A CLASSIFICATION-BASED POLYPHONIC PIANO TRANSCRIPTION APPROACH USING LEARNED FEATURE REPRESENTATIONS

Juhan Nam
Stanford University
juhan@ccrma.stanford.edu

Jiquan Ngiam
Stanford University
jngiam@cs.stanford.edu

Honglak Lee
Univ. of Michigan, Ann Arbor
honglak@eecs.umich.edu

Malcolm Slaney
Yahoo! Research
malcolm@ieee.org

ABSTRACT

Recently unsupervised feature learning methods have shown great promise as a way of extracting features from high dimensional data, such as image or audio. In this paper, we apply deep belief networks to musical data and evaluate the learned feature representations on classification-based polyphonic piano transcription. We also suggest a way of training classifiers jointly for multiple notes to improve training speed and classification performance. Our method is evaluated on three public piano datasets. The results show that the learned features outperform the baseline features, and also our method gives significantly better frame-level accuracy than other state-of-the-art music transcription methods.

1. INTRODUCTION

Music transcription is the task of transcribing audio into a score. It is a challenging problem because multiple notes are often played at once (polyphony), and thus individual notes interfere by virtue of their harmonic relations.

A number of methods have been proposed since Moorer first attempted to use computers for automatic music transcription [10]. State-of-the-art methods can be categorized into three approaches: iterative F0 searches, joint source estimation and classification-based approaches. Iterative F0-searching methods first find the predominant F0 and subtract its relevant sources (e.g. harmonic partials) from the input signal and then repeat the procedure on what remains until no additional F0s are found [6]. Joint source estimation examines possible combinations of sound sources by hypothesizing that the input signal is approximated by a weighted sum of the sound sources with different F0s [3].

While these two methods are based on utilizing the structure of musical tones, classification-based approaches address polyphonic transcription as a pattern-recognition problem. The idea is to use multiple binary classifiers, each of which corresponds to a note class. They are trained with short-time acoustic features and labels for the corresponding note class (i.e., note on/off) and then used to predict the note labels for new input data. Although classification-based approaches make minimum use of knowledge of acoustics, they show comparable results to iterative F0 searches and joint source estimation, particularly for piano music [9, 12]. However, when the training set is limited or the piano in the test set has different timbre, tuning or recording environments, classification-based approaches can overfit the training data, a problem common to many supervised learning tasks [13]. As a means to obtain features robust to acoustic variations, researchers have designed networks of adaptive oscillators on auditory filter banks or normalized spectrogram on the frequency axis [9, 12].

The majority of machine learning tasks rely on these kinds of hand-engineered approaches to extract features. Recently, on the other hand, unsupervised feature learning methods that automatically capture the statistical relationship in data and learn feature representations have shown great promise. In particular, deep belief networks have been successfully applied to many computer-vision and speech-recognition tasks as an alternative to typical feature-extraction methods, but also a few music-related tasks [4, 8].

In this paper, we apply deep belief networks to polyphonic piano transcription. Specifically, we extend a previous classification-based approach in two ways: (1) by using learned feature representations for note classifiers and (2) by jointly training the classifiers for multiple notes. In particular, the latter associates deep belief networks with multi-task learning. The results show that our approach outperforms compared music transcription methods for several test sets.

2. FEATURE LEARNING

Deep belief networks (DBNs) are constructed by stacking restricted Boltzmann machines (RBMs) and training them in a greedy layer-wise manner. In this section, we briefly review RBMs and how to build a deep structure.

2.1 Sparse Restricted Boltzmann Machines

The RBM is a two layer undirected graphical model that has hidden nodes **h** and visible nodes **v** [11]. The visible nodes represent the data while the hidden nodes represent the features discovered by training the RBM. For each possible assignment to the hidden and visible nodes, the RBM specifies the probability of the assignment (Eq. 1). The RBM has symmetric connections between the two layers denoted by a weight matrix W, but no connections within hidden nodes or visible nodes. This particular configuration makes it easy to compute the conditional probability distributions, when $\mathbf{v}$ or $\mathbf{h}$ is fixed (Eq. 2). In practice, one uses this conditional probability of the hidden nodes as the "learned" features:

$$-\log P(\mathbf{v},\mathbf{h}) \propto E(\mathbf{v},\mathbf{h}) = \frac{1}{2\sigma^2}\mathbf{v}^T\mathbf{v} - \frac{1}{\sigma^2}\left(\mathbf{c}^T\mathbf{v} + \mathbf{b}^T\mathbf{h} + \mathbf{h}^T W\mathbf{v}\right) \quad (1)$$

$$p(h_j|\mathbf{v}) = sigmoid(\frac{1}{\sigma^2}(b_j + \mathbf{w}_j^T\mathbf{v})) \quad (2)$$

where σ^2 is a scaling parameter, $\mathbf{b}$ and $\mathbf{c}$ are learned biases, and W is a learned weight matrix. This formulation models the visible nodes as real-valued Gaussian units and the hidden nodes as binary units. We further regularize the model with sparsity by encouraging each hidden unit to have a pre-determined expected activation using a regularization penalty [7].

2.2 Deep Belief Network

A deep network is composed of multiple non-linear hidden layers (as opposed to a shallow network with a single hidden layer). Each layer in a deep network builds upon representations discovered by the previous layer to represent more complex features of the data. A DBN is trained by "greedy layer-wise stacking" of RBMs. First, a single layer RBM is trained to model the data. This RBM learns a set of weights W and biases $\mathbf{b}, \mathbf{c}$ that we fix as the parameters of the first layer of the DBN. To learn the next layer of weights and biases, we compute the features discovered by the first layer RBM (Eq. 2) and apply them to a binary-binary RBM (which has binary input units instead of Gaussian) to learn another layer of representation; this forms the parameters for our next layer of features. Deeper layers are learned in a similar fashion. Hinton et al. showed that the preceding learning algorithm for a DBN always improves a variational lower bound on the log-likelihood of the data when training more layers [5].

After training, the features learned from a DBN are extracted using a feed-forward approximation for the probabilities of the hidden nodes at the deepest layer (i.e. cascades of sigmoids) given the visible nodes. These features can be used for tasks such as classification. In practice, one often further refines the features learned by the DBN by treating the feature extraction process and classifier as a deep feed-forward neural network. The initialization of the deep neural network using RBMs is often known as unsupervised "pre-training," while supervised training with backpropagation is often known as supervised "finetuning." The pre-training/finetuning approach for learning deep networks has been shown to be essential for training deep networks. Specifically, training a deep network with only supervised backpropagation from random initialization does not work as well as pre-training.

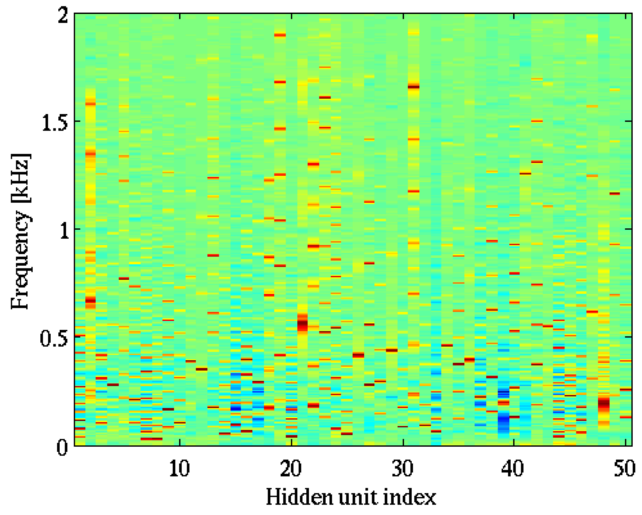


Figure 1: Randomly selected feature bases learned from spectrograms of piano music. Most feature bases capture harmonic distributions which correspond to various pitches while a few contain non-harmonic patterns.

2.3 Application To Audio Spectrogram

In this paper, we apply DBNs to audio spectrograms. The DBNs are built in two stages. The first stage performs unsupervised learning with sparse RBMs up to two hidden layers in order to find sparse hidden units that represent spectrogram frames. The second (optional) stage uses backpropagation to finetune the representation so that note classifiers have better discrimination power to correctly identify note on and off events. Figure 1 displays features bases (column vectors of matrix W) learned from spectrograms of classical piano music by a sparse RBM.

3. CLASSIFICATION-BASED TRANSCRIPTION

We build our polyphonic piano-transcription model based on Poliner and Ellis' frame-level note classification system [12,13]. Furthermore, we extend their system by using DBN-based feature representations and by jointly training classifiers for multiple notes.

3.1 Single-note Training

Poliner and Ellis' piano transcription system consists of 87 independent support vector machine (SVM) classifiers, each of which predicts the presence of a corresponding piano note when given an audio feature vector (a single column

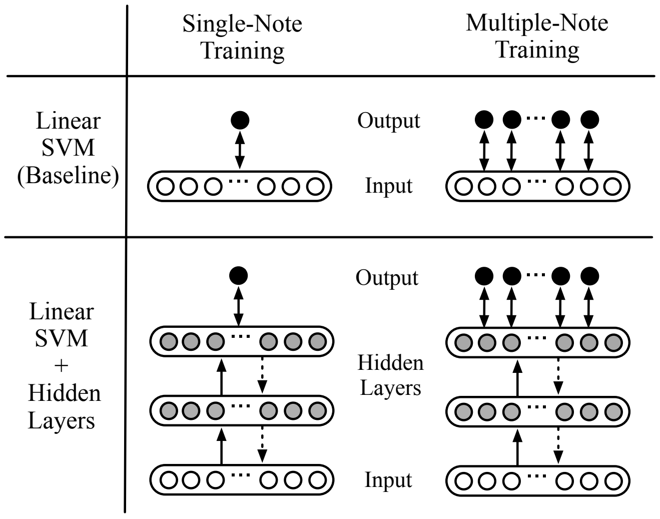


Figure 2: Network configurations for single-note and multiple-note training. Features are obtained from feed-forward transformation as indicated by the bottom-up arrows. They can be finetuned by back-propagation as indicated by the top-down arrows.

of a normalized spectrogram). Their transcription system requires individual supervised training for each note. Thus, we refer to this as single-note training.

We constrained the SVM in our experiments to a linear kernel because Poliner and Ellis reported that high-order kernels (e.g. RBF kernel) provided only modest performance gains with significantly more computation [13] and also a linear SVM is more suitable to large-scale data. We formed the training data by selecting spectrogram frames that include the note (positive examples) and those that do not include it (negative examples). Poliner and Ellis randomly sampled 50 positive (when available) and negative examples from each piano song per note. We used their sampling paradigm for single-note training.

While their system used a normalized spectrogram, we replaced it with DBN-based feature representations on spectrogram frames. As shown in the left column of Figure 2, the previous approach directly feeds spectrogram frames into SVM, whereas our approach transforms the spectrogram frames into mid-level features via one or two layers of learned networks and then feeds them into the classifier. We also finetuned the networks with the error from the SVM.

3.2 Multiple-note Training

When we experimented with single-note training described above, we observed that the classifiers are somewhat "aggressive", that is, they produced even more "false alarm" errors (detect inactive notes as active ones) than "miss" errors (fail to detect active notes). In particular, this significantly degraded onset accuracy. Also, it was substantially slow to finetune the DBN networks separately for each note. Thus, we suggest a way of training multiple binary classifiers at the same time. We refer to this as multiple-note training.

The idea is to sum 88 SVM objectives and train them with shared audio features and 88 binary labels (at a given time, a single audio feature has 88 corresponding binary labels), as if we train a single classifier. [1] This allows cross-validation to be jointly performed for 88 SVMs, thereby saving a significant amount of training time. On the other hand, this requires a different way of sampling examples. Since we combined all 88 notes in our experiments, all spectrogram frames except silent ones are a positive example to at least one SVM. Thus we sampled training data by selecting spectrogram frames at every K frame time. K was set to 16 as a trade-off between data reduction and performance. Note that this makes the ratio of positive and negative examples for each SVM determined by occurrences of the note in the whole training set, thereby having significantly more negative examples than positive ones for most SVMs. It turned out that this "unbalanced" data ratio makes the classifiers "less aggressive," as a result, increasing overall performance.

We illustrate multiple-note training in the right column of Figure 2. In fact, without finetuning the DBNs, multiple-note training is equivalent to single-note training with the unbalanced data ratio. The only difference is that the single-note training does separate cross-validation for each SVM. We compared multiple-note training to the single-note training with the unbalanced data ratio, but found no noticeable difference in performance. On the other hand, when we finetune the DBNs, these two training approaches become completely different. While single-note training produces separate DBN parameters for each note, multiple-note training allows the networks to shares the parameters among all notes by updating them with the errors from the combined SVMs. For example, when the multiple-note training looks at the presence of a C3 note given input features, it simultaneously checks out if other notes (e.g. C4 or C5) are played. This can be seen as an example of multi-task learning.

3.3 HMM Post-processing

The frame-level classification described above treats training examples independently without considering dependency between frames. Poliner and Ellis used HMM-based post-processing to temporally smooth the SVM prediction. They modeled each note independently with a two-state HMM. We also adopted this approach. In our implementation, however, we converted the SVM output (distance to the boundary) to a posterior probability using

$$p(y_i = 1|\mathbf{x}_i) = sigmoid(\alpha(\boldsymbol{\theta}^T\mathbf{x}_i)), \tag{3}$$

[1] The classifier we used is a linear SVM with a L2-regularized L2-loss [2]. We implemented the SVM in MATLAB using minFunc, which is a Matlab library found in `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`. Thus, summing 88 SVM objectives was done by simply treating 88 binary labels as a vector.

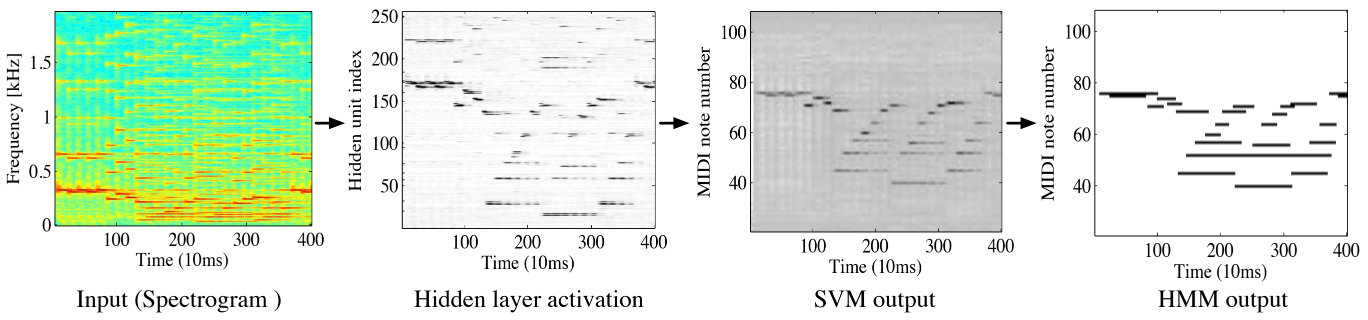


Figure 3: Signal transformation through the DBNs and classification stages

where $\mathbf{x}_i$ is a SVM input vector, $\boldsymbol{\theta}$ are SVM parameters, y_i is a label and α is a scaling constant. α was chosen from a pre-determined list of values as part of the cross-validation stage. The smoothing process was performed for each note class by running a Viterbi search based on a 2x2 transition matrix and a note on/off prior obtained from the training data, and the posterior probability.

Figure 3 shows signal transformation through the DBN networks along with HMM post-processing. The SVM output was computed as the distance to the decision boundary in a linear SVM. Note that the hidden layer activation is more similar to the final output than the spectrogram.

4. EVALUATION

4.1 Datasets

We used three datasets to evaluate our method.

Poliner and Ellis set consists of 124 MIDI files of classical piano music. They were rendered into 124 synthetic piano sound and 29 real piano recordings [12]. The first 60-second excerpt of each song was used.

MAPS is a large piano dataset that includes various patterns of playing and pieces of music [1]. We used 9 sets of piano pieces, each with 30 songs. They were created by various high-quality software synthesizers (7 sets) and Yamaha Disklavier (2 sets). We used the first 30-second excerpt of each song in the validation and test sets but the same length at a random position for the training set.

Marolt set consists of 3 synthetic piano and 3 real piano recordings [9]. This set was used only for test.

4.2 Pre-processing

We first computed spectrogram from the datasets with a 128-ms window and 10ms overlaps. To remove note dynamics, we normalized each column by dividing entries with their sum, and then compressed it using cube root, commonly used as an approximation to the loudness sensitivity of human ears. Furthermore, we applied PCA whitening to the normalized spectrogram, retaining 99% of the training data variance and adding 0.01 to the variance before the whitening. This yielded roughly 50-60% dimensionality reduction and lowpass filtering in the PCA domain. The ground truth was created from the MIDI files. We extended note offset times by 100ms in all training data to make up for room effect in the piano recordings. The extended note length was experimentally determined.

4.3 Unsupervised Feature Learning

We trained the first and second-layer DBN representations using the pre-processed spectrogram. The hidden layer size was chosen to 256 and the expected activation of hidden units(sparsity) was cross-validated over 0.05, 0.1, 0.2 and 0.3, while other parameters were kept fixed.

4.4 Evaluation Metrics

We primarily used the following metric of accuracy:

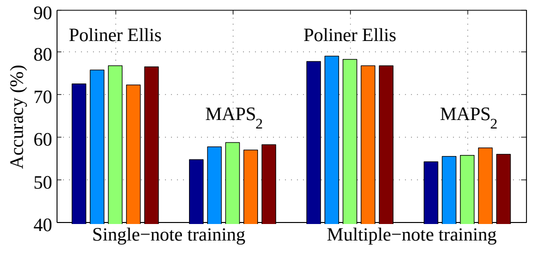
$$\text{Accuracy} = \frac{\text{TP}}{\text{FP+FN+TP}}, \tag{4}$$

where TP (true positive) is the number of correctly predicted examples, FP (false positives) is the number of note-off examples transcribed as note-on, FN (false negative) is the number of note-on examples transcribed as note-off. This metric is used for both frame-level and onset accuracy. Frame-level accuracy is measured by counting the correctness of frames every 10ms, and onset accuracy is by searching a note onset of the correct pitch within 100 ms of the ground-truth onset. In addition, we used the F-measure for frame-level accuracy to compare our results to those published using the metric.

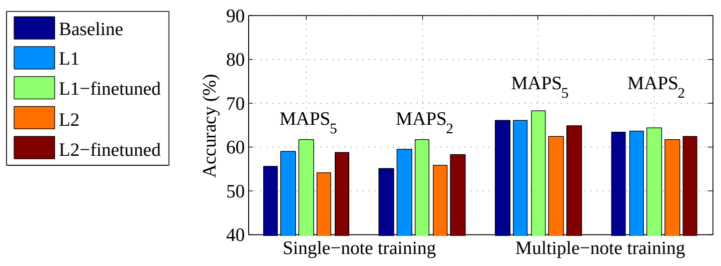
4.5 Training Scenarios

Our method is evaluated in two different scenarios. In the first scenario, we mainly used the Poliner and Ellis set, splitting it into training, validation and test data following [12]. In order to avoid overfitting to the specific piano set, we selected 26 songs from two synthesizer pianos sets in MAPS and used them as an additional validation set. For convenience, we refer to this subset as MAPS_2. In the second scenario, we used five remaining synthesizer piano sets in MAPS for training to examine if our method generalizes well when trained on diverse types of timbre and recording conditions. For validation, we randomly took out 26 songs from the five piano sets, calling them MAPS_5 to distinguish it from the actual training data. We additionally used MAPS_2 for validation in the second scenario as well. [2]

[2] The lists of MAPS songs for training, validation and test are specified in `http://ccrma.stanford.edu/~juhan/ismir2011.html`



(a) Scenario 1



(b) Scenario 2

Figure 4: Frame-level accuracy on validation sets in two scenarios. The first and second-layer DBN features are referred to as L1 and L2.

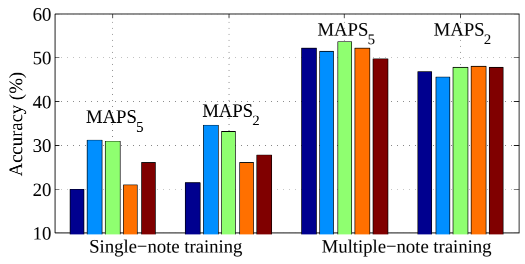


Figure 5: Onset accuracy on validation sets (scenario 2)

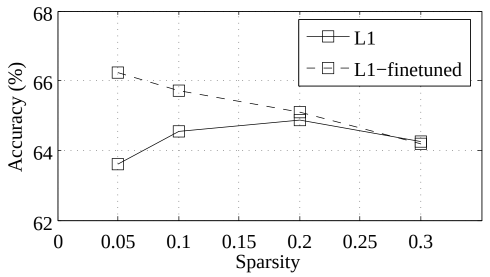


Figure 6: Frame-level accuracy VS sparsity (hidden layer activation in RBMs)

4.6 Validation Results

We compare the baseline feature (normalized spectrogram by cube root) to the first- and second-layer DBN features and their finetuned versions on validation sets in the two scenarios. The results are shown in Figure 4 and Figure 5.

In scenario 1, DBN features generally outperform the baseline. In single-note training, finetuned L1-features give the highest accuracy on both validation sets. In multiple-note training, unsupervised L1- or L2-features achieve slightly better results. In comparison of the two training methods, either one appears to be not superior to the other, showing subtle differences: Multiple-note training gives slightly better results when the same piano set are used for validation (Poliner and Ellis), whereas single-note training does a little better job when different pianos set (MAPS$_2$) are used.

In scenario 2, the results show that DBN L1-features always achieve better results than the baseline but DBN L2-features generally give worse accuracy. Finetuning always improves results on both validation sets, although the increment is very limited on MAPS$_2$ in multiple-note training. In comparison of the two training methods, multiple-note training outperforms single-note training for both validation sets, particularly giving the best accuracy on MAPS$_2$. The superiority of multiple-note training is even more apparent in onset accuracy as shown in Figure 5.

Figure 6 shows the influence of sparsity (hidden layer activation in RBMs) on frame-level accuracy. The accuracy is the average value on two validation sets (MAPS$_5$ and MAPS$_2$) when L1 features are used in multiple-note training and scenario 2. The results indicate that relatively less sparse features perform better before finetuning; however, with finetuning, sparse features achieve the highest accuracy as well as the best improvement.

4.7 Test Results: Comparison With Other Methods

The validation results show that a single layer of DBN is the best-performing feature representation and multiple-note training is better than single-note training. Thus, we chose DBN L1-features and multiple-training to run our system on test sets. Also, we evaluated both unsupervised and finetuned features.

Table 1 shows results on the Poliner and Ellis test set, and Marolt set. We divided the table into two groups to make a fair comparison. The upper group uses the same dataset for both training and testing (the Poliner and Ellis set) whereas the lower group assumes that the piano tones in the test sets were "unheard" in training or uses different transcription algorithms. In the upper group, Poliner and Ellis' transcription system adopted a normalized spectrogram and a nonlinear SVM. Our method outperformed their approach for both test sets. In the lower group, our method trained with MAPS (scenario 2) also produced better accuracy than the two published results on both sets. Note that, in both groups, unsupervised features give better results than finetuned features when different piano sets are used for training and testing. As for onset accurary, we achieved 62% in training scenario 1 on the Poliner and Ellis test set, which is very close to the Poliner and Ellis' result (62.3%).

Table 2 compares our method with other algorithms evaluated on the MAPS test set, composed of 50 songs selected from the two Disklavier piano sets by [15]. The finetuned DBN-features in our method give the highest frame-level accuracy among compared methods.

Algorithms	P. and E.	Marolt
Poliner and Ellis [12] †	67.7%	44.6%
Proposed (S1-L1)	71.5%	**47.2%**
Proposed (S1-L1-finetuned)	**72.5%**	46.45%
Marolt [9] †	39.6%	46.4%
Ryyananen and Klapuri [14] †	46.3%	50.4%
Proposed (S2-L1)	**63.8%**	**52.0%**
Proposed (S2-L1-finetuned)	62.5%	51.4%

Table 1: Frame-level accuracy on the Poliner and Ellis, and Marolt test set. The upper group was trained with the Poliner and Ellis train set while the lower group was with other piano recordings or uses different methods. S1 and S2 refer to training scenarios. †These results are from Poliner [12].

Algorithms	Precision	Recall	F-measure
Marolt [9] †	74.5%	57.6%	63.6%
Vincent et al. [15] †	71.6%	65.5%	67.0%
Proposed (S2-L1)	**80.6%**	67.8%	73.6%
Proposed (S2-L1-ft.)	79.6%	**69.9%**	**74.4%**

Table 2: Frame-level accuracy on the MAPS test set in F-measure. "ft" stands for finetuned. †These results are from Vincent [15].

5. DISCUSSION AND CONCLUSIONS

We have applied DBNs to classification-based polyphonic piano transcription. The results show that a learned feature representation by a DBN, particularly L1 features, provide better transcription performance than the baseline features and our classification approach outperforms compared piano transcription methods.

Our evaluation shows that finetuning generally improves accuracy, particularly when sparse features are used. However, unsupervised features often work better when the system is tested on different piano sets. This indicates that unsupervised features are more robust to acoustic variations.

We also suggested multiple-note training. Compared to single-note training, this method improved not only transcription accuracy but also training speed. In our computing environment, multiple-note training was more than five times faster than single-note training when the DBNs are finetuned.

Our method is based on frame-level feature learning and binary classification under simple two-state note event modeling. We think that more refinements will be possible by modeling richer states to represent dynamic properties of musical notes.

6. REFERENCES

[1] V. Emiya, R. Badeau and B. David: "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transaction on Audio, Speech and Language Processing*, vol.18, no.6, pp.1643–1654, 2010.

[2] R. Fan, K. Chang, C. Hseigh, X. Wang and C. Lin: "LIBLINEAR: a Library for Large Linear Classification," *Journal of Machine Learning Research*, 9:1871–1974, 2008.

[3] M. Goto: "A predominant-f0 estimation method for CD recordings:MAP estimation using EM algorithm for adaptive tone models," *Preceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2001.

[4] P. Hamel and D. Eck: "Learning Features from Music Audio With Deep Belief Networks," *Proceedings of the 11th International Society for Music Information Retrieval Conference*, 2010.

[5] G. E. Hinton, S. Osindero, and Y. W. Teh: "A fast learning algorithm for deep belief nets," *Neural computation*, 18(7):1527–1554, 2006.

[6] A. Klapuri: "A perceptually motivated multiple-f0 estimation method," *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005.

[7] H. Lee, C. Ekanadham, and A. Ng: "Sparse deep belief net model for visual area V2," *Advances in Neural Information Processing Systems*, 2007.

[8] H. Lee, Y. Largman, P. Pham, and A.Y. Ng: "Unsupervised feature learning for audio classification using convolutional deep belief networks," *Advances in Neural Information Processing Systems(NIPS)*, 22, 2009.

[9] M. Marolt: "A connectionist approach to automatic transcription of polyphonic piano music," *IEEE Transactions on Multimedia*, vol.6, no.3, pp.439–449, 2004.

[10] J.A.Moorer: "On the transcription of musical sound by computer," *Computer Music Journal*, vol.1, no.4, pp.32–38, 1987.

[11] P. Smolensky: "Information processing in dynamical systems:Foundation of harmony theory," In D.E. Rumelhart, J.L. McClelland (Eds.), *Parallel Distributed Processing*, vol.1, chapter.6, pp.194–281, Cambridge, MIT Press, 1986

[12] G. Poliner and D. Ellis: "A discriminative model for polyphonic piano transcription," *EURASIP Journal on Advances in Signal Processing*, vol.2007, 2007.

[13] G. Poliner and D. Ellis: "Improving generalization for classification-based polyphonic piano transcription," *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2007.

[14] M. Ryynanen and A. Klapuri: "Polyphonic music transcription using note event modeling," *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005.

[15] E. Vincent, N. Bertin, R.Badeau: "Adaptive Harmonic Spectral Decomposition for Multiple Pitch Estimation," *IEEE Transaction on Audio, Speech and Language Processing*, vol.18, no.3, pp.528–537, 2010.