Speaker-dependent Multipitch Tracking Using Deep Neural Networks

Yuzhou Liu[1], *DeLiang Wang*[1,2]

[1]Department of Computer Science and Engineering, The Ohio State University, USA
[2]Center for Cognitive and Brain Sciences, The Ohio State University, USA
{liuyuz,dwang}@cse.ohio-state.edu

Abstract

Multipitch tracking is a challenging problem for speech and signal processing. In this paper, we use deep neural networks (DNNs) to model the probabilistic pitch states of two simultaneous speakers. To closely capture speaker-dependent information and improve the accuracy of speaker assignment, we train a DNN for each enrolled speaker (speaker-dependent DNN). We also explore the feasibility of training a DNN for each speaker pair in the system (speaker-pair-dependent DNN). A factorial hidden Markov model (FHMM) then integrates the pitch probabilities and generates most likely pitch contours with a junction tree algorithm. We evaluate our system on the GRID corpus. Experiments show that our approach substantially outperforms state-of-the-art multipitch trackers on both same-gender and different-gender two-talker mixtures.

Index Terms: multipitch tracking, deep neural networks, speaker-dependent modeling, factorial hidden Markov model.

1. Introduction

There is a long-standing interest in estimating the pitch, or the fundamental frequency (F0) of speech. A reliable estimation of pitch is critical for many speech applications, including speech separation [1] [2], speaker identification [3] and automatic speech recognition [4]. Over the last few decades, various algorithms have been designed for tracking the pitch of a single speaker in clean recordings [5] [6]. However, pitch tracking when speech is severely corrupted by noise or interfering speakers is still a challenging problem.

This paper is concerned with pitch tracking in a situation when multiple speakers (two in this study) are talking at the same time. A number of studies have investigated this problem. Wu, Wang and Brown [7] built a probabilistic representation of pitch on top of a channel/peak selection mechanism and tracked continuous pitch contours with a hidden Markov model (HMM). Sha and Saul [8] modeled the instantaneous frequency spectrogram with nonnegative matrix factorization and used the inferred weight coefficients to determine pitch candidates. Bach and Jordan [9] proposed a model based on the direct probabilistic modeling of the spectrogram and tracked several pitches with a factorial HMM (FHMM). Jin and Wang [10] further improved Wu, Wang and Brown's system by designing new techniques for channel selection and pitch score estimation in the context of reverberant and noisy signals. All of the abovementioned studies built a general system without modeling the characteristics of any specific speaker, denoted as speaker-independent models. Although most speaker-independent models perform reasonably well for estimating pitch periods, they fail to correctly assign pitch estimates to the underlying speakers for multipitch tracking. To alleviate this problem, Duan *et al.* [11] proposed a post processing approach, which used the pitch estimation of speaker independent models as input and reassigned pitch streams using a constrained clustering algorithm. On the other hand, speaker-dependent models have also been investigated recently. Wohlmayr, Stark and Pernkopf [12] modeled the probability of pitch periods using speaker-dependent Gaussian mixture models (GMMs), and then used a speaker-dependent FHMM to track pitches of two simultaneous speakers. They have shown significant improvement over a previous speaker-independent model [7].

In this paper, we adopt the idea of speaker-dependent modeling and propose a discriminative technique to generate the probability of pitches at each time frame. Firstly, we use deep neural networks (DNNs) to model the posterior probability that a pair of frequency bins (pitch states) are pitched given the frame-level observation. A DNN is a feedforward neural network that contains more than one hidden layer [13]. Recently, Han and Wang [14] used DNNs to model the posterior probability of pitch states for single-pitch tracking, which motivates the use of DNNs for multipitch tracking in this study. We expect DNNs to generate more accurate pitch probabilities than Wolhmayr *et al.*'s GMMs [12]. To reflect the speaker dependency of our model, we train a DNN for each speaker enrolled in the system. In addition, we explore the feasibility of training DNNs for different pairs of speakers. Secondly, we use an FHMM for pitch tracking. Under the framework of the FHMM, the pitch state of each speaker evolves within its own Markov chain, while the emission probability is derived by the posterior probability estimated by DNNs. We then use the junction tree algorithm [15] to track the most likely pitch trajectories.

The rest of the paper is organized as follows. The next section describes the overall system architecture. The details of the proposed multipitch tracking algorithm are discussed in Section 3. Experimental results and comparisons are presented in Section 4. Finally, Section 5 concludes the paper.

2. System overview

A flowchart of our proposed multipitch tracker is illustrated in Figure 1. The input to the system is a speech mixture $v_{(t)}$ sampled at 1.6 KHz:

$$v_{(t)} = u_{(t)}^{(1)} + u_{(t)}^{(2)} \tag{1}$$

where $u_{(t)}^{(1)}$ and $u_{(t)}^{(2)}$ are utterances from two different speakers, mixed at 0 dB in our study. Given the mixture, our system first extracts frame-level features $\boldsymbol{y}_{(m)}$. The features are then fed into DNNs to derive the posterior probability of pitches at frame m, i. e., $p(x_{(m)}^{(1)}, x_{(m)}^{(2)}|\boldsymbol{y}_{(m)})$, where $x_{(m)}^{(1)}$ and $x_{(m)}^{(2)}$ denote pitch states of two speakers at frame m. Both $x_{(m)}^{(1)}$ and $x_{(m)}^{(2)}$ have 68 states $(s^1, s^2, s^3, \ldots, s^{68})$, where s^1 refers to an

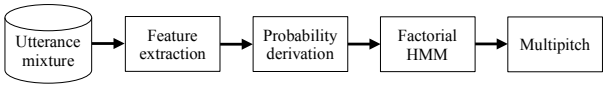


Figure 1: *Overall system architecture.*

unvoiced or silent state, and s^2 to s^{68} encode different pitch frequencies ranging from 60 to 404 Hz [14]. Specifically, we quantize the pitch frequency range 60 to 404 Hz using 24 bins per octave in a logarithmic scale, resulting in a total of 67 bins. The value of $p(x_{(m)}^{(1)} = s^i, x_{(m)}^{(2)} = s^j|\boldsymbol{y}_{(m)})$ equals one if groundtruth pitches fall into the i^{th} and j^{th} frequency bin respectively. Depending on the level of speaker dependency, we propose two types of DNNs to estimate the posterior probability, which are the speaker-dependent DNN and the speaker-pair-dependent DNN. The detailed settings of DNNs can be found in Section 3. We then convert the posterior probability $p(x_{(m)}^{(1)}, x_{(m)}^{(2)}|\boldsymbol{y}_{(m)})$ to the emission probability of an FHMM $p(\boldsymbol{y}_{(m)}|x_{(m)}^{(1)}, x_{(m)}^{(2)})$, and apply the junction tree algorithm to infer the most likely pitch trajectories.

3. Algorithm description

3.1. Feature extraction

Good features should encode the information of pitch and speaker identity at the same time. We have investigated three features: cochleagram, log spectrogram and Mel-frequency cepstral coefficients in a development set, and decide to use the cochleagram feature in the following experiments due to its superior performance.

3.1.1. Cochleagram feature

To get the cochleagram feature, we first decompose the input signal into the time-frequency domain by using a bank of 64 gammatone filters whose center frequencies range from 50 Hz to 8000 Hz. Gammatone filters model the impulse response of auditory filters and are widely used in speech applications [21]. We then divide each sub-band signal into 20 ms frames with a 10 ms frame shift. The cochleagram is derived by computing the energy of each subband signal at each frame. In the end, we loudness compress the cochleagram with a cubic root operation to get the final cochleagram feature.

3.1.2. Feature post processing

To make use of the temporal context, we concatenate neighboring frames into the feature vector. Assume the frame-level feature before post processing is $\hat{\boldsymbol{y}}_{(m)}$, the final feature vector can be written as:

$$\boldsymbol{y}_{(m)} = [\hat{\boldsymbol{y}}_{(m-d)}, \ldots, \hat{\boldsymbol{y}}_{(m)}, \ldots, \hat{\boldsymbol{y}}_{(m+d)}] \tag{2}$$

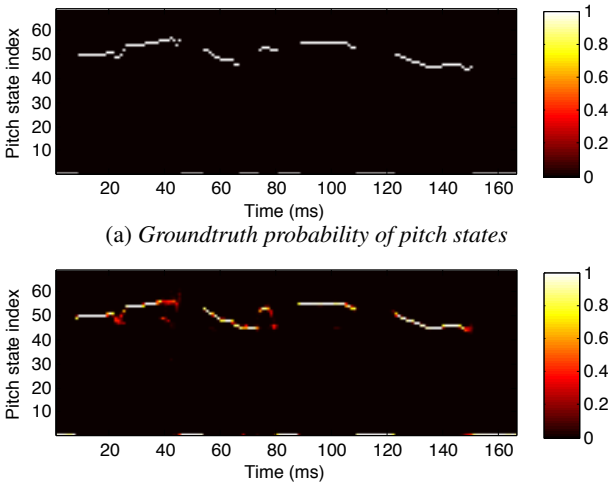
where d is chosen as 5 from cross-validation.

3.2. DNNs for posterior probability estimation

We introduce two types of DNNs for posterior probability estimation.

3.2.1. Speaker-dependent DNN

The goal of DNNs is to model the posterior probability that a pair of pitch states occur at frame m, i. e., $p(x_{(m)}^{(1)}, x_{(m)}^{(2)}|\boldsymbol{y}_{(m)})$. However, this would be difficult without the prior knowledge of the underlying speakers. In this section, we focus on training speaker-dependent DNNs to model the posterior probability.



(a) *Groundtruth probability of pitch states*

(b) *Probability of pitch states estimated by DNN*

Figure 2: *Comparison between the groundtruth and estimated pitch-state probability.*

According to the chain rule in probability theory:

$$p(x_{(m)}^{(1)}, x_{(m)}^{(2)}|\boldsymbol{y}_{(m)}) = p(x_{(m)}^{(1)}|\boldsymbol{y}_{(m)})p(x_{(m)}^{(2)}|x_{(m)}^{(1)}, \boldsymbol{y}_{(m)}) \tag{3}$$

we can estimate $p(x_{(m)}^{(1)}|\boldsymbol{y}_{(m)})$ and $p(x_{(m)}^{(2)}|x_{(m)}^{(1)}, \boldsymbol{y}_{(m)})$ in turn to get $p(x_{(m)}^{(1)}, x_{(m)}^{(2)}|\boldsymbol{y}_{(m)})$. In this study, we estimate the pitch-state probability of speaker one $p(x_{(m)}^{(1)}|\boldsymbol{y}_{(m)})$ by training a DNN. The input layer of the DNN corresponds to the feature vector of a speech mixture. There are four hidden layers in the DNN, and each hidden layer has 1024 hidden units with the ReLU activation function [16]. The output layer has 68 sigmoid units (O_1^1 to O_1^{68}), where O_1^j estimates $p(x_{(m)}^{(1)} = s^j|\boldsymbol{y}_{(m)})$. Hence there are 67 '0's and a '1' in the desired output. The value '1' corresponds to the frequency bin of the groundtruth pitch. We use the mean square error as the cost function. The standard backpropagation algorithm and dropout regularization (dropout rate 0.2) are used to train the network, with no pre-training [17]. We adopt stochastic gradient descent along with a momentum term (0.9) for the optimization. The training process requires mixtures of speaker one and a set of interfering speakers. During testing, the output of the DNN is scaled to sum to one. Figure 2 compares the groundtruth and the estimated pitch-state probability of speaker one in a female-female test mixture.

As shown in Figure 2, the DNN is powerful enough to model the conditional probability of $x_{(m)}^{(1)}$ even without knowing $x_{(m)}^{(2)}$. Thus we further assume the conditional independence between $x_{(m)}^{(1)}$ and $x_{(m)}^{(2)}$:

$$p(x_{(m)}^{(2)}|x_{(m)}^{(1)}, \boldsymbol{y}_{(m)}) = p(x_{(m)}^{(2)}|\boldsymbol{y}_{(m)}) \tag{4}$$

In the next step, we train another DNN to model $p(x_{(m)}^{(2)}|\boldsymbol{y}_{(m)})$ using exactly the same structure and training methodology as for the first DNN. After estimating $p(x_{(m)}^{(2)}|\boldsymbol{y}_{(m)})$, the original posterior probability can be obtained by:

$$p(x_{(m)}^{(1)}, x_{(m)}^{(2)}|\boldsymbol{y}_{(m)}) = p(x_{(m)}^{(1)}|\boldsymbol{y}_{(m)})p(x_{(m)}^{(2)}|\boldsymbol{y}_{(m)}) \tag{5}$$

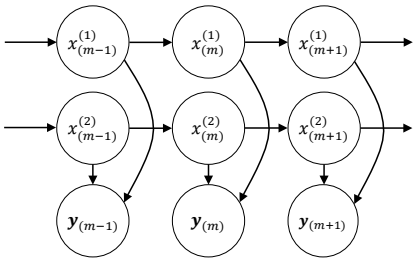


Figure 3: *A Factorial HMM with two Markov chains.*

As we train a DNN for each enrolled speaker, we denote this model as the speaker-dependent DNN (SD-DNN).

When training the SD-DNN for speaker one, we can concatenate the pitch state of the interfering speaker k into the DNN's output. The resulting output has 136 units, $[O_1^1, \ldots, O_1^{68}, O_k^1, \ldots, O_k^{68}]$. The training of the new DNN is conducted by jointly minimizing the mean square error of the target and the interfering pitch state, where the error of the interfering pitch state can be regarded as a regularization term for the training process. Other settings exactly follow the initial DNN. During the test phase, we only use the output corresponding to the target speaker (O_1^1 to O_1^{68}) to estimate $p(x_{(m)}^{(1)}|\boldsymbol{y}_{(m)})$. The same method applies to every speaker in the system. We denote this model as the regularized speaker-dependent DNN (RSD-DNN).

3.2.2. Speaker-pair-dependent DNN

A speaker-pair-dependent DNN (SPD-DNN) is a DNN trained and tested exclusively on a specific pair of speakers. The structure and training of a SPD-DNN are the same as a RSD-DNN. The input layer corresponds to the frame-level feature vector and the output layer estimates the pitch-state probabilities of two speakers (i. e., $[O_1^1, \ldots, O_1^{68}, O_2^1, \ldots, O_2^{68}]$). Only a single DNN is needed when testing on a specific speaker pair. The estimated posterior probability of pitch states is computed by:

$$p(x_{(m)}^{(1)} = s^i, x_{(m)}^{(2)} = s^j|\boldsymbol{y}_{(m)}) = \alpha O_1^i O_2^j \tag{6}$$

where α is a scalar.

As a SPD-DNN is trained exclusively on a speaker pair, it can accurately capture the identity information of both speakers. We expect it to yield better results when two speakers are of the same gender and can not be easily distinguished from each other.

Finally, we analyze the training cost of all DNNs. For a system with N speakers enrolled, we need to train N DNNs for the SD-DNN and $\frac{N(N-1)}{2}$ DNNs for the SPD-DNN. Although SPD-DNNs are expected to generate better results, they would come at a higher computational cost.

3.3. Inference on a factorial HMM

A factorial HMM model is a probabilistic graphical model that contains several Markov chains [18]. In this study, we only discuss the case of two Markov chains, as shown in Figure 3. The hidden variables ($x_{(m)}^{(1)}$, $x_{(m)}^{(2)}$) are the pitch states of two speakers, and the observation variable is the feature vector $\boldsymbol{y}_{(m)}$. The Markov assumption implies that $\boldsymbol{y}_{(m)}$ is independent of all variables given $x_{(m)}^{(1)}$ and $x_{(m)}^{(2)}$. Assume the total number of frames is M, we denote the sequence of variables in boldface: $\boldsymbol{x} = \bigcup_{m=1}^{M}\{x_{(m)}^{(1)}, x_{(m)}^{(2)}\}$, $\boldsymbol{y} = \bigcup_{m=1}^{M}\{\boldsymbol{y}_{(m)}\}$. The overall probability of the model is given by:

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(x_{(1)}^{(1)})p(x_{(1)}^{(2)})p(\boldsymbol{y}_{(1)}|x_{(1)}^{(1)}, x_{(1)}^{(2)}) \prod_{m=2}^{M} p(x_{(m)}^{(1)}|x_{(m-1)}^{(1)})p(x_{(m)}^{(2)}|x_{(m-1)}^{(2)})p(\boldsymbol{y}_{(m)}|x_{(m)}^{(1)}, x_{(m)}^{(2)}) \tag{7}$$

Prior probabilities and transition matrices of the hidden variables are computed from the training data speaker-dependently. Laplace smoothing is applied during the training process. The emission probability can be computed using the estimated posterior probability and Bayes rule:

$$p(\boldsymbol{y}_{(m)}|x_{(m)}^{(1)}, x_{(m)}^{(2)}) = \frac{\alpha p(x_{(m)}^{(1)}, x_{(m)}^{(2)}|\boldsymbol{y}_{(m)})}{p(x_{(m)}^{(1)})p(x_{(m)}^{(2)})} \tag{8}$$

where α is a constant.

Once all probabilities are derived, we apply the junction tree algorithm [15] to infer the most likely sequence of pitch states. The time complexity of the junction tree algorithm is $O(2 \times 68^3 \times M)$ in our study, which is still tractable. We then convert derived pitch states to mean frequencies of corresponding frequency bins. In the end, we use a moving average window of length three to smooth frequencies and get the final pitch estimate.

4. Experimental results and comparisons

For evaluations and comparisons, we use the GRID database [19], which is also used in [12]. The corpus consists of high quality recordings of 1000 sentences spoken by each of 34 speakers (18 male, 16 female). Two male and two female speakers (speaker No. 1, 2, 18, 20 respectively, denoted as MA1, MA2, FE1 and FE2) are selected to train and test the SD-DNN, RSD-DNN and SPD-DNN based methods. We denote these four speakers as Set One. For each speaker in Set One, 997 sentences are used for training, while the remaining three sentences (see [12]) are used for testing. Another ten male and ten female speakers (speaker No. 3, 5, 6, 9, 10, 12, 13, 14, 17, 19; 4, 7, 8, 11, 15, 16, 21, 22, 23, 24 respectively) are used as interfering speakers in the training of the SD-DNN/RSD-DNN, where again 997 sentences of each speaker are used for training. We denote these 20 speakers as Set Two. Groundtruth pitches are extracted from single-speaker utterances using RAPT [5]. The choice of RAPT is justified in [20], where RAPT outperforms other pitch trackers on clean speech signals. The details of the training and test set are described as follows:

- SD-DNN/RSD-DNN: training mixtures are created by mixing utterances of the target speaker in Set One with utterances of interfering speakers in Set Two at 0 dB. Each utterance of the target speaker is mixed with five utterances of each interfering speaker in Set Two. Thus there are 99,700 training mixtures for every target speaker. The test is conducted within Set One. We extensively mix test utterances for each speaker pair in Set One, resulting in a total of $3\times3\times6 = 54$ test mixtures.
- SPD-DNN: for each speaker pair in Set One, we build the training set by mixing utterances of the two speakers at 0 dB. To generate enough training data, we make sure that each utterance of one speaker is mixed with about 100 utterances of the other speaker. Thus approximately 99,700 mixtures are created to train each speaker pair. We use the same test set as for the SD-DNN/RSD-DNN.

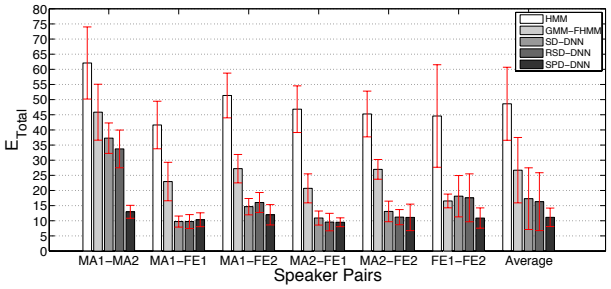


Figure 4: *E_{total} of different approaches tested on the GRID corpus. Error bars depict the mean and standard deviation of a method over the test mixtures of a given speaker pair.*

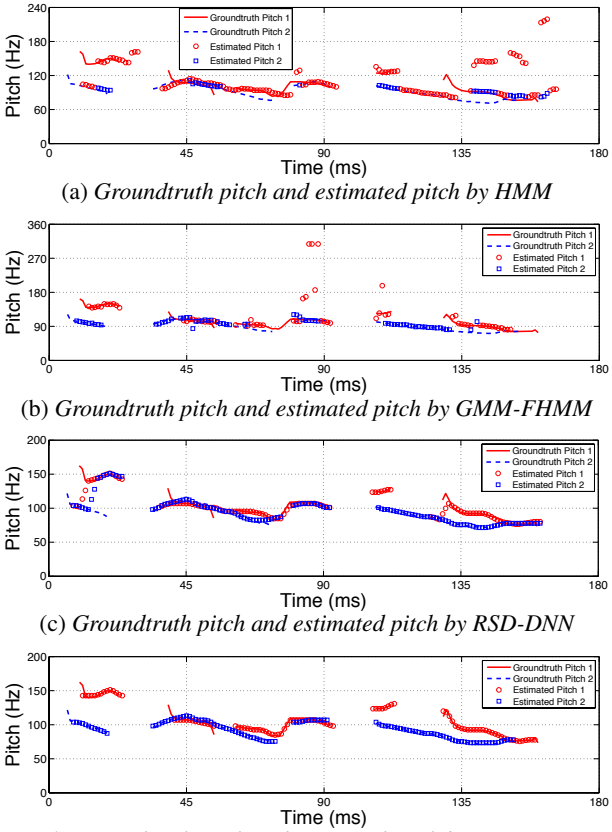
We evaluate pitch tracking results using the error measure proposed in [12], which jointly evaluates the performance in terms of pitch accuracy and speaker assignment. Assume the ground truth pitch trajectories are $f_0^1[m]$ and $f_0^2[m]$, we globally assign each estimated pitch trajectory to a groundtruth pitch trajectory based on the minimum mean square error and denote the assigned estimated pitch trajectories as $\tilde{f}_0^1[m]$ and $\tilde{f}_0^2[m]$. The pitch frequency deviation of speaker i ($i \in \{1, 2\}$) is:

$$\Delta f^{(i)}[m] = \frac{|\tilde{f}_0^i[m] - f_0^i[m]|}{f_0^i[m]} \tag{9}$$

The voicing decision error E_{ij} denotes the percentage of time frames where i pitch points are detected as j pitch points. For each speaker i, the permutation error $E_{Perm}^{(i)}$ is set to one at time frames where the voicing decision for both estimates is correct, but $\Delta f^{(i)}[m]$ exceeds 20%, and $\tilde{f}_0^i[m]$ is within the 20% error bound of the other reference pitch (i. e., the error due to incorrect speaker assignment). The overall permutation error E_{Perm} is the percentage of time frames where either $E_{Perm}^{(1)}$ or $E_{Perm}^{(2)}$ is one. Next, for each speaker i, the gross error $E_{Gross}^{(i)}$ is set to one at time frames where the voicing decision for both estimates is correct, but $\Delta f^{(i)}[m]$ exceeds 20% and no permutation error is detected. The overall gross error E_{Gross} is the percentage of time frames where either $E_{Gross}^{(1)}$ or $E_{Gross}^{(2)}$ is one. The fine detection error $E_{Fine}^{(i)}$ is defined as the average of $\Delta f^{(i)}[m]$ in percent at time frames where $\Delta f^{(i)}[m]$ is smaller than 20%. $E_{Fine} = E_{Fine}^{(1)} + E_{Fine}^{(2)}$. The total error is used as the performance measure in this study: $E_{total} = E_{01}+E_{02}+E_{10}+E_{12}+E_{20}+E_{21}+E_{Perm}+E_{Gross}+E_{Fine}$.

We compare our methods (SD-DNN, RSD-DNN and SPD-DNN based) with two state-of-the-art multipitch trackers: Jin and Wang's [10] (denoted as HMM) and Wohlmayr *et al.*'s speaker-dependent approach [12] (denoted as GMM-FHMM). Jin and Wang' approach does not require training. Wohlmayr *et al.* trained their models on the same corpus and groundtruth pitch as we use, and we directly adopt their code in testing.

The results of our methods and those of the comparison algorithms are shown in Figure 4. As shown in the figure, all DNN based methods have lower total errors than other approaches. The advantages hold apparently for the first five speaker pairs. For the last speaker pair FE1-FE2, our approach performs comparably with GMM-FHMM but better than HMM. The two types of SD-DNNs perform almost the same across all speaker pairs, which implies that the regularization does not significantly boost the performance of the SD-DNN. On the other hand, sharing the same network structure with the RSD-DNN, the SPD-DNN outperforms both SD-DNNs, especially when two speakers are of the same gender.



(a) *Groundtruth pitch and estimated pitch by HMM*

(b) *Groundtruth pitch and estimated pitch by GMM-FHMM*

(c) *Groundtruth pitch and estimated pitch by RSD-DNN*

(d) *Groundtruth pitch and estimated pitch by SPD-DNN*

Figure 5: *Multipitch tracking results on a test sample (pbbv6n and priv3n) for the MA1-MA2 speaker pair.*

Such results are expected as SPD-DNNs are trained on speaker pairs and can accurately distinguish the differences between the speakers. In Figure 5, we illustrate pitch tracking results on a test sample of MA1-MA2. The HMM based approach fails to assign pitches to their underlying speakers. The GMM-FHMM based approach works well in terms of speaker assignment, but performs poorly when two pitch trajectories are close to each other. Moreover, the resulting pitch contour of GMM-FHMM lacks continuity. The RSD-DNN based method outperforms previous approaches in terms of pitch continuity, but it still has incorrect speaker assignment at some frames. The SPD-DNN based method generates very good pitch trajectories in terms of both pitch accuracy and speaker assignment.

5. Conclusion

We have proposed speake-dependent and speaker-pair-dependent DNNs to estimate the posterior probabilities of pitch states for multipitch tracking. In comparison to the GMM-FHMM structure [12], the use of DNNs allows us to estimate pitch states discriminatively. Experiments have shown that all DNN based methods outperform two state-of-the-art multipitch trackers. The speaker-pair-dependent DNN performs especially well when the test sample is generated by two speakers of the same gender.

6. Acknowledgements

This research was supported in part by an AFOSR grant (FA9550-12-1-0130) and the Ohio Supercomputer Center.

7. References

[1] K. Hu and D. L. Wang, "SVM-based separation of unvoiced-voiced speech in cochannel conditions," *Proceedings of ICASSP*, pp. 4545–4548, 2012.

[2] G. Hu and D. L. Wang, "A tandem algorithm for pitch estimation and voiced speech segregation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 18, pp. 2067–2079, 2010.

[3] X. Zhao, Y. Shao, and D. L. Wang, "CASA-Based Robust Speaker Identification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, pp. 1608–1616, 2012.

[4] C. Chen, R. Gopinath, M. Monkowski, M. Picheny, and K. Shen, "New methods in continuous Mandarin speech recognition," in *Proc. Eurospeech'97*, pp. 1543–1546, 1997.

[5] D. Talkin, W. B. Kleijn, and K. K. Paliwal, Eds, "A robust algorithm for pitch tracking (RAPT)," in *Speech Coding and Synthesis*, Amsterdam, The Netherlands: Elsevier, pp. 495–518, 1995.

[6] P. Boersma and D. Weenink, "Praat, a system for doing phonetics by computer," *Glot Int.*, vol. 5, pp. 341–345, 2001.

[7] M. Wu, D. L. Wang, and G. Brown, "A multipitch tracking algorithm for noisy speech," *IEEE Trans. Speech Audio Process.*, vol. 11, pp. 229–241, 2003.

[8] F. Sha and L. K. Saul, "Real-time pitch determination of one or more voices by nonnegative matrix factorization," in *Advances in Neural Information Processing Systems 17*, pp. 1233–1240, 2005.

[9] F. Bach, and M. Jordan, "Discriminative training of hidden Markov models for multiple pitch tracking," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, pp. 489–492, 2005.

[10] Z. Jin and D. L. Wang, "HMM-based multipitch tracking for noisy and reverberant speech," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, pp. 1091–1102, 2011.

[11] Z. Duan, J. Han, and B. Pardo, "Multi-pitch streaming of harmonic sound mixtures," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, pp. 138–150, 2014.

[12] M. Wohlmayr, M. Stark, and F. Pernkopf, "A probabilistic interaction model for multipitch tracking with factorial hidden Markov models," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, pp. 799–810, 2011.

[13] G. E. Hinton, S. Osindero, and Y. -W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[14] K. Han and D. L. Wang, "Neural network based pitch tracking in very noisy speech," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, pp. 2158–2168, 2014.

[15] M. Jordan, Z. Ghahramani, T. Jaakkola and L. K. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, pp. 183–233, 1999.

[16] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist. JMLR W&CP Volume*, vol. 15, pp. 315–323, 2011.

[17] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[18] Z. Ghahramani and M. Jordan, "Factorial hidden Markov models," *Machine Learning*, vol. 29, pp. 245–273, 1997.

[19] M. Cooke, J. Barker, S. Cunningham and X. Shao, "An audio-visual corpus for speech perception and automatic speech recognition," *J. Acoust. Soc. Amer.*, vol. 120, pp. 2421–2424, 2006.

[20] T. Drugman and A. Alwan, "Joint robust voicing detection and pitch estimation based on residual harmonics," in *Proc. Interspeech*, pp. 1973–1976, 2011.

[21] D. L. Wang and G. Brown, Eds., *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*. Hoboken, NJ: Wiley-IEEE Press, 2006.