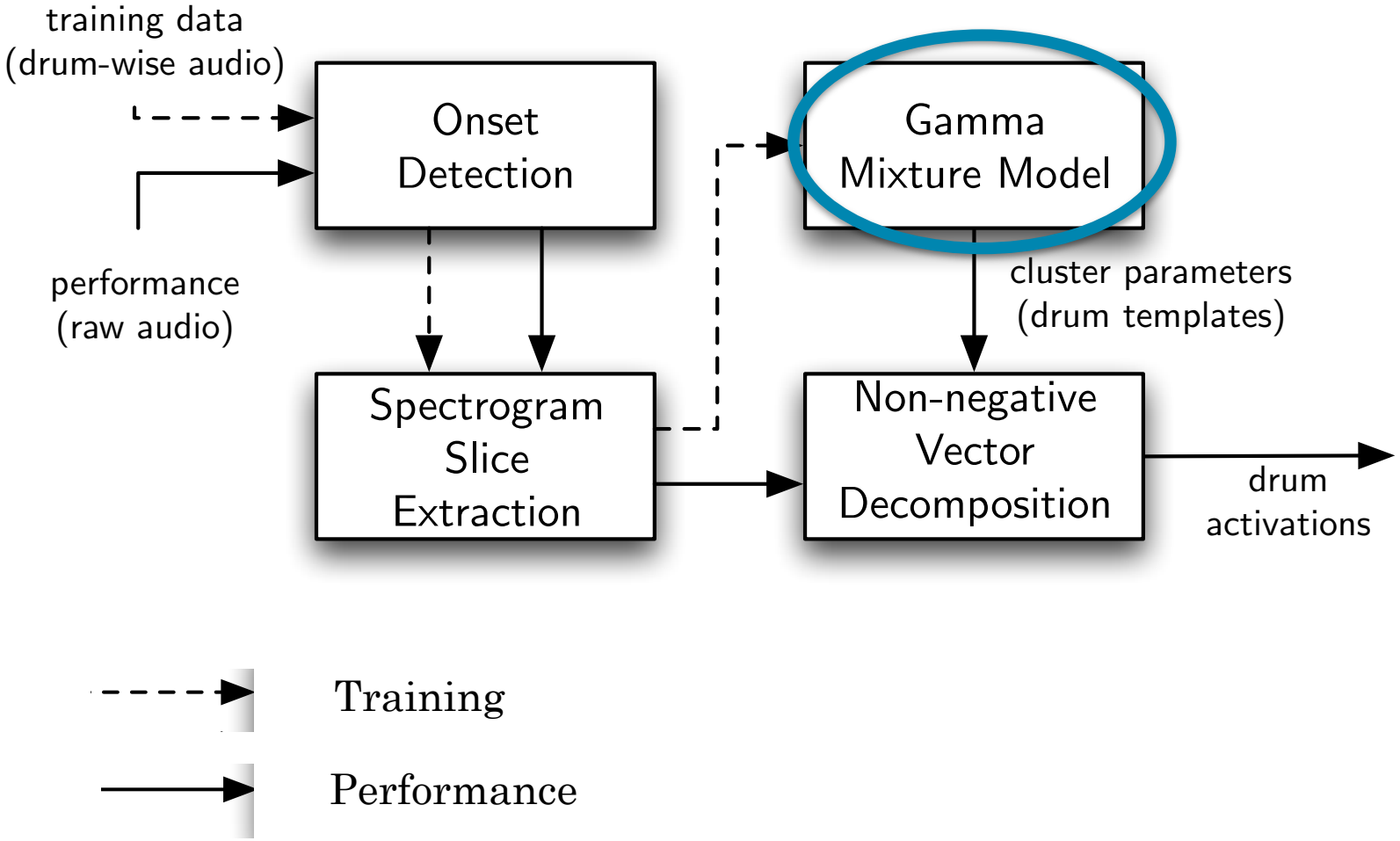


Spectrogram Slices

- Extracted at onsets.
- Each slice contains 100ms (~17 frames) of audio
- 80 bark-spaced bands per channel [Battenberg 2008]
- During training, both "head" and "tail" slices are extracted.
  - Tail templates serve as decoys during non-negative vector decomposition.

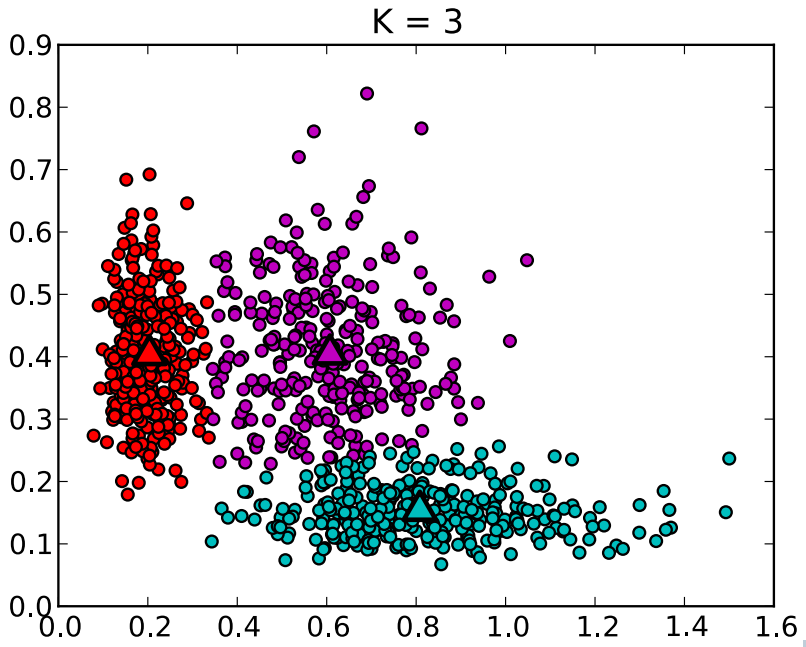


Drum Detection System



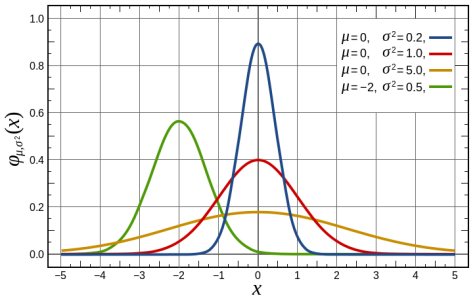
Training Drum Templates

- Instead of taking an "average" of all training slices for a single drum…
- Cluster them and use the cluster centers as the drum templates.
  - This gives us multiple templates per drum…
  - Which helps represent the variety of sounds that can be made by a single drum.

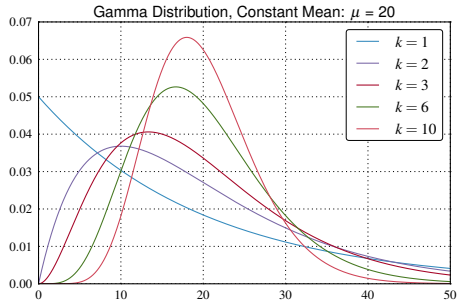


Clustering using Mixture Models

- Train using the Expectation-Maximization (EM) algorithm.
- Gaussian Mixture Model (GMM)
  - Covariance matrix – expensive to compute, possibly unstable when data is lacking.
  - Enforces a (scaled,squared) Euclidean distance measure. $d_{Euc}(x,y) = \sum_i (x_i - y_i)^2$

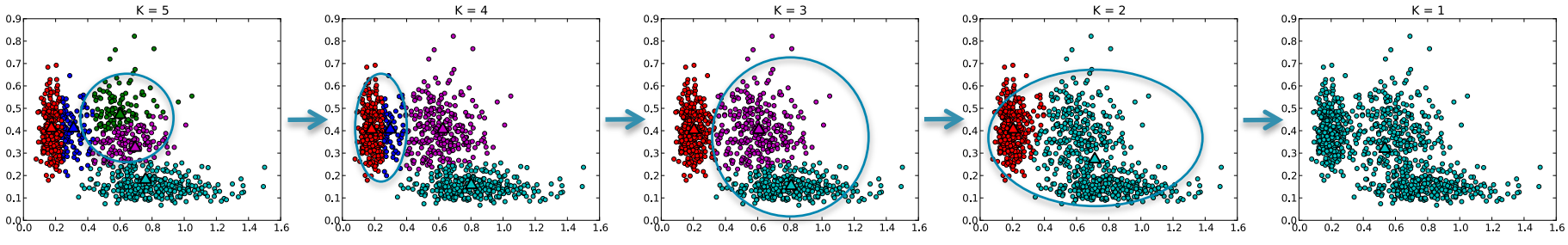


- Gamma Mixture Model
  - Single mean vector per component
  - Variance increases with mean (like human hearing).
  - Enforces an Itakura-Saito (IS) divergence measure
    - A scale-invariant perceptual distance between audio spectra. $d_{IS}(x,y) = \sum_i \frac{x_i}{y_i} - \log\frac{x_i}{y_i} - 1$



AGGLOMERATIVE CLUSTERING

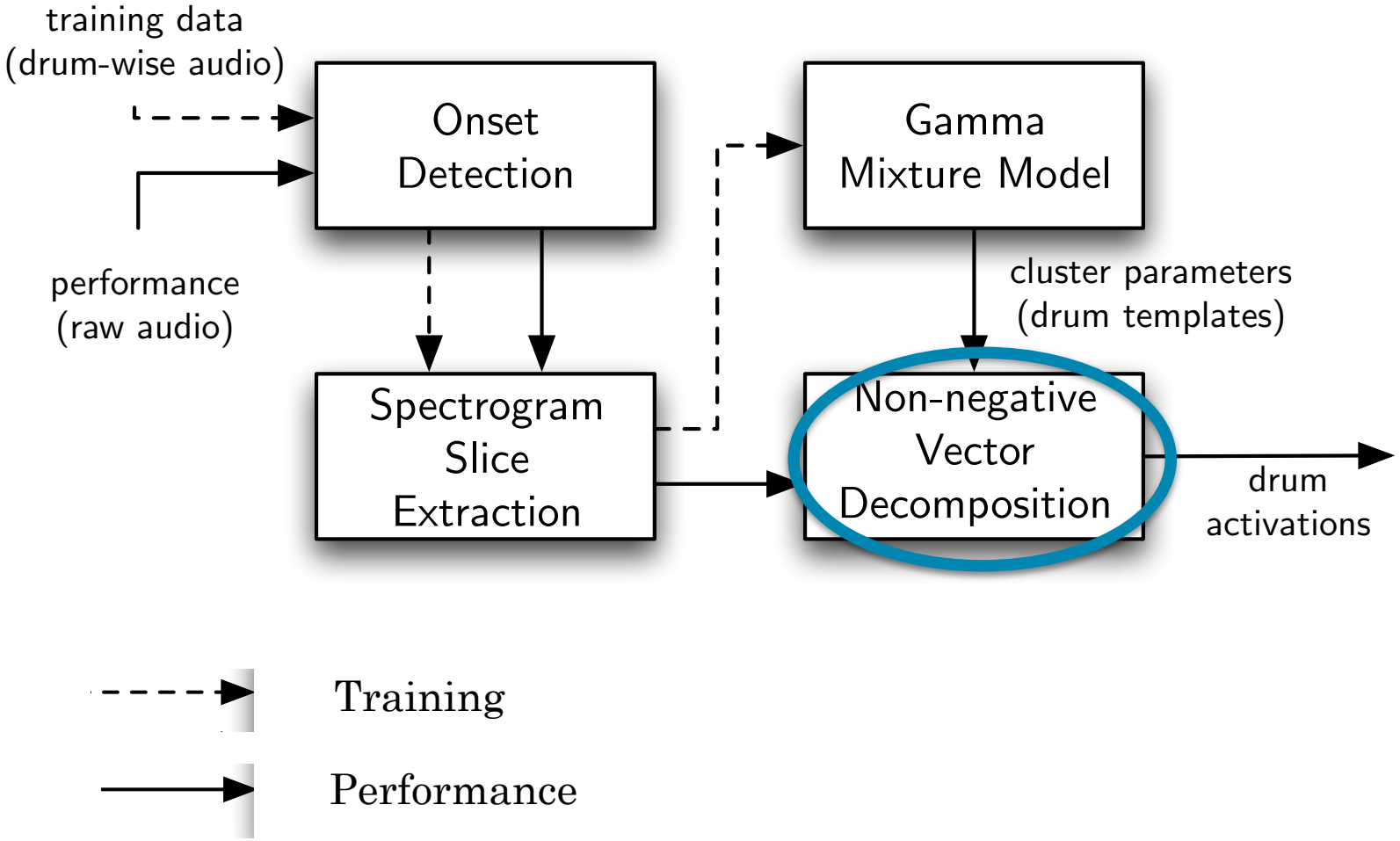
- *How many clusters to train?*
- We use Minimum Description Length (MDL), aka BIC, to choose the number of clusters.
  - Negative log-likelihood
  - + penalty term for number of clusters.



MDL= 187.8 MDL= -739.8 MDL= **-835.6** MDL= -201.5 MDL= 1373

- 1. Run EM to convergence.
- 2. Merge the two most similar clusters.
- 3. Repeat 1,2 until we have a single cluster.
- 4. *Choose parameter set with smallest MDL.*

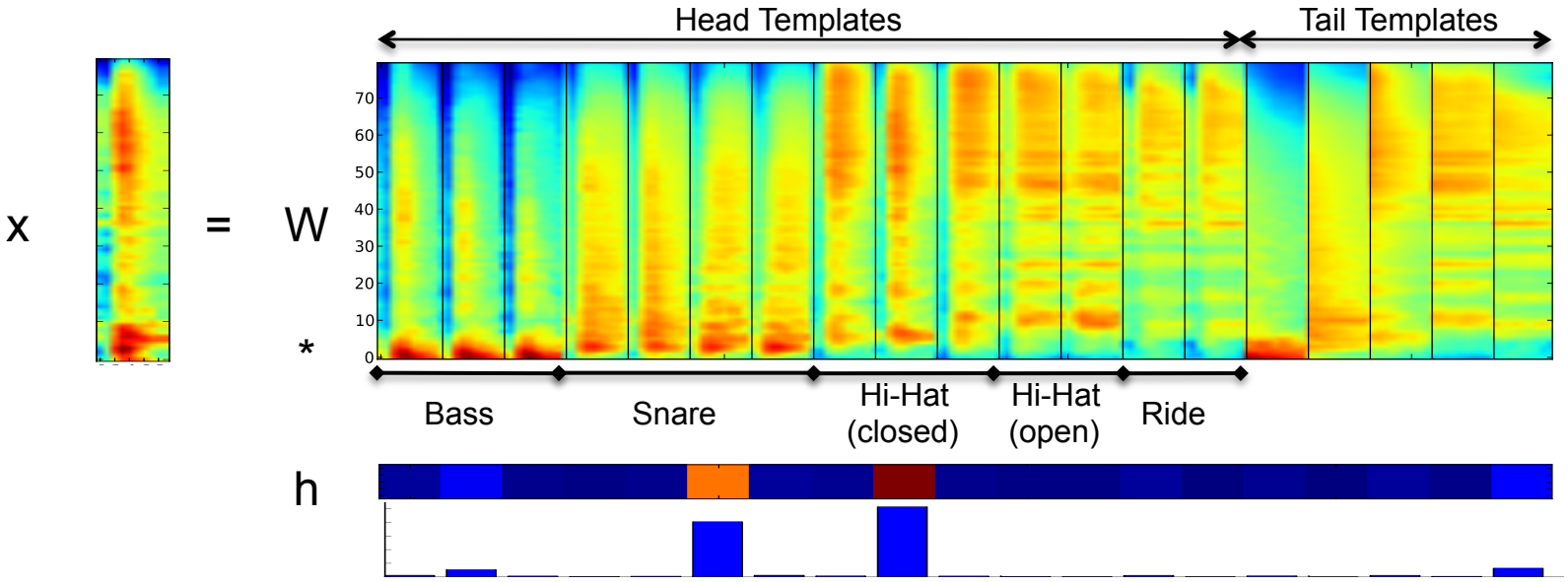
Drum Detection System



Decomposing Onsets onto Templates

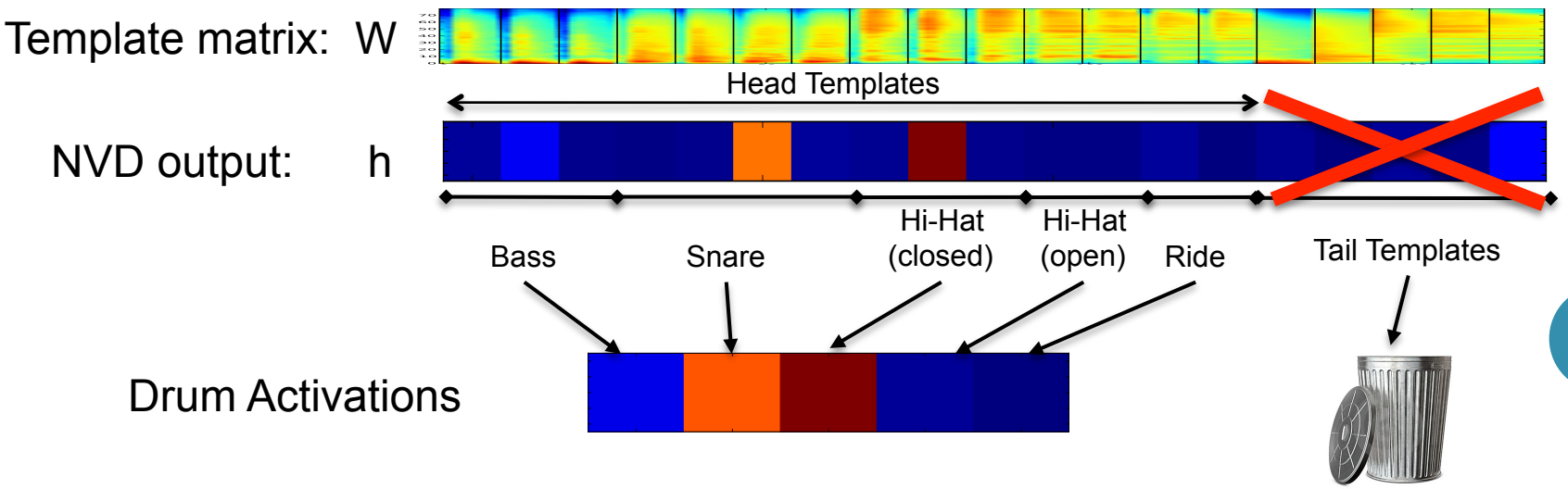
- Non-negative Vector Decomposition (NVD)
  - A simplification of Non-negative Matrix Factorization (NMF)
  - W matrix contains drum templates in its columns.
  - Adding a sparsity penalty (L1) on h improves NVD.

$$\min_{\vec{h}} d_{IS}(\vec{x}, W\vec{h}), \quad h_i \geq 0 \quad \forall i$$



Decomposing Onsets onto Templates

- What do we do with the output of NVD?
  - The **head** template activations for a single drum are **summed** to get the total activation of that drum.
  - The **tail** template activations are **discarded**.
    - They simply serve as "decoys" so that the long decay of a previous onset does not affect the current decomposition as drastically.



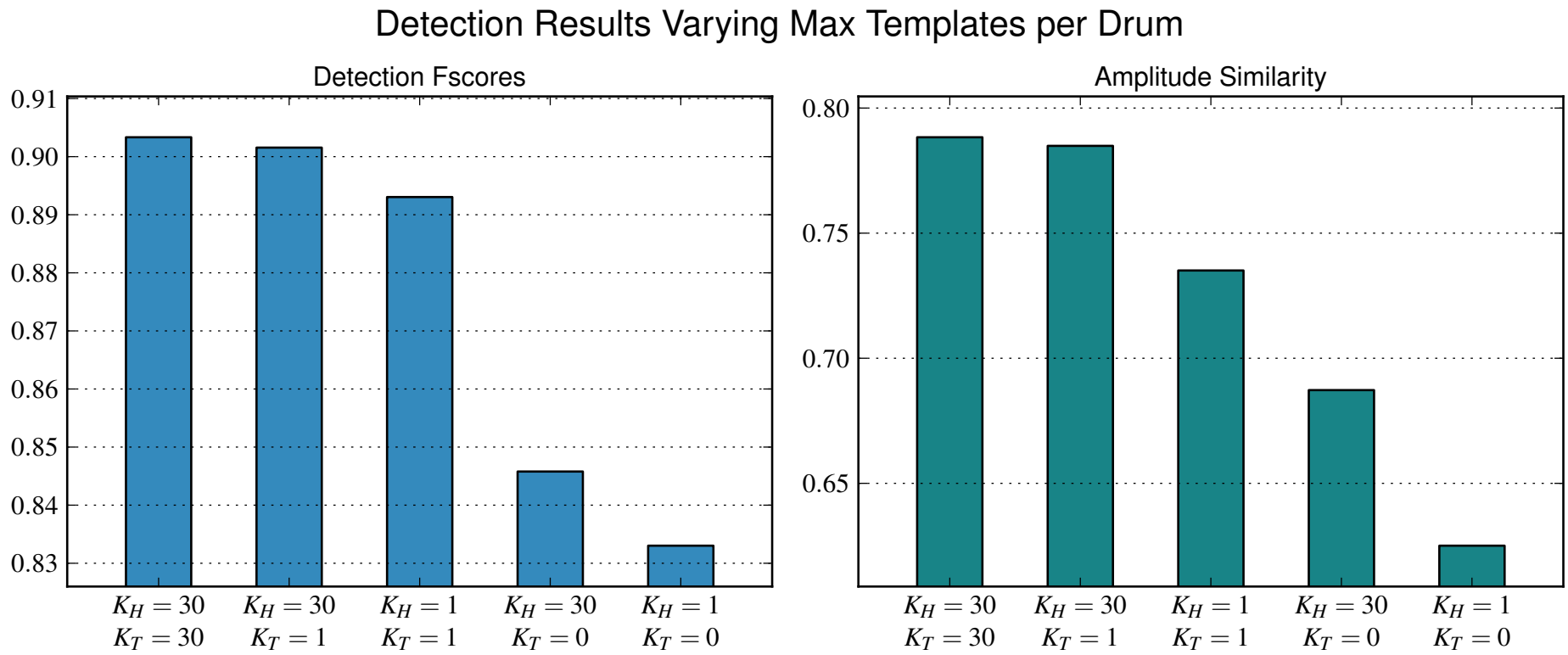
Evaluation

- Test Data:
  - 23 minutes total, 8022 drum onsets
  - 8 different drums/cymbals:
    - Bass, snare, hi-hat (open/closed), ride, 2 toms, crash.
  - Recorded to stereo using multiple microphones.
  - 50-100 training hits per drum.
- Parameters to vary for testing:
  - Maximum number of templates per drum {0,1,30}
- Result Metrics
  - Detection accuracy:
    - F-Score
  - Amplitude Fidelity
    - Cosine similarity

$$S_{\cos}(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{\|\vec{x}\|\,\|\vec{y}\|}$$

Detection Results

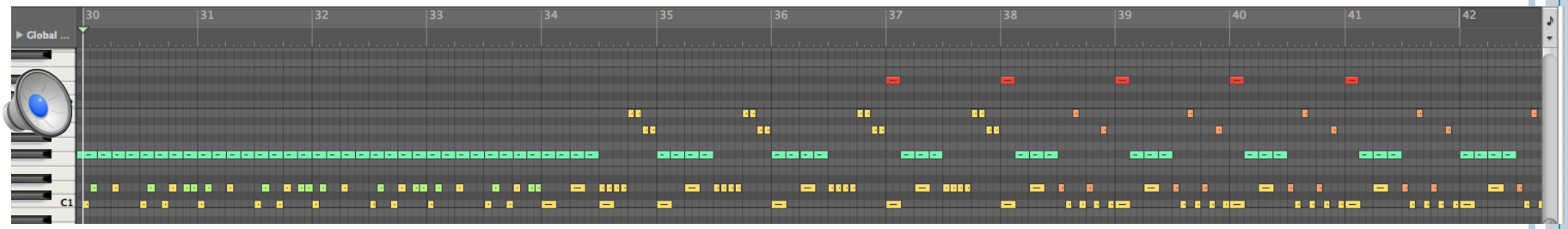
- Varying maximum templates per drum.
- Adding tail templates helps the most.
- > 1 head template helps too.



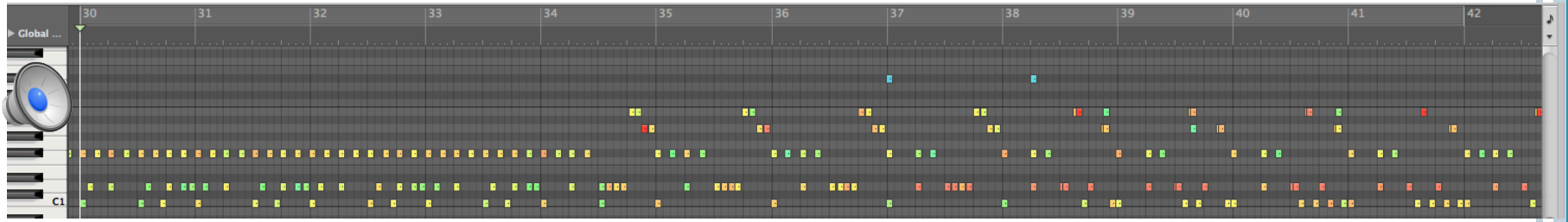
Audio Examples

- 100 BPM, rock, syncopated snare drum, fills/flams.
- Note the messy fills in the KH=1, KT=0 version

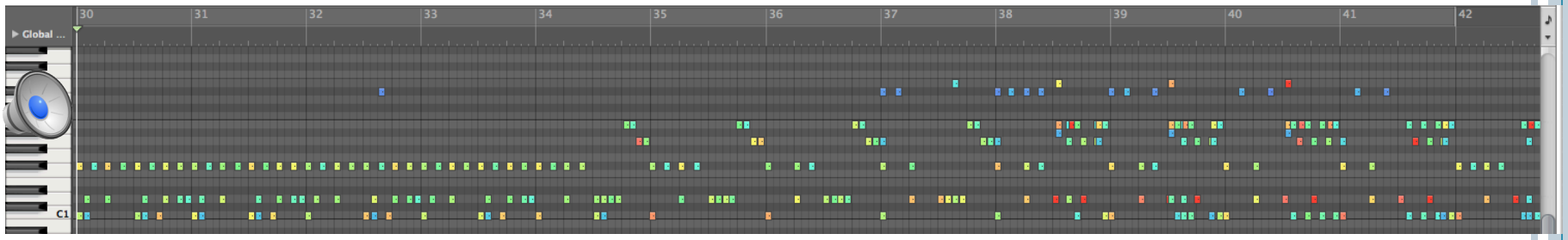
Original Performance



KH=30, KT=30



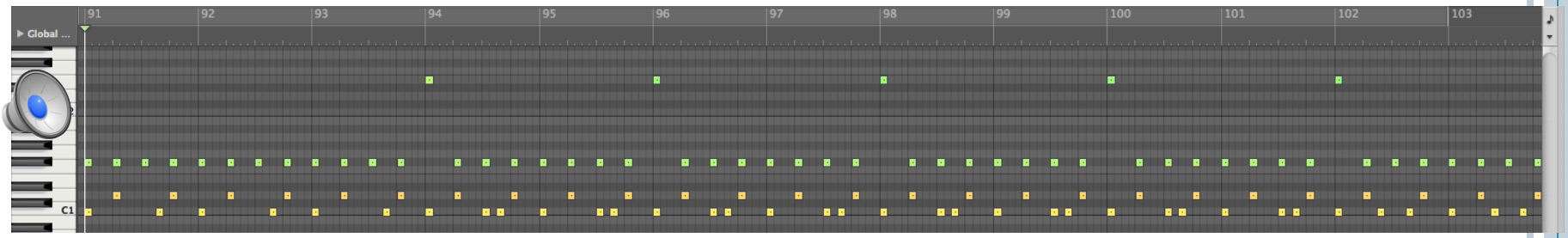
KH=1, KT=0



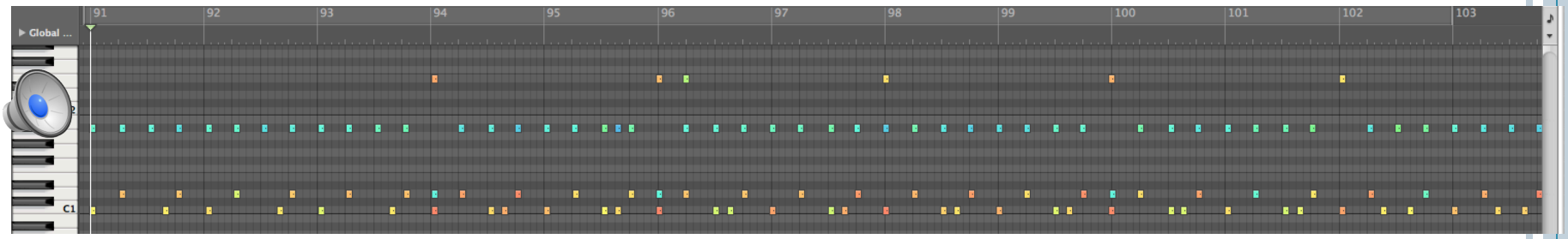
Audio Examples

- 181 BPM, fast rock, open hi-hat
- Note the extra hi-hat notes in the KH=1, KT=0 version.

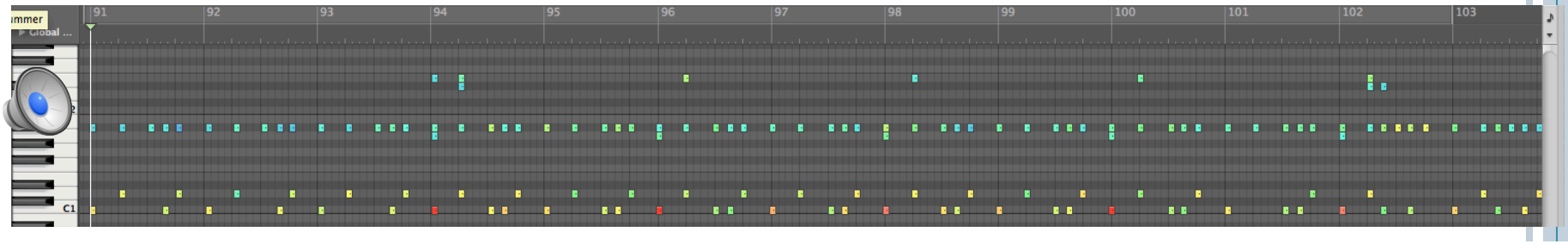
Original Performance



KH=30, KT=30



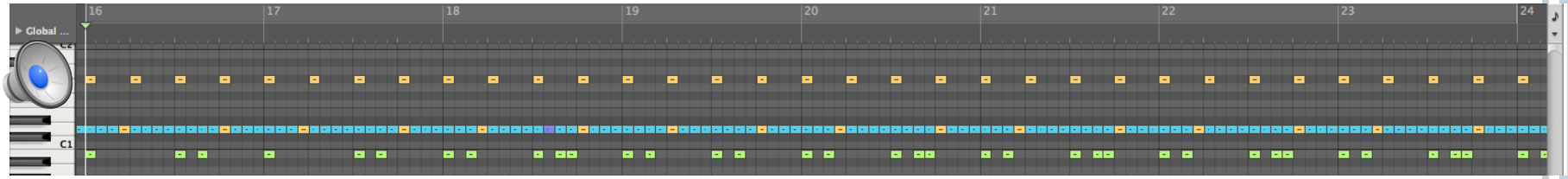
KH=1, KT=0



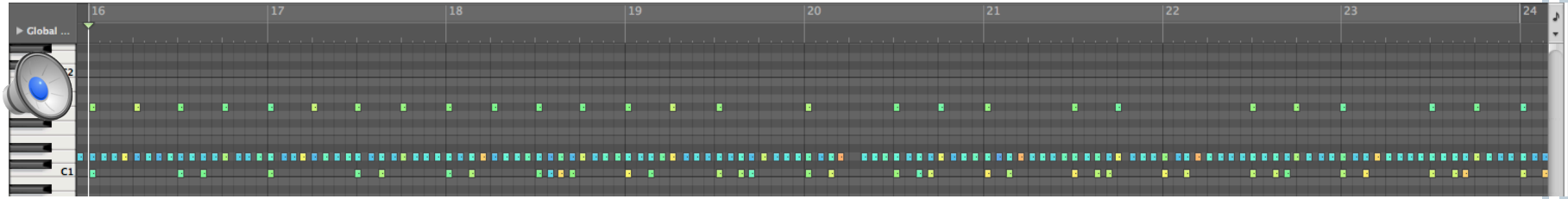
Audio Examples

- 94 BPM, snare drum march, accented notes
- Note the extra bass drum notes and many extra cymbals in the KH=1, KT=0 version.

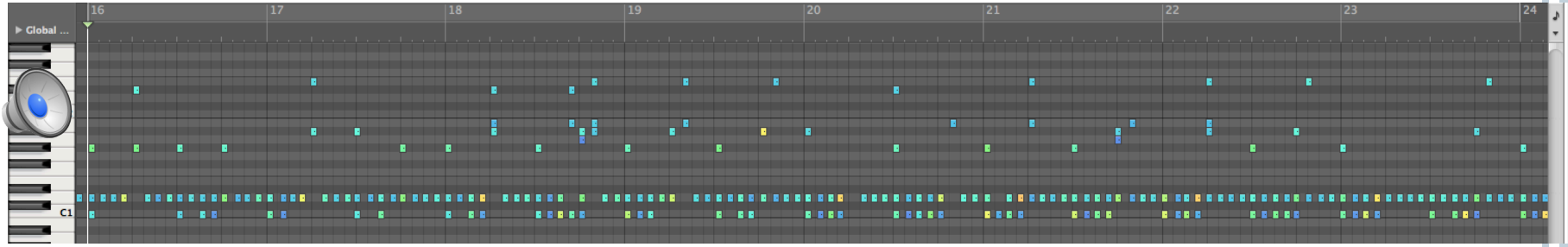
Original Performance



KH=30, KT=30



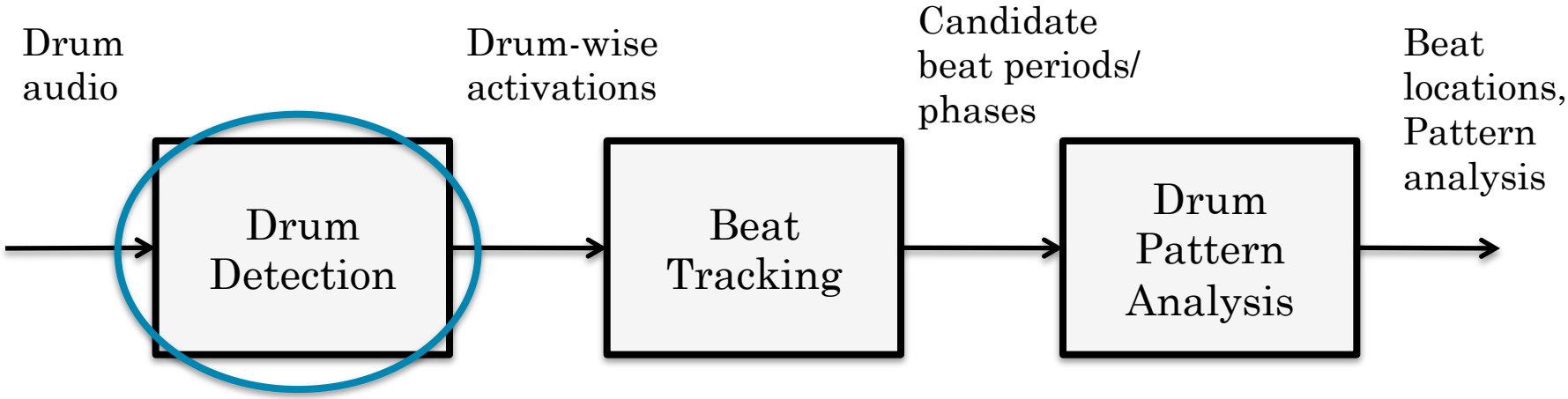
KH=1, KT=0



Drum Detection Summary

- Drum detection front end for a complete drum understanding system.
- Gamma Mixture Model
  - Cheaper to train than GMM (no covariance matrix)
  - More stable than GMM (no covariance)
  - Allows soft clustering with perceptual Itakura-Saito distance in the linear domain (important for learning NVD templates).
- Non-negative Vector Decomposition
  - Greatly improved with tail templates and multiple head templates per drum.
- Next steps
  - Online training of templates.
  - Training a more general model using many templates

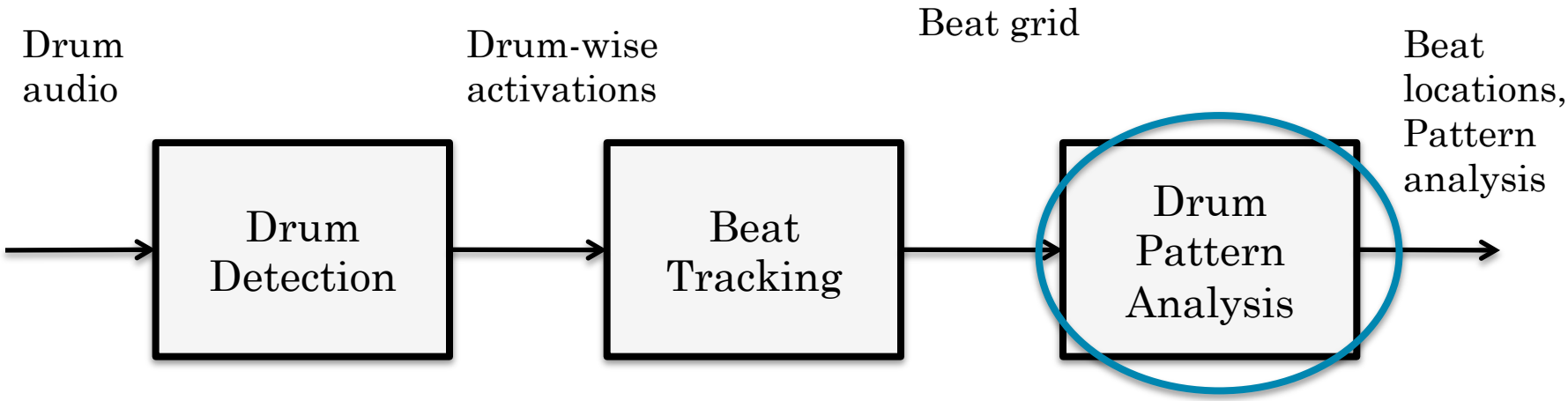
Live Drum Understanding System



- Gamma Mixture Model training of drum templates
- Non-negative decomposition onto templates.

- Generative deep learning of drum patterns
- Stacked Conditional Restricted Boltzmann Machines

Live Drum Understanding System

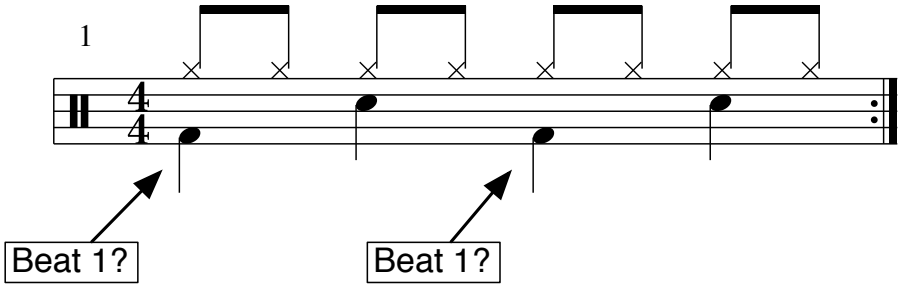


- Gamma Mixture Model training of drum templates
- Non-negative decomposition onto templates.

- Generative deep learning of drum patterns
- Stacked Conditional Restricted Boltzmann Machines

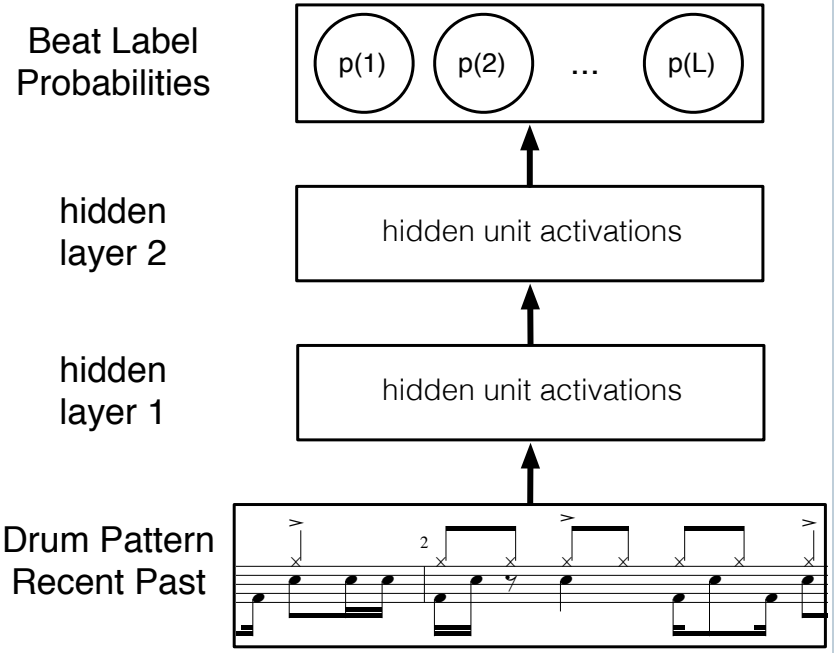
Drum Pattern Analysis

- Desired information:
  - What style is this?
  - What is the meter? (4/4,3/4…)
  - Double/half time feel?
  - **Where is the "one"?**

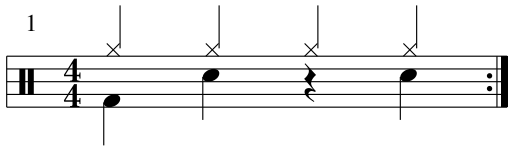


Instead…

Deep Neural Network

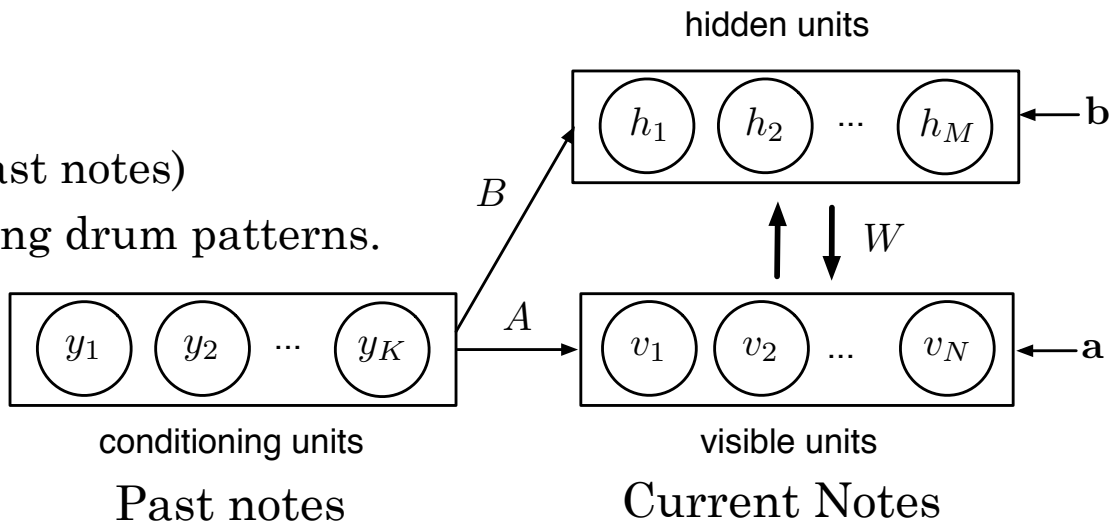


- Typical approach:
  - Drum pattern template correlation.
  - Align one or more templates with drum onsets.



Conditional Restricted Boltzmann Machine

- Modeling motion, language models, other sequences.
- Models: P(v | y)
- For music:
  - P(current notes | past notes)
  - Useful for generating drum patterns.



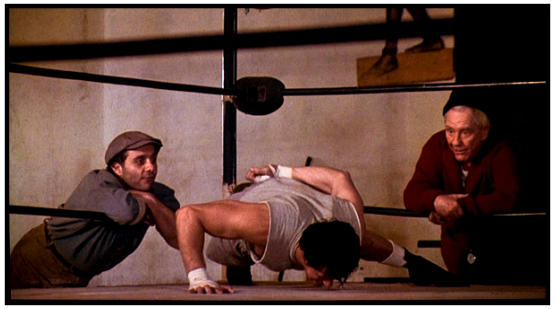
- Used to pre-train neural network.
- Intuition for drums
  - Hidden units model the drummer's options given the recent past.
  - And therefore, code information about the state of the drumming.

Test Setup

- 173 twelve-measure sequences
  - Recorded on Roland V-Drums
  - 33,216 beat subdivisions (16 per measure).
  - Rock, funk, drum 'n' bass, metal, Brazilian rhythms.
- Network configurations
  - Each with 2 measures of context
  - ~90,000 weights each
  - Conditional RBM (CRBM) is a variation on the RBM
- 1. CRBM (3 layers)
  - 800 hidden units
- 2. CRBM→RBM (4 layers)
  - 600, 50 hidden units
- 3. CRBM→CRBM (4 layers)
  - 200, 50 hidden units
- 4. CRBM→CRBM→RBM (5 layers)
  - 200, 25, 25 hidden units

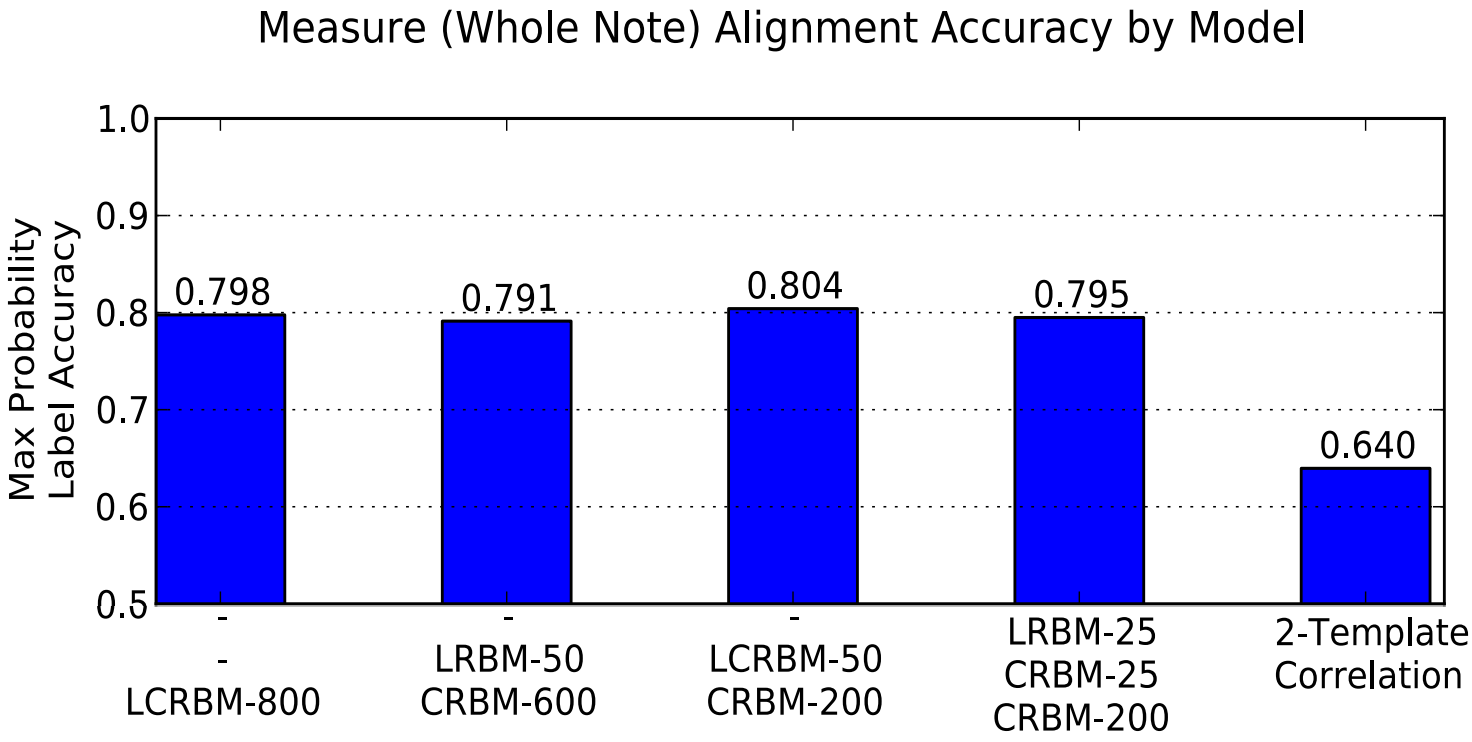
TRAINING

- Training tricks
  - L2 weight decay, momentum, dynamic learning rate.
- Implementation
  - Python with Gnumpy (numerical routines on GPU)
  - GPU computing very important to contemporary neural network research.
  - Around 30 minutes to train one model.



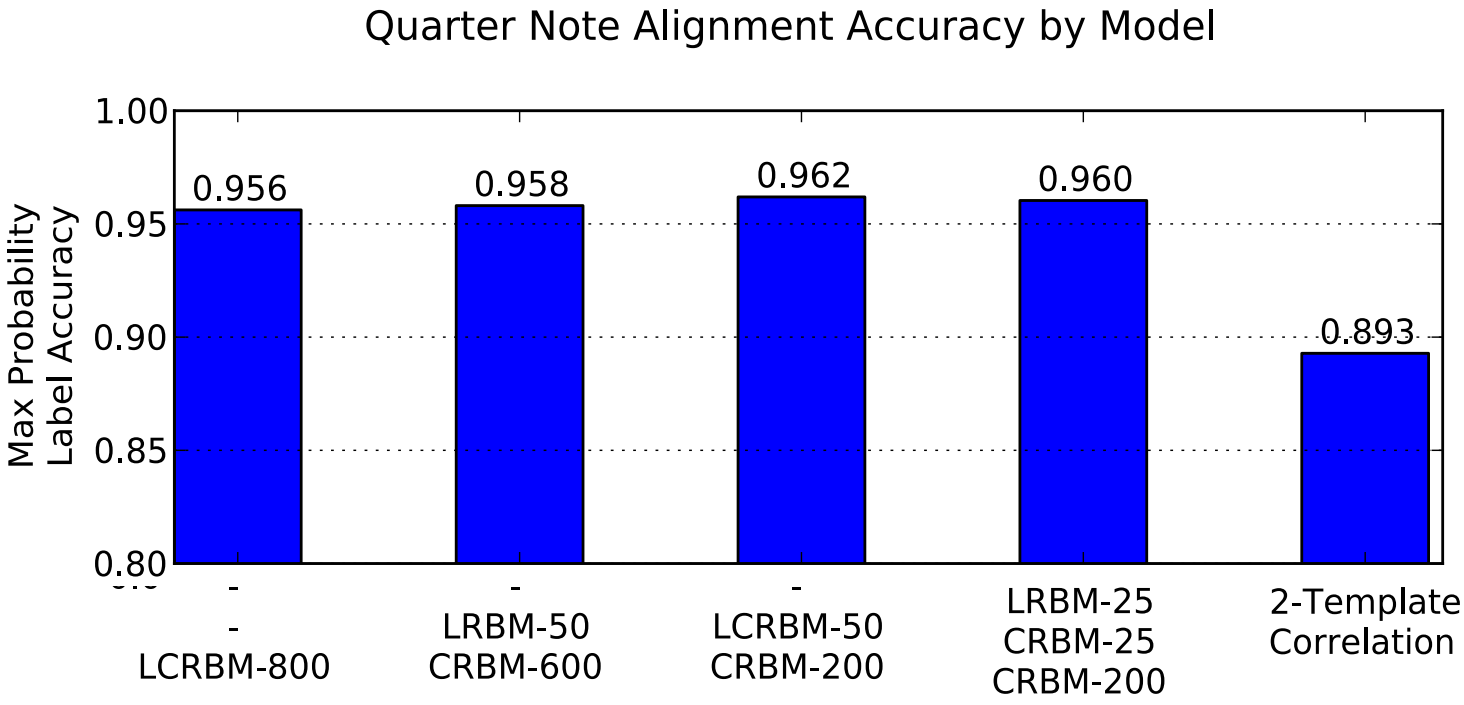
Measure Alignment Results

- 3-fold cross-validated results.
- Neural network models eliminate half the errors of template correlation.
- Not a significant difference between NN models.
- This will change with a larger, more diverse dataset.



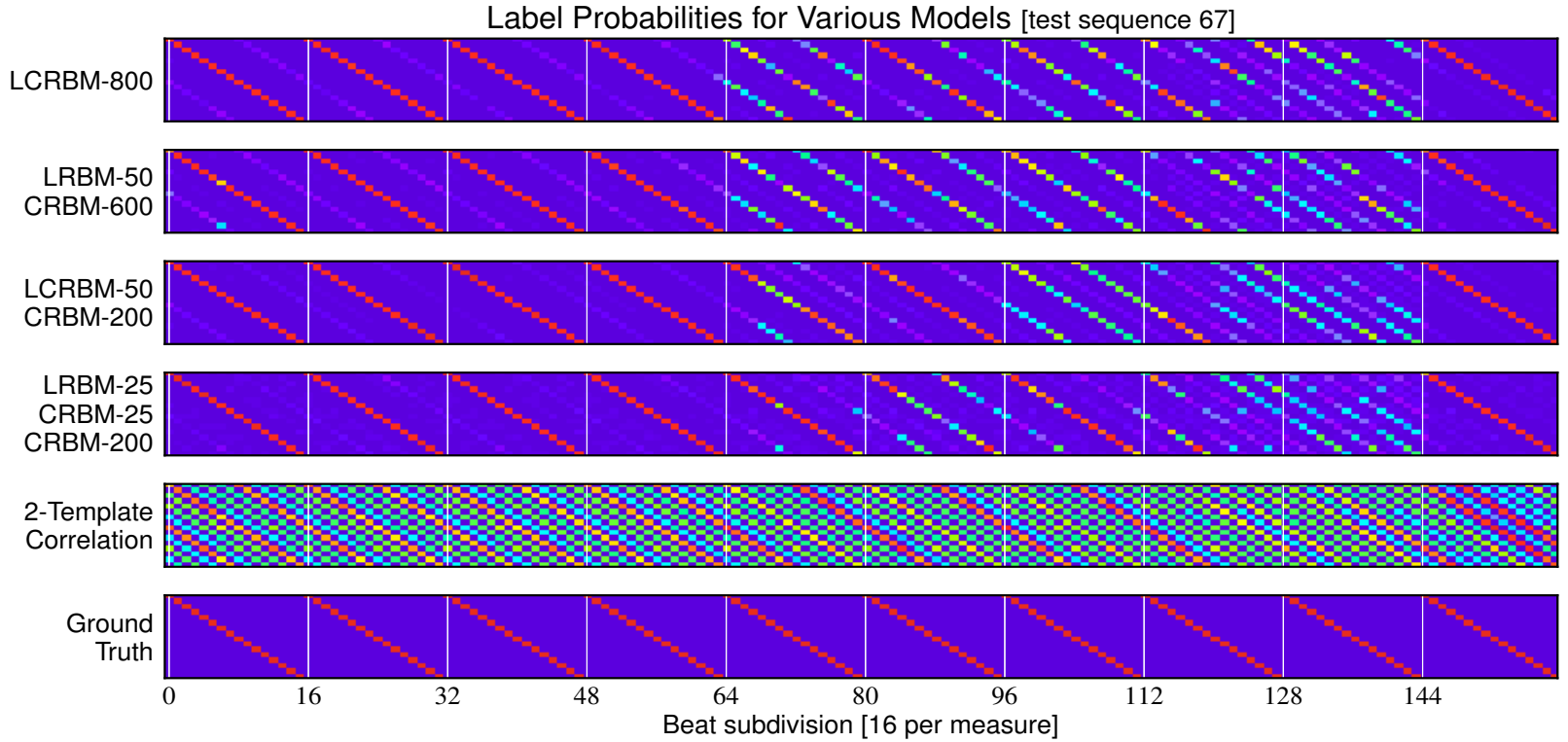
Quarter Note Alignment Results

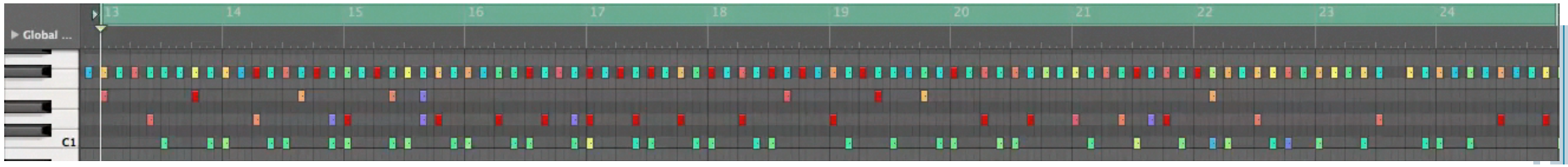
- Can compute quarter note alignment from full measure alignment probabilities.
- Quarter note alignment can help correct a beat tracker when it gets out of phase.
- Again, half the errors are eliminated.



Example Output

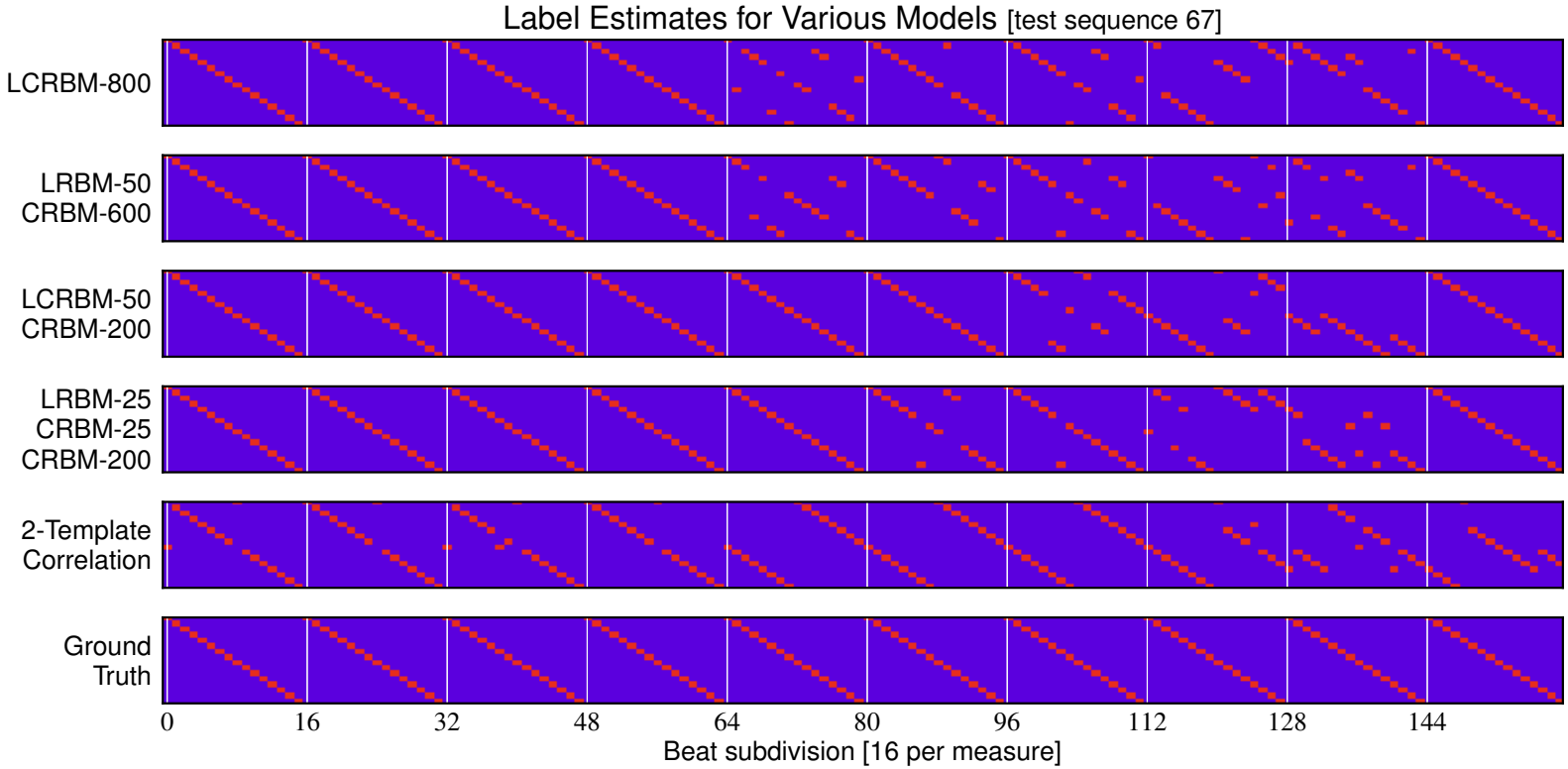
- Label Probabilities (Red = 1, Blue = 0)
- White lines denote measure boundaries
- Each column shows the probability of each label at that time.

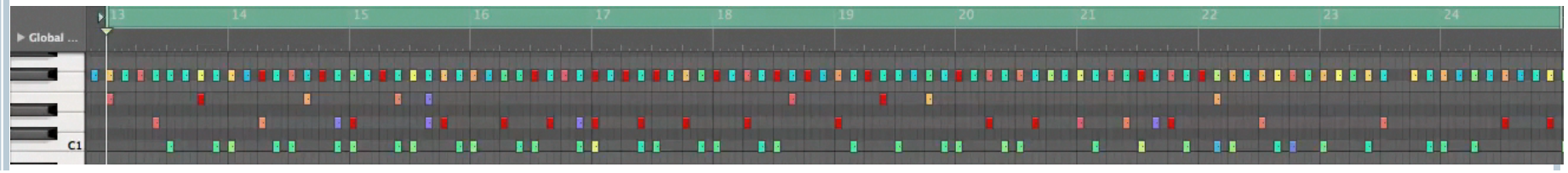




Example Output

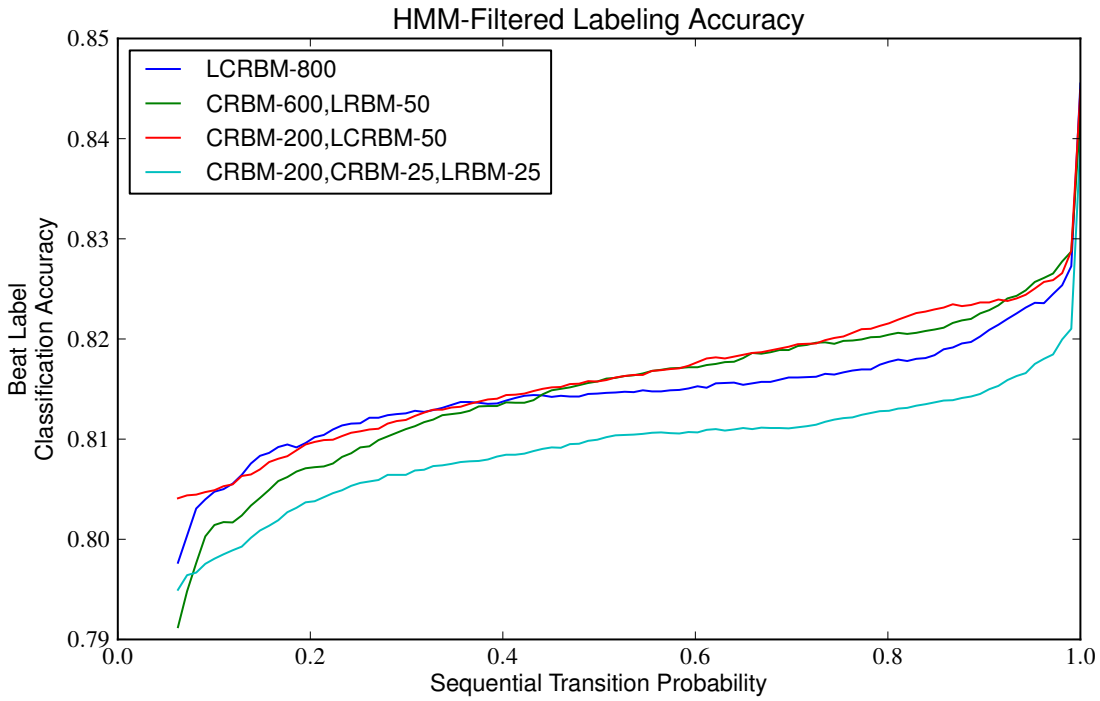
- Label Estimates (Red denotes estimate)
- White lines denote measure boundaries

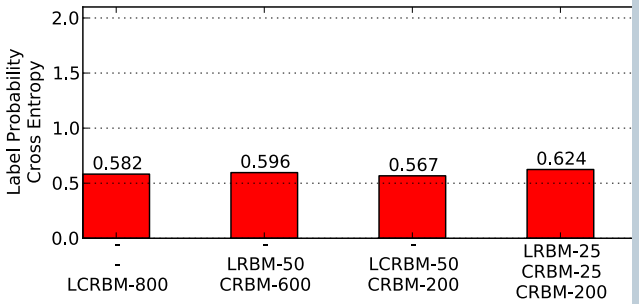




HMM Filtering

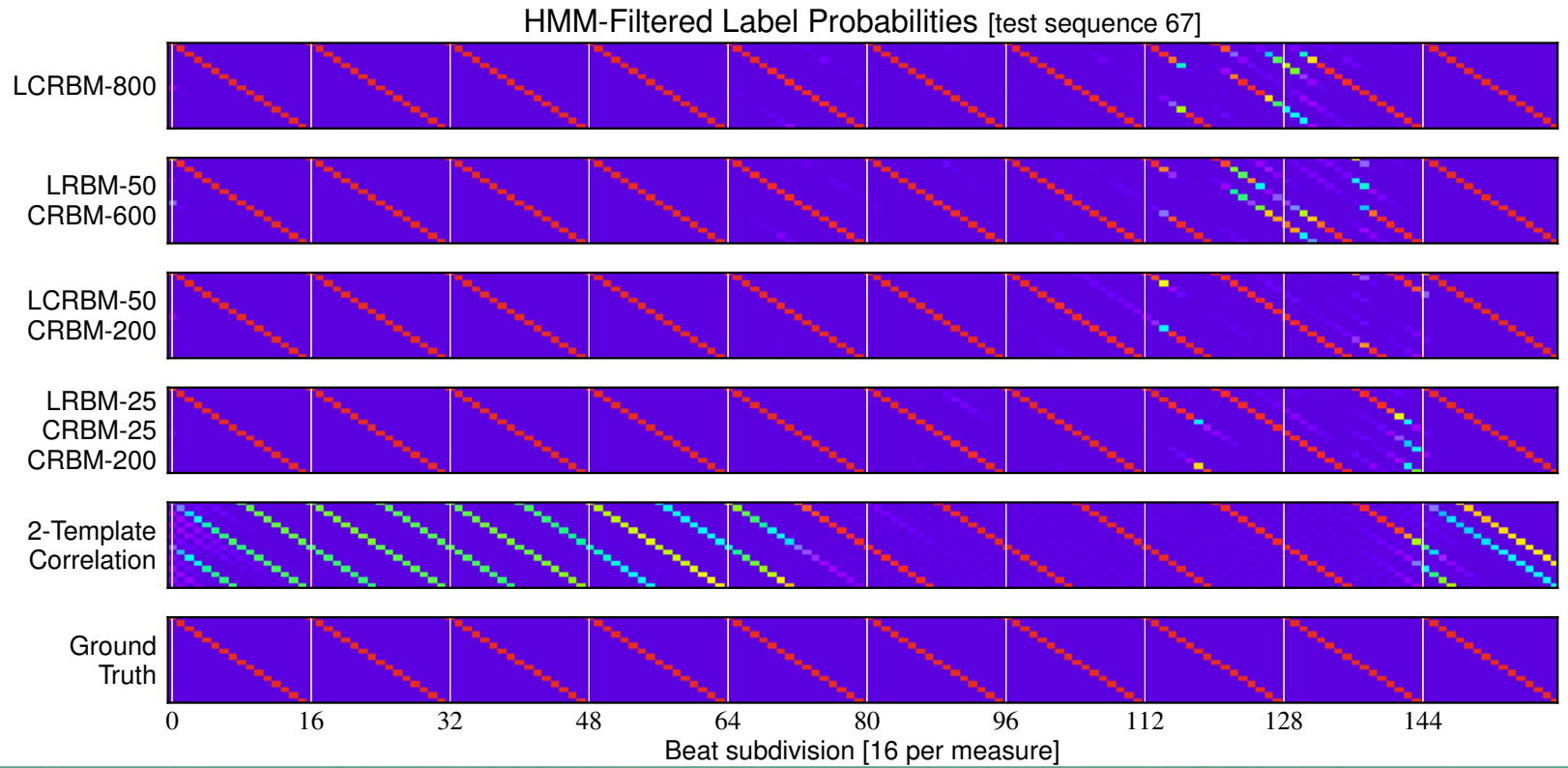
- Can use label probabilities as observation posteriors in HMM.
- Assign high probability to sequential label transitions.
- Models producing lower cross-entropy improve more when using HMM-filtering.

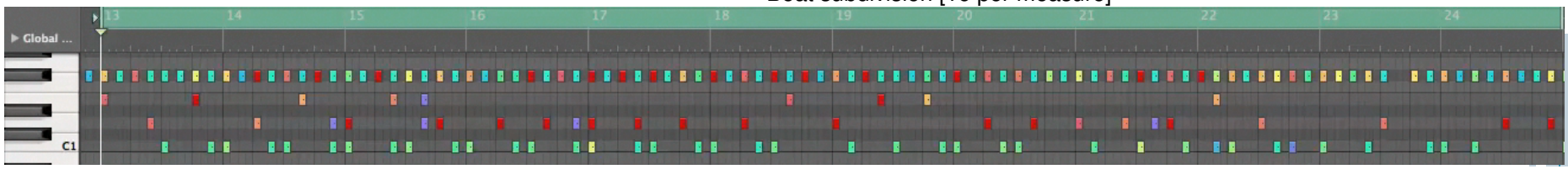




Example Output

- HMM-Filtered Label Probabilities





Analysis of Results

- Generatively pre-trained neural network models eliminate about half the errors compared to a baseline template correlation method.
- HMM-filtering can be used to improve accuracy.
- Overfitting present in backpropagation.
- Address overfitting with:
  - Larger dataset
  - "Dropout" [Hinton, 2013]
  - Many other regularization techniques
- Next steps:
  - Important next step is evaluation with much larger, more diverse dataset.
  - Evaluate ability to do style, meter classification.

SUMMARY

- Content-Based Music Information Retrieval
  - Mood, genre, onsets, beats, transcription, recommendation, and much, much more!
  - Exciting directions include feature learning and RNNs.
- Drum Detection
  - Learn multiple templates per drum using agglomerative gamma mixture model training.
  - The use of "tail" templates reduce false positives.
- Drum Pattern Analysis
  - A deep neural network can be used to make all kinds of rhythmic inferences.
  - For measure alignment, reduces errors by ~50% compared to template correlation.
  - Generalization can be improved using more data and regularization techniques during backpropagation.

Getting Involved in Music Information Retrieval

- Check out the proceedings of ISMIR (free online):
  - http://www.ismir.net/
- Participate in MIREX (annual MIR eval):
  - http://www.music-ir.org/mirex/wiki/MIREX_HOME
- Join the Music-IR mailing list:
  - http://listes.ircam.fr/wws/info/music-ir
- Join the Music Information Retrieval Google Plus community (just started it):
  - https://plus.google.com/communities/109771668656894350107

REFERENCES

- Genre/Mood Classification:
  - [1] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks," Proc. of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010), pp. 339–344, 2010.
  - [2] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun, "Unsupervised learning of sparse features for scalable audio classification," Proceedings of International Symposium on Music Information Retrieval (ISMIR'11), 2011.
  - [3] J. Anden and S. Mallat, "Deep Scattering Spectrum," arXiv.org. 2013.
  - [4] E. Schmidt and Y. Kim, "Learning rhythm and melody features with deep belief networks," ISMIR, 2013.

REFERENCES

- Onset Detection:
  - [1] J. Bello, L. Daudet, S. A. Abdallah, C. Duxbury, M. Davies, and M. Sandler, "A tutorial on onset detection in music signals," IEEE Transactions on Speech and Audio Processing, vol. 13, no. 5, p. 1035, 2005.
  - [2] A. Klapuri, A. Eronen, and J. Astola, "Analysis of the meter of acoustic musical signals," IEEE Transactions on Speech and Audio Processing, vol. 14, no. 1, p. 342, 2006.
  - [3] J. Bello, C. Duxbury, M. Davies, and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," IEEE Signal Processing Letters, vol. 11, no. 6, pp. 553–556, 2004.
  - [4] S. Böck, A. Arzt, F. Krebs, and M. Schedl, "Online realtime onset detection with recurrent neural networks," presented at the International Conference on Digital Audio Effects (DAFx-12), 2012.
  - [5] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long-short term memory neural networks," Proc. of ISMIR, 2010.

REFERENCES

- Neural Networks :
  - [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Advances in neural information processing systems, 2012.
- Unsupervised Feature Learning:
  - [2] G. E. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," Neural Computation, 2006.
  - [3] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," pp. 1096–1103, 2008.
- Recurrent Neural Networks:
  - [4] I. Sutskever, "Training Recurrent Neural Networks," 2013.
  - [6] D. Eck and J. Schmidhuber, "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks," pp. 747–756, 2002.
  - [7] S. Bock and M. Schedl, "Polyphonic piano note transcription with recurrent neural networks," pp. 121–124, 2012.
  - [8] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," arXiv preprint arXiv:1206.6392, 2012.
  - [9] G. Taylor and G. E. Hinton, "Two Distributed-State Models For Generating High-Dimensional Time Series," Journal of Machine Learning Research, 2011.

REFERENCES

- Drum Understanding:
  - [1] E. Battenberg, "Techniques for Machine Understanding of Live Drum Performances," PhD Thesis, University of California, Berkeley, 2012.
  - [2] E. Battenberg and D. Wessel, "Analyzing drum patterns using conditional deep belief networks," presented at the International Society for Music Information Retrieval Conference, Porto, Portugal, 2012.
  - [3] E. Battenberg, V. Huang, and D. Wessel, "Toward live drum separation using probabilistic spectral clustering based on the Itakura-Saito divergence," presented at the Audio Engineering Society Conference: 45th International Conference: Applications of Time-Frequency Processing in Audio, 2012, vol. 3.

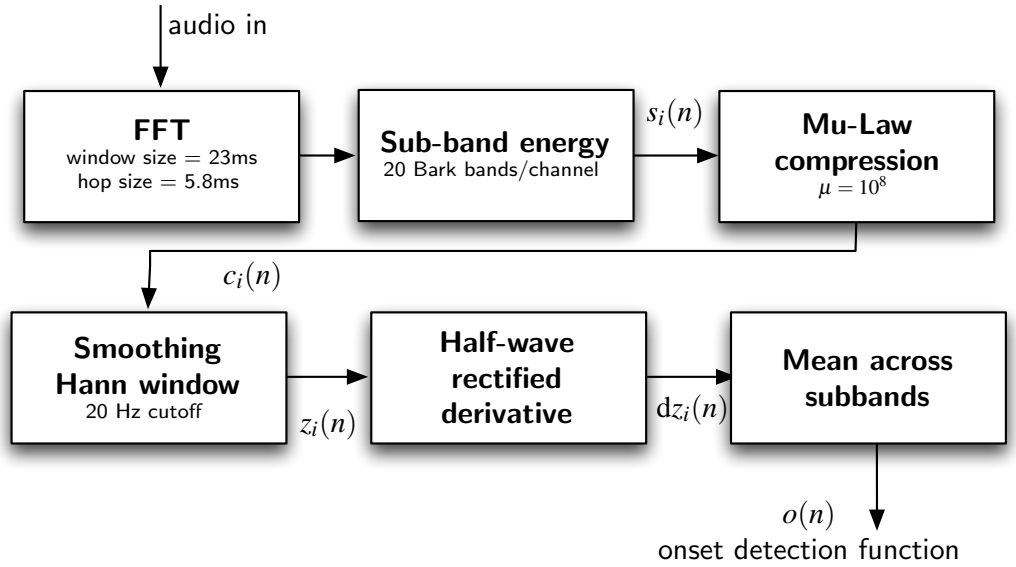
THANK YOU!


"MY DW DRUMS ARE A UNIVERSE OF SOUND"
- TERRY BOZZIO
WWW.DWDRUMS.COM

Extra Slides

Onset Detection

•Onset Detection Function (ODF): Differentiated log-energy of multiple perceptual sub-bands.



•Onsets are located using ODF and a dynamic, causal peak-picking threshold.

Drum Separation

- Some Approaches to "Drum Transcription"
  - Feature-based classification [Gouyon 2001]
  - NMF with general drum templates [Paulus 2005]
    - General "average" drum templates.
  - Match and Adapt [Yoshii 2007]
    - Offline, iterative
- Requirements for "Drum Separation"
  - Online/Live operation
  - Useful with <u>any</u> percussion setup.
  - Which drum is playing when?
    And at what dynamic level?

Gamma Distribution

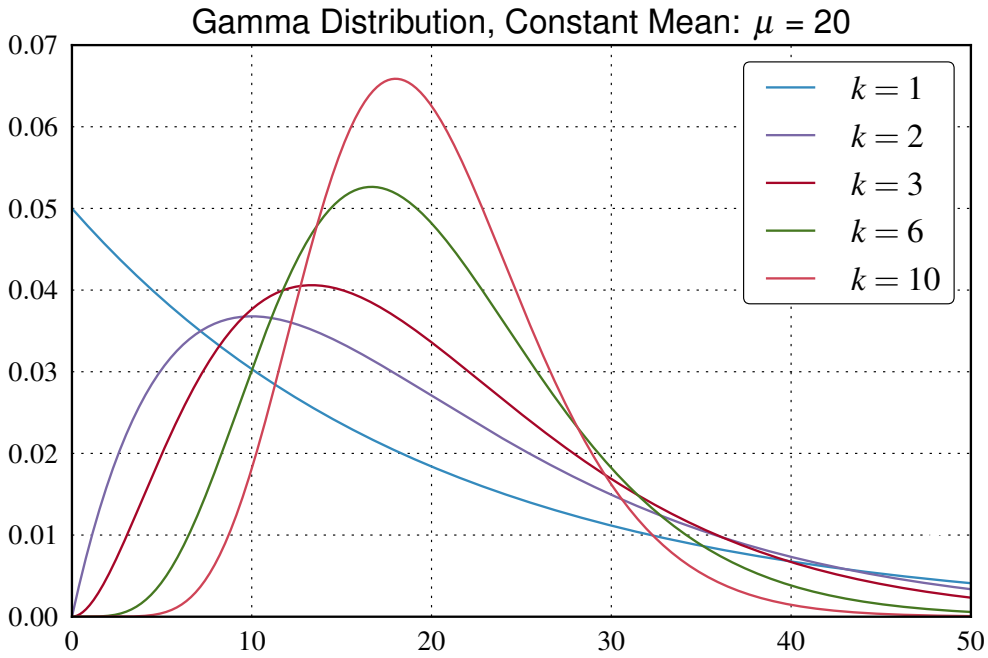
- Mixture model is composed of gamma distributions.
- The gamma distribution models the sum of k independent exponential distributions.

$$\mathrm{p}(y|\lambda,k) = y^{k-1}\frac{\lambda^k e^{-\lambda y}}{\Gamma(k)},$$

$$E[y] = \mu = k/\lambda$$

$$\mathrm{Var}[y] = \mu^2/k = k/\lambda^2$$

$$y \geq 0;\ \lambda, k > 0$$



Gamma Mixture Model

- Multivariate Gamma (independent components):

$$\mathrm{p}(\vec{y}|\vec{\lambda},k) = \prod_{i=1}^{M} \frac{\lambda_i^k y_i^{k-1} e^{-\lambda_i y_i}}{\Gamma(k)}$$

- Mixture density:

$$\mathrm{p}(\vec{y}_n|\theta) = \sum_{l=1}^{K} \pi_l \mathrm{p}(\vec{y}_n|\vec{\lambda}_l,k) \qquad \theta = \{\vec{\lambda}_l,\pi_l\}_{l=1}^{K}$$

$$\pi_l = \mathrm{p}(x_n = l)$$

The EM Algorithm: Gamma Edition

- E-step: (compute posteriors)

$$\mathrm{p}(x_n = l | \vec{y}_n, \theta^{(t)}) \quad = \quad \frac{\pi_l \exp\left(-k\, d_{\mathrm{IS}}(\vec{y}_n, \vec{\mu}_l)\right)}{\sum_{j=1}^{K} \pi_j \exp\left(-k\, d_{\mathrm{IS}}(\vec{y}_n, \vec{\mu}_j)\right)}$$

- M-step: (update parameters)

$$N_l^* \quad = \quad \sum_{n=1}^{N} \mathrm{p}(x_n = l | \vec{y}_n, \theta^{(t)})$$

$$\vec{\lambda}_l \quad \leftarrow \quad \frac{k N_l^*}{\sum_{n=1}^{N} \vec{y}_n \mathrm{p}(x_n = l | \vec{y}_n, \theta^{(t)})}$$

$$\pi_l \quad \leftarrow \quad \frac{N_l^*}{N}$$

AGGLOMERATIVE CLUSTERING

- *How many clusters to train?*
- We use Minimum Description Length (MDL) to choose the number of clusters.
  - Negative log-likelihood
  - + penalty term for number of clusters.

$$\begin{aligned} \mathrm{MDL}(K,\theta) &= -\sum_{n=1}^{N} \log\left(\sum_{l=1}^{K} \mathrm{p}(\vec{y}_n|\vec{\lambda}_l)\pi_l\right) + \tfrac{1}{2}L\log(NM) \\ L &= KM + (K-1) \end{aligned}$$

- 1. Run EM to convergence.
- 2. Merge the two most similar clusters.
- 3. Repeat 1,2 until we have a single cluster.
- 4. *Choose parameter set with smallest MDL.*

Decomposing Onsets onto Templates

- To solve this problem (add L1 penalty):

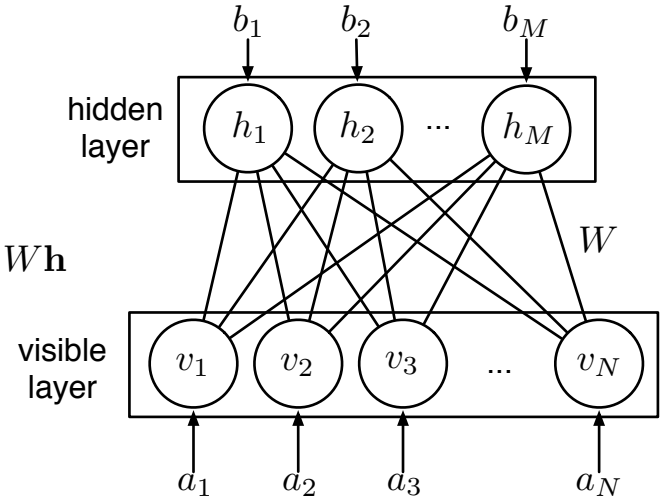
$$\min_{\vec{h}} d_{IS}(\vec{x}, W\vec{h}), \quad h_i \geq 0 \quad \forall i$$

- I use the IS distance as the cost function in the above.
  - While the IS distance is not strictly convex, in practice it is non-increasing under the following update rule:

$$\vec{h}_i \leftarrow \vec{h}_i . \frac{W^T((W\vec{h}_i)^{\cdot -2} . \vec{x}_i)}{W^T(W\vec{h}_i)^{\cdot -1}}$$

Restricted Boltzmann Machine

- Stochastic autoencoder [Hinton 2006]
  - Probabilistic graphical model
  - Weights/biases define:
  - $$\mathrm{p}(\mathbf{v},\mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z} \qquad E(\mathbf{v},\mathbf{h}) = -\mathbf{a}^{\mathrm{T}}\mathbf{v} - \mathbf{b}^{\mathrm{T}}\mathbf{h} - \mathbf{v}^{\mathrm{T}}W\mathbf{h}$$



- Factorial conditionals:

$$\mathrm{p}(\mathbf{v}|\mathbf{h}) = \prod_i \mathrm{p}(v_i|\mathbf{h}) \qquad \mathrm{p}(\mathbf{h}|\mathbf{v}) = \prod_i \mathrm{p}(h_i|\mathbf{v})$$

- Logistic activation probabilities

$$\mathrm{p}(v_i = 1|\mathbf{h}) = \sigma(a_i + \textstyle\sum_j W_{ij} h_j)$$
$$\mathrm{p}(h_i = 1|\mathbf{v}) = \sigma\left(b_i + \textstyle\sum_i W_{ij} v_i\right)$$

- Binary units (as opposed to real-valued units)
  - Act as a strong regularizer (prevent overfitting)
- Training: maximize likelihood of data under the model $\mathrm{p}(\mathbf{v})$

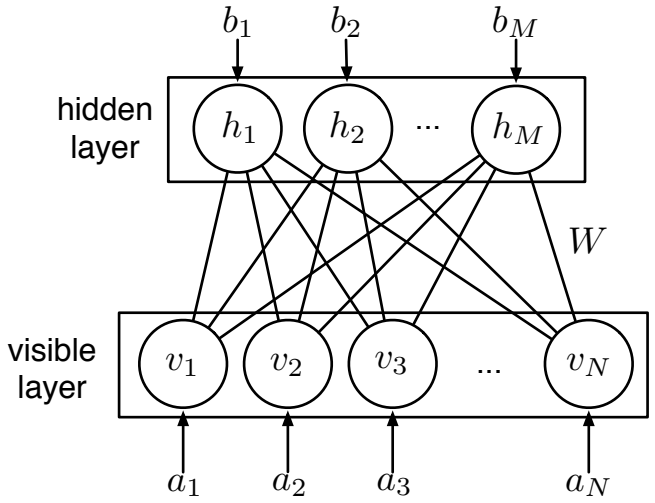
Contrastive Divergence

- ML learning can be done using Gibbs sampling to compute samples from:
  - The joint $\mathrm{p}(\mathbf{v}, \mathbf{h})$
  - By taking alternating samples from:

$$\mathrm{p}(v_i = 1|\mathbf{h}) = \sigma(a_i + \textstyle\sum_j W_{ij} h_j)$$
$$\mathrm{p}(h_i = 1|\mathbf{v}) = \sigma\left(b_i + \textstyle\sum_i W_{ij} v_i\right)$$

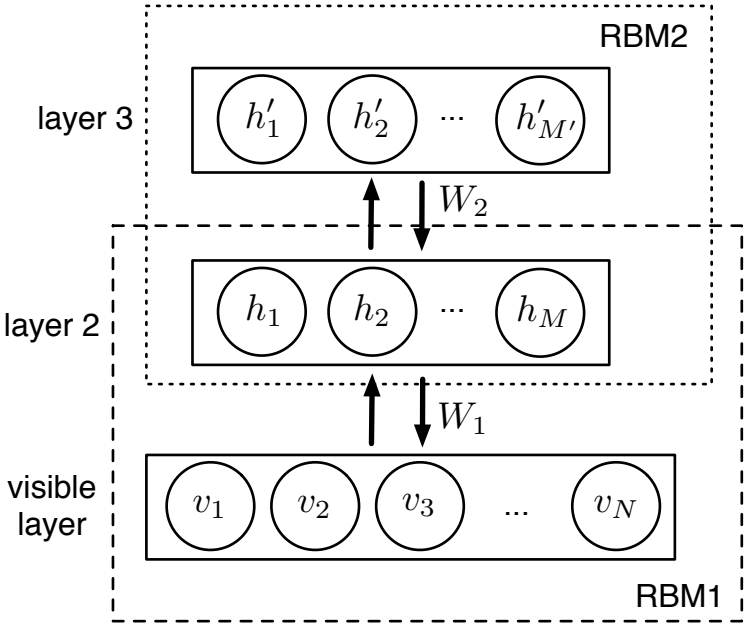
- But, this can take a very long time to converge.
- Approximation: "Contrastive Divergence" [Hinton 2006]
  - Take only a few alternating samples (k) for each update. (CD-k)

$$\Delta W_{ij} \propto \langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_k$$
$$\Delta a_i \propto \langle v_i \rangle_0 - \langle v_i \rangle_k$$
$$\Delta b_j \propto \langle h_j \rangle_0 - \langle h_j \rangle_k$$



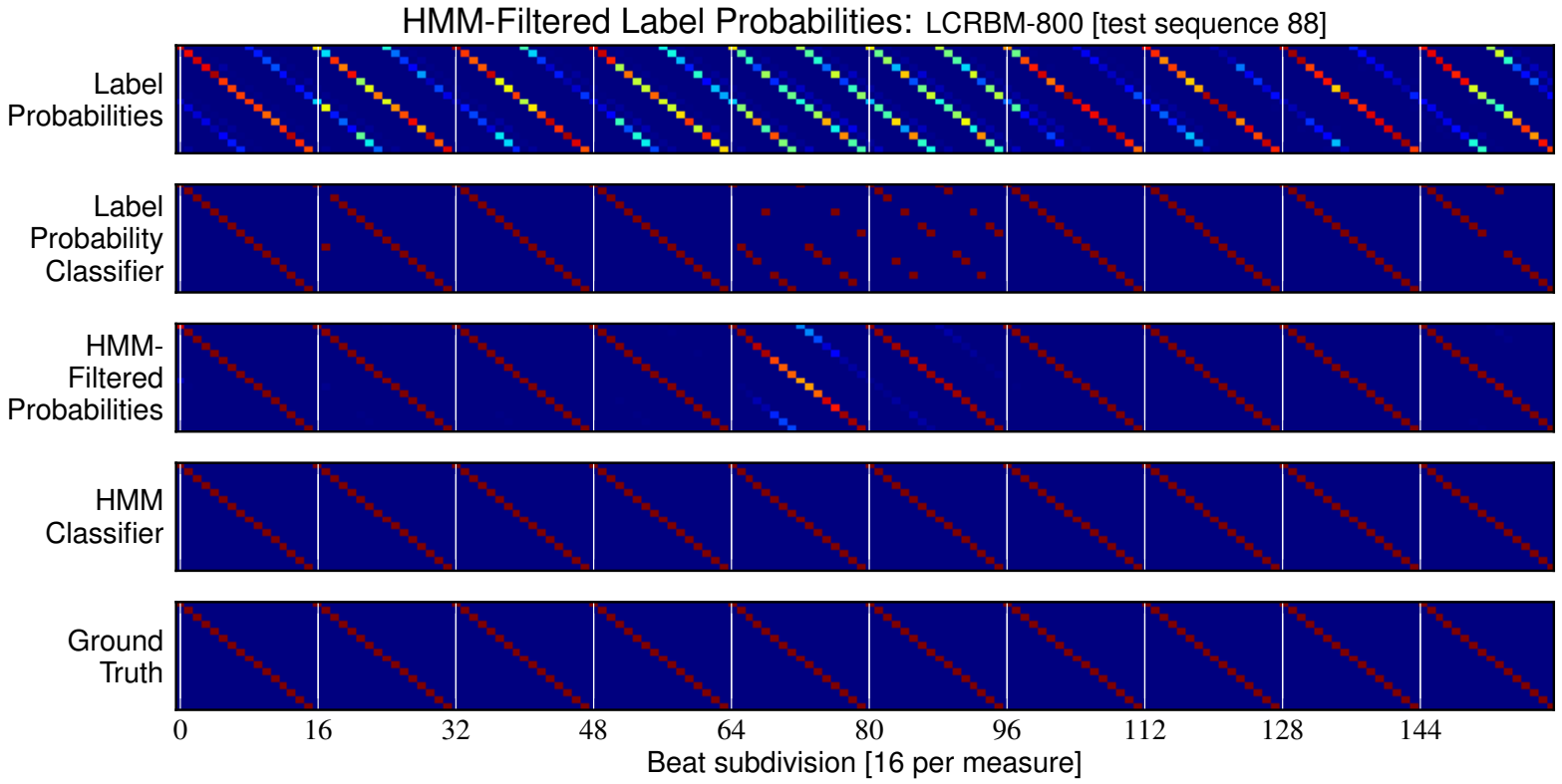
Deep Belief Networks (Stacking RBMs)

- Deep Belief Network (4+ layers)
  - Pre-train each layer as an RBM
- Greedy Pre-training
  - Train first level RBM
  - Use real-valued hidden unit activations of an RBM as input to subsequent RBM.
  - Train next RBM
  - Repeat until *deep* enough.
- Fine-Tuning
  - After pre-training, network is discriminatively fine-tuned using backpropagation of the cross-entropy error.



Example HMM Filtering

- White lines denote measure boundaries.



Example Output

- HMM-Filtered Label Estimates