LEARNING RHYTHM AND MELODY FEATURES WITH DEEP BELIEF NETWORKS

Erik M. Schmidt and Youngmoo E. Kim
Music and Entertainment Technology Laboratory (MET-lab)
Electrical and Computer Engineering, Drexel University
{eschmidt, ykim}@drexel.edu

ABSTRACT

Deep learning techniques provide powerful methods for the development of deep structured projections connecting multiple domains of data. But the fine-tuning of such networks for supervised problems is challenging, and many current approaches are therefore heavily reliant on pre-training, which consists of unsupervised processing on the input observation data. In previous work, we have investigated using magnitude spectra as the network observations, finding reasonable improvements over standard acoustic representations. However, in necessarily supervised problems such as music emotion recognition, there is no guarantee that the starting points for optimization are anywhere near optimal, as emotion is unlikely to be the most dominant aspect of the data. In this new work, we develop input representations using harmonic/percussive source separation designed to inform rhythm and melodic contour. These representations are beat synchronous, providing an event-driven representation, and potentially the ability to learn emotion informative representations from pre-training alone. In order to provide a large dataset for our pre-training experiments, we select a subset of 50,000 songs from the Million Song Dataset, and employ their 30-60 second preview clips from 7digital to compute our custom feature representations.

1. INTRODUCTION

Deep learning is rapidly becoming one of the most popular topics in the machine learning community, and such approaches offer powerful methods for finding deep structured connections in data. But the success of these methods often hinges on pre-training, or unsupervised methods that are used to provide a starting point to perform gradient descent optimization. As there is no guarantee of convexity in these problems, finding a useful initial starting point is paramount, as the best case scenario is generally limited to finding a good local minima.

In previous work, we have looked into deep learning methods for the prediction of musical emotion [1–3]. Deep belief networks (DBNs) were trained on magnitude spectra observations with the goal of predicting Arousal-Valence (A-V) coordinates, where valence indicates positive versus negative emotion, and arousal indicates emotional intensity. Using these models, the individual layers were treated as basis functions for feature extraction, and the learned representations were shown to outperform standard music information retrieval (Music-IR) features (e.g., mel-frequency cepstral coefficients).

But in looking to further improve these approaches, many questions remain in the pre-training methodology. Unsupervised methods such as restricted Boltzman machine (RBM) pre-training reduce the dimensionality of data based on the most prominent aspects. For instance, very compelling results have been shown on text data, where reducing text documents to two dimensions related directly to document type [4]. If the goal is to build a document type classifier, then this approach will yield an excellent starting position, but if it is document emotion we wish to model, then such a starting point may be no better than random. The same is true in music; if we cannot have the expectation of learning useful domains from unsupervised pre-training, then we should have low expectations for the supervised fine-tuning.

In this new work, we develop DBN input representations specifically designed to allow the model to learn about rhythm and melodic contour in an unsupervised fashion. Learning to understand these phenomena necessarily requires the ability to parse musical events, and we therefore begin with beat tracking, such that the models can be provided with a history of feature snapshots at musically relevant time points. In all of our feature extraction, we utilize harmonic/percussive source separation (HPSS) [5], allowing us to deconstruct the spectrum, separating out melody and harmonic sources from drums and percussive sources.

With the percussive spectra, we compute a beat synchronous percussion timbre feature, allowing us to parse different drum sounds and construct rhythm models by analyzing a history of samples. For the harmonic spectra, we compute a 48-dimensional beat synchronous chroma representation that allows the ability to track melodic contour over multiple octaves. In addition, we investigate the use of the 2-d FFT of beat synchronous chroma over four beat

segments, providing a shift (transposition) invariant feature for melodic contour that has been shown to be successful in cover song recognition [6].

In order to provide a reasonable dataset for pre-training we employ a set of 50,000, 30-60 second audio clips from 7digital that were randomly selected from the Million Song Dataset [7]. The DBNs are fine-tuned for predicting musical emotion using a publicly available dataset of time-varying musical emotion data [8].

2. BACKGROUND

Deep learning and DBN based feature learning is a topic of expanding attention in the machine listening community [9]. Lee *et al.* was one of the first to apply deep belief networks to acoustic signals, employing an unsupervised convolutional approach [10]. Their system employed PCA to provide a dimensionality reduced representation of the magnitude spectrum as input to the DBN and showed slight improvement over MFCCs for speaker, gender, and phoneme detection.

Hamel and Eck applied DBNs to the problems of musical genre identification and autotagging [11]. Their approach used raw magnitude spectra as the input to their DBNs, which were constructed from three layers and employed fifty units at each layer. The system was trained using greedy-wise pre-training and fine-tuned on a genre classification dataset, consisting of 1000, 30-second clips. The learned representations showed reasonable increases in performance over standard feature representations on both genre recognition and autotagging. The authors have also found significant improvement in moving to multi-timescale representations [12, 13].

Battenberg and Wessel applied conditional DBNs in modeling drum patterns in recent work, which incorporated an autoregressive time-varying restricted Boltzman machine model that can be used for generating sequences [14]. One downside of the conditional RBM for the application discussed in this new work is that the input history (past samples) only contributes to the bias term between the visible and hidden layer, and therefore the full information about rhythm may not be available in the upper hidden model layers.

3. DATA COLLECTION

In this paper, we use a universal background model style pre-training, initializing our models on a dataset of 50,000 songs, followed by fine-tuning on a 240 song labeled dataset of 15-second clips annotated with A-V emotion.

3.1 Unsupervised Pre-Training Data

For the unsupervised pre-training phase we seek to employ a large dataset in order to expose our model to a wide distribution of musical data. As such, we select a subset of 50,000 tracks from the Million Song Dataset (MSD). As the MSD includes only proprietary features, and we seek to handcraft original domains, we employ their 30-60 second preview clips from the 7digital API [1] . In order to ensure quality audio, we first download clips for the entire MSD and filter out any songs with less than 128 kbps MP3 bitrate, lower than 22050 Hz sampling rate, clips shorter than 30 seconds, clips that were found to be silent, and ones that had bad frames or file corruption issues.

3.2 Supervised Fine-Tuning Dataset

For the supervised musical emotion training dataset, we employ a corpus annotated in previous work using Amazon's Mechanical Turk (MTurk) [8]. The dataset contains 240, 15-second song clips that were sampled from a larger corpus that was annotated at 1-second intervals using a game-based approach. Each song clip was selected to ensure a uniform distribution was provided across the four quadrants of the A-V space. The goals of the MTurk activity were to assess the effectiveness of the game and to determine any biases created though collaborative labeling. Overall, the datasets were shown to be highly correlated, with arousal $r = 0.712$, and valence $r = 0.846$. The MTurk dataset is available to the research community [2] and is densely annotated, containing $4,064$ label sequences in total (16.93 ± 2.690 ratings per song).

4. ACOUSTIC REPRESENTATIONS

As previously discussed, learning to understand musical attributes, such as rhythm and melody, necessarily requires the ability to parse musical events. As such, the success of these methods hinges on our ability to accurately beat track music audio. All acoustic representations developed in this work employ harmonic/percussive source separation. With beat tracking, HPSS allows us to find the best onsets possible using percussive spectra. With rhythm features, it allows us to isolate just percussion (or percussive contributions of other pitched instruments), and to create features based on the timbre of percussion on the beat. Finally, with pitch features, it allows us to isolate harmonic (pitched) sources when creating our chroma representations. Figure 1 shows the feature extraction process for each stage in our processing. Beat tracking is shown in the center, and percussion and pitch features are on the left and right, respectively.

4.1 Harmonic/Percussive Source Separation

As the time/frequency considerations are different for each of our feature extraction chains (i.e., beat tracking, pitch, percussion timbre), we must perform HPSS three times for each of our 50,000 pre-training songs. As a result, we elect to use an efficient median filtering based approach [5]. The general idea of HPSS is that harmonic signals correspond to horizontal lines in the spectrogram (i.e., Fourier series) and percussive signals correspond to vertical lines (i.e., impulses). In Figure 2, we show a simple audio example of a guitar and drums mix.

[1] http://developer.7digital.net/

[2] http://music.ece.drexel.edu/research/emotion/moodswingsturk

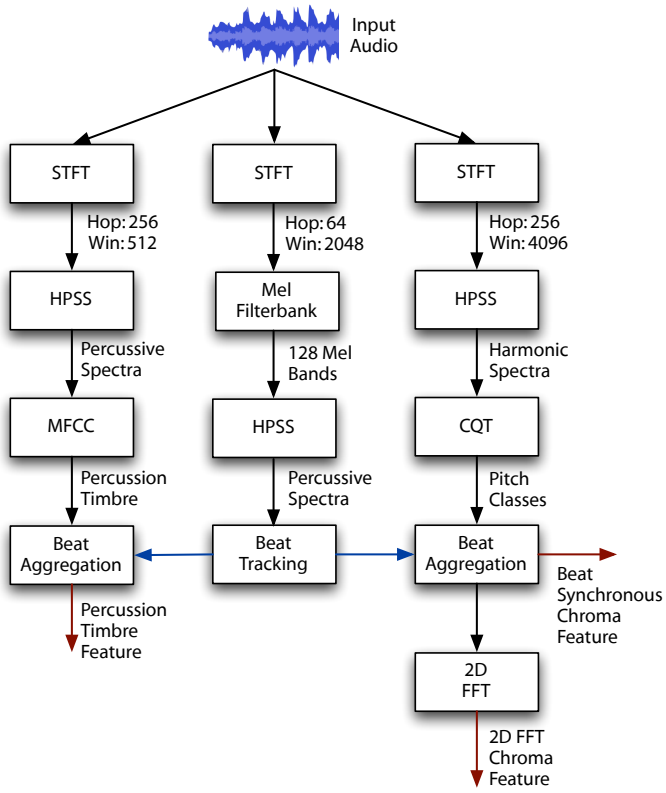

Figure 1. Feature extraction process for percussion timbre (left), beat detection (center), and pitch chroma representations (right).

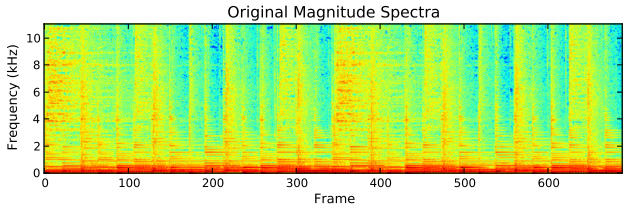

Figure 2. Original audio input spectra.

Median filtering based HPSS performs two passes of median filtering (vertically and horizontally) in order to generate spectral masks, and is therefore extremely efficient. Figure 3 shows the HPSS separation for the spectrogram shown in Figure 2.

4.2 Beat Tracking

Our beat tracking approach begins with an STFT with a 64 frame hop ($\sim$ 3msec) to provide maximal time resolution, followed by the application of a 128 bin mel-spaced filter bank that provides vertical smoothing in the spectrum, thus making percussive onsets more prominent. Following the filter bank, the onset profile is computed via a multidimensional Laplace filter, and the filter means are used in an Ellis-style beat tracker using librosa [3] [15].

[3] https://github.com/bmcfee/librosa

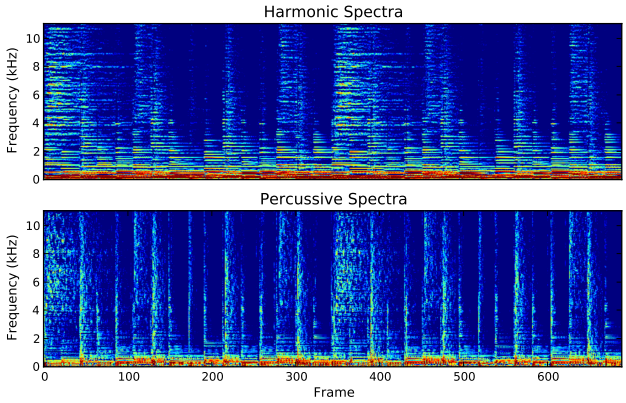

Figure 3. Harmonic percussive source separation.

4.3 Percussion Timbre

In order to train a model to understand rhythm, we extract a percussion timbre feature. This feature is shown in the left column of Figure 1, where we first extract the STFT with a window size of 512 samples ($\sim$ 23msec) and hop size of 256 ($\sim$ 11.6msec). We then compute HPSS, followed by MFCCs of the percussive spectra, providing a percussion timbre feature (e.g., to differentiate the boom sound of a bass drum vs. the hit of a snare). We then beat aggregate this feature such that the DBN is provided with event-driven feature updates to learn rhythmic styles. As shown in Figure 4, we can parse rhythm visually.

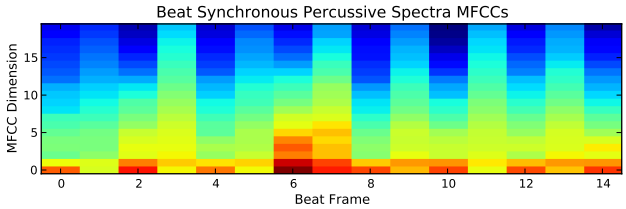

Figure 4. Beat synchronous aggregation of mel-frequency cepstral coefficients computed from the percussive spectrogram (percussion timbre).

4.4 Pitch Chroma

Following a similar pattern to the percussion timbre, we begin our chroma representation with HPSS as well, but with a much larger STFT window size. Here we use a 4096 ($\sim$ 186msec) window in order to provide reasonable frequency precision on bass frequencies. Next, we apply a magnitude constant-Q transform (CQT) filter bank starting at G_2 (98Hz), the first CQT filter that fits comfortably into our STFT representation, and spanning four octaves up to $F\sharp_6/G\flat_6$ (1479.98). Figure 5 displays our 48-dimensional chroma representation.

To learn a model of how the chroma evolve, we will need to present the DBN with multiple frames, and we therefore elect to center those event around beats, as we can have a reasonable expectation of a correlation with note onsets. This also greatly reduces the number of train-

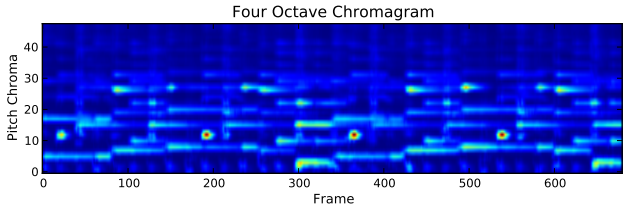

Figure 5. Four octave pitch chroma.

ing frames in our dataset, making the approach more computationally feasible. Figure 6 shows the beat aggregation of our chroma feature.

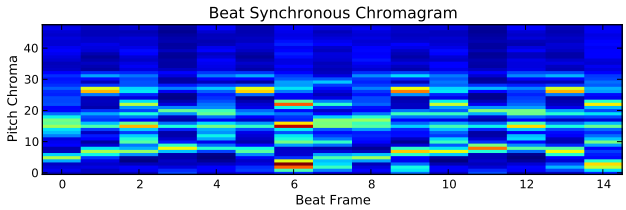

Figure 6. Beat synchronous chroma.

4.5 Chroma 2-d FFT

For our DBN input representation, we investigate using the magnitude 2-d FFT of our beat synchronous chroma representation. This feature was previously investigated in the realm of cover song detection, where it was found to perform well using 75-beat patches, providing shift (transposition) and time invariant properties for melody within a song [6]. Here we shorten this observation to just four beats, with the goal of obtaining shift/transposition invariance, but still retaining time information. Figure 7 shows this feature, where it is computed for each shift of a four beat window.

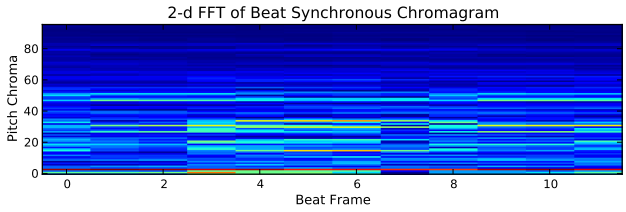

Figure 7. 2-d FFT of beat synchronous chroma.

5. DEEP BELIEF NETWORKS

A trained deep belief network shares an identical topology to a neural network, though they offer a far-superior training procedure, which begins with an unsupervised pre-training that models the hidden layers as restricted Boltzman machines (RBMs) [4, 16, 17] . A graphical depiction of our first layer RBM is shown in Figure 8, which uses four beat synchronous frames of observations in the input. An RBM is a generative model that contains only a single hidden layer, and in simplistic terms they can be thought of as two sets of basis vectors, one which reduces the dimensionality of the data and the other that reconstructs it.

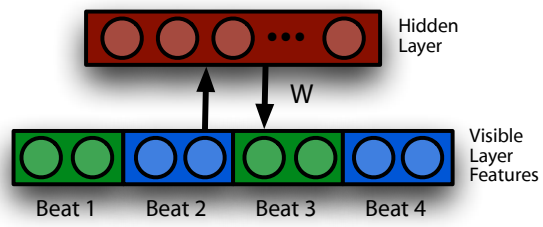

Figure 8. A restricted Boltzman machine with multiple beat observations.

RBMs are Markov random fields (MRFs) with hidden units, in a two layer architecture where we have visible units $\mathbf{v}$ and hidden units $\mathbf{h}$. During pre-training, we learn RBMs "greedily," where we learn them one at a time from the bottom up. That is, after we learn the first RBM we retain only the forward weights, and use them to create the input for training the next RBM layer.

For our first layer RBM we employ a Gaussian-binomial RBM, where the visible data is represented as a Gaussian distribution. The advantage of this representation over the standard binomial-binomial is that a sigmoid function is not applied during inference when estimating the visible layer from the hidden. With the standard binomial units, most visible values are forced to 0 or 1,

$$p(v_i = 1|\mathbf{h}) = \sigma(b_i + \sum_j w_{ij} h_j), \tag{1}$$

where the visible layer is $\mathbf{v} \in \mathbb{R}^{1\times I}$, the hidden layer $\mathbf{h} \in \mathbb{R}^{1\times J}$, and the model has parameters $\mathbf{W} \in \mathbb{R}^{I\times J}$, with biases $\mathbf{c} \in \mathbb{R}^{1\times J}$ and $\mathbf{b} \in \mathbb{R}^{1\times I}$.

The Gaussian-binomial RBM allows a more continuous range,

$$p(v_i|\mathbf{h}) = \mathcal{N}(b_i + \sum_j w_{ij} h_j, 1). \tag{2}$$

During our greedy-wise pre-training we use Gaussian-binomial RBMs at the first layer, which presents continuous data, but all subsequent layers use standard binomial-binomial RBMs.

For the Gaussian-binomial RBM, we have an energy distribution of the form,

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i \in \text{visible}} \frac{(v_i - b_i)^2}{2} - \sum_{j \in \text{hidden}} c_j h_j - \sum_{i,j} v_i h_j w_{ij}, \tag{3}$$

and the standard binomial-binomial RBM has an energy function of the form,

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in \text{visible}} b_i v_i - \sum_{j \in \text{hidden}} c_j h_j - \sum_{i,j} v_i h_j w_{ij}. \tag{4}$$

As in the typical approach to deep learning, after pre-training we form a multi-layer perceptron using only the forward weights of the RBM layers. As our goal is to learn feature detectors for a regression problem, we lastly attach a linear regression layer and report the prediction error for fine-tuning as the mean squared error of the estimators. We

trained our DBNs using Theano, [4] a Python-based package for symbolic math compilation.

6. EXPERIMENTS AND RESULTS

In the following experiments, we investigate employing DBNs for rhythm and melody feature learning. For each DBN, we use shrinking layer sizes, where layer 0 contains 75 nodes, layer 1 contains 50 nodes, and layer 2 contains 25 nodes. The goal with this approach is to best take advantage of the dimensionality reduction power of RBMs.

For our pre-training dataset, we use the 50,000 7digital preview clips described in Section 3.1, with beat synchronous features that are represented by taking every shift of a 4 beat window, and vectorizing the input as shown in Figure 8. For each feature type, we show the pre-training visible data dimensionality and total number of examples below in Table 1.

DBN Model Domain	Input Dimensionality	Number of Pre-Training Examples
Rhythm	80	$4,645,595$
Pitch Chroma	192	$4,645,526$
2-d FFT Chroma	384	$4,495,526$

Table 1. DBN pre-training Data

Pre-training epochs of 5, 10, 20, and 50 are investigated for all feature types with a learning rate of 10^{-5}. The best validation scores were found for 10 epochs with both the chroma and 2-d FFT of chroma, and 50 epochs with the percussion timbre representation. For each pre-trained model, we perform gradient descent back propagation fine-tuning for each fold, where for each input example $\mathbf{x}_i$, we train the model to produce the emotion space parameter vector $\mathbf{y}_i$,

$$\mathbf{y}_i = [\mu_a,\ \mu_v]. \tag{5}$$

In performing fine-tuning we note that our DBNs are beat-synchronous, but our labeled data is annotated at one-second intervals. In order to fine-tune our DBNs to predict emotion, we use linear interpolation to estimate the values of emotion on the beat. However, since we seek to compare this method to that of previous work, it necessarily must be evaluated on the second-by-second data test set. Therefore, after DBNs are trained and layer-wise features are computed, we then aggregate DBN features over the past 1-second, as is done with the standard feature domains, providing features at the same rate as the original labels.

We evaluate these learned representations in the context of multiple linear regression (MLR), as we have investigated in prior work [18–20], where we develop regressors to predict the parameterization vector $\mathbf{y}_i$ of a two-dimensional Gaussian in A-V space,

$$\mathbf{y}_i = [\mu_a,\ \mu_v,\ \sigma^2_{aa},\ \sigma^2_{vv},\ \sigma^2_{av}]. \tag{6}$$

[4] http://deeplearning.net/software/theano/

In all supervised experiments, the model training is cross-validated 5 times, dividing the dataset into 50% training, 20% verification, and 30% testing. To avoid the well-known album-effect, we ensured that any songs that were recorded on the same album were either placed entirely in the training or testing set. Note, for three songs in the dataset, the beat tracker returned less than four beats in the labeled portion of the song, and as a result they had to be removed from the sets. Those songs are IDs: 2996, 5232, 6258. We post updated results for standard features over previous work in Table 2, and note that their removal leaves the results nearly unchanged [3].

As in previous approaches, we use Euclidean distances to evaluate our A-V mean predictions in a normalized space (i.e., axes bound between -0.5 and 0.5), and Kullback-Liebler divergences to analyze our Gaussian predictions. We note a slight difference in the KL divergence calculation from our previous work,

$$\begin{aligned}\mathrm{KL}(p||q) = &\frac{1}{2}\Big(\log\frac{|\Sigma_q|}{|\Sigma_p|} + \mathrm{tr}(\Sigma_q^{-1}\Sigma_p) + \\ &(\mu_q-\mu_p)^T\Sigma_q^{-1}(\mu_q-\mu_p) - d\Big),\end{aligned} \tag{7}$$

where in previous work we had omitted the $\frac{1}{2}$ multiplier term (see [20]).

Feature Type	Average Mean Distance	Average KL Divergence
MFCC	0.140 ± 0.004	0.642 ± 0.069
Chroma	0.182 ± 0.005	1.654 ± 0.143
Spectral Shape	0.153 ± 0.005	0.755 ± 0.074
Spectral Contrast	$\mathbf{0.139 \pm 0.005}$	$\mathbf{0.647 \pm 0.072}$
ENT	0.151 ± 0.005	0.700 ± 0.079

Table 2. Emotion regression results from previous work for fifteen second clips.

Results for the different DBN feature types are shown in Table 3. For each learned feature type, we investigate that feature alone, as well as that feature in combination with the others. As we pull the spectrum apart with HPSS to learn the different DBN features, it makes sense that we should put the two domains back together for prediction. The rhythm feature, which uses the percussion timbre as input, is the best single performing feature at a mean error of 0.128, and the best result overall is when combining the pitch and 2-d FFT of chroma at 0.113 mean error.

7. DISCUSSION AND FUTURE WORK

This work presented a novel approach for training deep belief networks for understanding rhythm and melody. The fine-tuned DBN features easily outperformed any other singular existing representation, and the combination of the rhythm and melody DBN features outperformed any other system previously tested on this dataset.

In moving forward with deep learning approaches that require pre-training, we believe that it should be based around input observations from which high level musical

DBN Layer	DBN Feature Type	Pre-training Model Error		Fine-tuning Model Error	
		Mean Distance	KL Divergence	Mean Distance	KL Divergence
Layer 0	Rhythm	$\mathbf{0.146 \pm 0.007}$	$\mathbf{0.681 \pm 0.083}$	0.139 ± 0.009	0.652 ± 0.085
Layer 1	Rhythm	0.148 ± 0.007	0.708 ± 0.086	0.132 ± 0.013	0.598 ± 0.094
Layer 2	Rhythm	0.156 ± 0.006	0.754 ± 0.086	$\mathbf{0.128 \pm 0.016}$	$\mathbf{0.582 \pm 0.104}$
Layer 0	Pitch	$\mathbf{0.160 \pm 0.004}$	$\mathbf{0.786 \pm 0.105}$	0.143 ± 0.015	0.678 ± 0.122
Layer 1	Pitch	0.161 ± 0.005	0.792 ± 0.086	0.131 ± 0.022	0.618 ± 0.149
Layer 2	Pitch	0.165 ± 0.006	0.815 ± 0.095	$\mathbf{0.129 \pm 0.024}$	$\mathbf{0.608 \pm 0.153}$
Layer 0	2-d FFT Pitch	$\mathbf{0.171 \pm 0.007}$	$\mathbf{0.889 \pm 0.103}$	0.148 ± 0.014	0.716 ± 0.132
Layer 1	2-d FFT Pitch	0.175 ± 0.007	0.926 ± 0.116	0.137 ± 0.022	0.669 ± 0.166
Layer 2	2-d FFT Pitch	0.175 ± 0.006	0.915 ± 0.112	$\mathbf{0.129 \pm 0.024}$	$\mathbf{0.620 \pm 0.170}$
Layer 0	Rhythm+Pitch	$\mathbf{0.145 \pm 0.006}$	$\mathbf{0.685 \pm 0.082}$	0.129 ± 0.012	0.604 ± 0.091
Layer 1	Rhythm+Pitch	0.147 ± 0.007	0.709 ± 0.078	0.117 ± 0.019	0.534 ± 0.122
Layer 2	Rhythm+Pitch	0.153 ± 0.007	0.743 ± 0.089	$\mathbf{0.114 \pm 0.022}$	$\mathbf{0.514 \pm 0.125}$
Layer 0	Rhythm+2-d FFT Pitch	$\mathbf{0.147 \pm 0.005}$	$\mathbf{0.707 \pm 0.075}$	0.131 ± 0.013	0.606 ± 0.098
Layer 1	Rhythm+2-d FFT Pitch	0.151 ± 0.006	0.739 ± 0.077	0.119 ± 0.019	0.552 ± 0.128
Layer 2	Rhythm+2-d FFT Pitch	0.156 ± 0.007	0.763 ± 0.082	$\mathbf{0.113 \pm 0.022}$	$\mathbf{0.514 \pm 0.131}$

Table 3. Emotion regression results for Mechanical Turk annotated clips. Rhythm features use percussion timbre as input, pitch features use beat synchronous chroma, and 2-d FFT pitch features use our four beat 2-d FFT of chroma representation. Feature combination results are all early fusion based (concatenation of dimensions).

ideas like rhythm, melody, and harmony can easily be extracted. Furthermore, as understanding these ideas necessarily requires the presentation of time-series data, future approaches should further investigate the best way to present this information to the first DBN layer.

In continuing this work, we wish to further analyze the optimal number of beat synchronous frames to present to the DBN input, as well as investigating smaller units of musical events, such as eighth or sixteenth note feature updates. It would also be interesting to apply these learned features in the context of a graphical model such as a conditional random field, as investigated in prior work [21].

8. ACKNOWLEDGMENT

This work is supported by National Science Foundation awards IIS-0644151 and CNS-0960061.

9. REFERENCES

[1] E. M. Schmidt and Y. E. Kim, "Learning emotion-based acoustic features with deep belief networks," in *IEEE WASPAA*, New Paltz, NY, 2011.

[2] ——, "Modeling the acoustic structure of musical emotion with deep belief networks," in *NIPS Workshop on Music and Machine Learning*, 2011.

[3] E. M. Schmidt, J. Scott, and Y. E. Kim, "Feature learning in dynamic environments: Modeling the acoustic structure of musical emotion," in *ISMIR*, Porto, Portugal, October 2012.

[4] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[5] D. FitzGerald, "Harmonic/percussive separation using median filtering," in *DAFx*, Graz, Austria, September 2010.

[6] T. Bertin-Mahieux and D. P. W. Ellis, "Large-scale cover song recognition using the 2d fourier transform magnitude," in *ISMIR*, Porto, Portugal, October 2012.

[7] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *ISMIR*, Miami, FL, October 2011.

[8] J. A. Speck, E. M. Schmidt, B. G. Morton, and Y. E. Kim, "A comparative study of collaborative vs. traditional annotation methods," in *ISMIR*, Miami, Florida, 2011.

[9] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics," in *ISMIR*, Porto, Portugal, October 2012.

[10] H. Lee, Y. Largman, P. Pham, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *NIPS*, 2009.

[11] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks," in *ISMIR*, Utrecht, Netherlands, 2010.

[12] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck, "Temporal pooling and multiscale learning for automatic annotation and ranking of music audio," in *ISMIR*, Miami, FL, October 2011.

[13] P. Hamel, Y. Bengio, and D. Eck, "Building musically-relevant audio features through multiple timescale representations," in *ISMIR*, Porto, Portugal, October 2012.

[14] E. Battenberg and D. Wessel, "Analyzing drum patterns using conditional deep belief networks," in *ISMIR*, Porto, Portugal, October 2012.

[15] D. P. W. Ellis, "Beat tracking by dynamic programming," *JNMR*, vol. 36, no. 1, pp. 51–60, March 2007.

[16] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[17] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *NIPS*, 2007.

[18] E. M. Schmidt, D. Turnbull, and Y. E. Kim, "Feature selection for content-based, time-varying musical emotion regression," in *ACM MIR*, Philadelphia, PA, 2010.

[19] E. M. Schmidt and Y. E. Kim, "Prediction of time-varying musical mood distributions from audio," in *ISMIR*, Utrecht, Netherlands, 2010.

[20] ——, "Prediction of time-varying musical mood distributions using Kalman filtering," in *IEEE ICMLA*, Washington, D.C., 2010.

[21] ——, "Modeling musical emotion dynamics with conditional random fields," in *ISMIR*, Miami, FL, 2011.