DEEP DOMAIN ADAPTATION FOR POLYPHONIC MELODY EXTRACTION

Kavya Ranjan Saxena, Vipul Arora

Department of Electrical Engineering, Indian Institute of Technology Kanpur, India

ABSTRACT

Extraction of the predominant pitch from polyphonic audio is one of the fundamental tasks in the field of music information retrieval and computational musicology. To accomplish this task using machine learning, a large amount of labeled audio data is required to train the model that predicts the pitch contour. But a classical model pre-trained on data from one domain (source), e.g., songs of a particular singer or genre, may not perform comparatively well in extracting melody from other domains (target). The performance of such models can be boosted by adapting the model using some annotated data in the target domain. In this work, we study various adaptation techniques applied to machine learning models for polyphonic melody extraction. Experimental results show that meta-learning-based adaptation performs better than simple fine-tuning. In addition to this, we find that this method outperforms the existing state-of-the-art non-adaptive polyphonic melody extraction algorithms.

Index Terms— melody extraction, domain adaptation, fine-tuning, model agnostic meta-learning

1. INTRODUCTION

Extracting melody from polyphonic audio is a fundamental and important task in the music information retrieval field. The aim is to extract the pitch of the dominant singing voice from polyphonic audio. There are many downstream applications of melody extraction, including music recommendation [1], cover song identification [2], music generation [3] and voice separation [4]. Deep learning models provide excellent performance when sufficiently large annotated data is available in the source domain. But this performance degrades when these models are applied to different target domains with a very different data distribution as compared to source domains. In the audio domain, a model trained on source domain data, for example, songs of a particular singer or genre, may not perform comparatively well in extracting melody from other target domain data. Domain adaptation [5][6] caters to this problem, to enable the model trained on the source domain to perform well on the target domain. In this paper, we apply domain adaptation methods to the problem of polyphonic melody extraction. In particular, we study fine-tuning and meta-learning-based domain adaptation approaches. Meta-learning has been recently used to improve the performance of few-shot learning problems [7][8]. The common approaches to meta-learning are metric-based [7], model-based [8], and optimization-based [9] learning that improve the learning speed [10]. To the best of our knowledge, no such work on meta-learning-based domain adaptation for polyphonic melody extraction has been conducted as per the literature.

2. ADAPTIVE MELODY EXTRACTION

In this paper, we discuss two domain adaptation techniques, so that the model trained on the source domain is able to perform comparatively well in extracting melody from other target domains.

The audio waveforms are first merged into a mono channel and then downsampled to 8 kHz. By using a short-time Fourier transform, we calculate the magnitude spectrogram of the audio. The spectrogram is calculated using a 1024-point Hanning window and a hop size of 10 ms. The spectrogram (corresponding to the 5 second audio) is given as an input to the model which is classified into one of the 442 pitch classes, including a non-voiced class. The voiced pitch classes range from E2 (82.4 Hz) to B6 (1975.7 Hz) with a resolution of 1/8 semitone.

2.1. Fine-tuning based adaptation

One of the basic adaptation techniques is fine-tuning a pre-trained model [11]. In this technique, for a model \( f_{[\phi, \psi]} \) pre-trained on the source dataset \( D^S \), the feature extractor layers \( \phi \) are concatenated with a new classifier layer \( \psi \) (discarding the previous classifier layer \( \theta \)) whose weights are updated in the subsequent training over the target dataset \( D^T \). The weights of the rest of the model layers are frozen, i.e., not updated. During fine-tuning the weights of the model are updated by the gradient descent algorithm as follows:

\[
[\phi, \psi] \leftarrow [\phi, \psi] - \alpha \nabla_{[\phi, \psi]} L_{D^T}(f_{[\phi, \psi]}) \tag{1}
\]

where \( \alpha \) is the learning rate and \( L(.) \) is the categorical cross-entropy loss.
2.2. Meta-learning based adaptation

Another adaptation technique presented in this paper is the model agnostic meta-learning (MAML) based adaptation. MAML focuses on learning good initialization parameters for a model trained on the source domain so that it quickly adapts to the target domain with little training data. In this work, an audio segment of 5 seconds is considered an episode. Different audios differ in singers, genres, background instruments, etc. We aim at creating a model that generalizes well to any audio and improves further with little training data.

In the meta-training phase, a particular episode \( b \) is further divided into support set \( T^S_b \) and query set \( T^Q_b \). Let the model be \( f_{\theta_0} \) where \( \theta_0 \) are the trainable parameters that are initialized randomly. For a particular episode \( b \), the model is initialized as \( \theta^b_0 = \theta_0 \), and is updated from the support set \( T^S_b \) as follows:

\[
\theta^b_i = \theta^b_{i-1} - \alpha \nabla_{\theta^b_{i-1}} L_{T^Q_b}(f_{\theta^b_{i-1}})
\]

(2)

where \( \alpha \in \mathbb{R}^+ \) is the learning rate of the model, \( \theta^b_i \) are the updated weights of the model after \( i \) steps, \( L_{T^Q_b}(f_{\theta^b_{i-1}}) \) is the categorical cross-entropy loss on the support set after \( (i-1) \) update steps. After \( N \) update steps, the updated parameters become \( \theta^b_N \). This process of updating is called inner-loop optimization. The updated parameters \( \theta^b_N \) are used for inference on the query set \( T^Q_b \).

\( \theta_0 \) is updated using the loss over the query set. This process of updating \( \theta_0 \) is called outer-loop optimization which is expressed by:

\[
\theta_0 \leftarrow \theta_0 - \beta \nabla_{\theta_0} L_{T^Q_b}(f_{\theta^b_N})
\]

(3)

where \( \beta \in \mathbb{R}^+ \) is the learning rate and \( L_{T^Q_b}(f_{\theta^b_N}) \) is the categorical cross-entropy loss on the query set \( T^Q_b \).

In the meta-testing phase, given a new episode \( b' \), the updated \( \theta_0 \) is used to initialize the model as \( \theta^b_0 = \theta_0 \), thus enabling it to adapt to a new task with small support set \( T^S_{b'} \). The model performance is finally evaluated over the query set \( T^Q_{b'} \). The results are averaged over multiple new episodes to assess the generalizability of the model.

Algorithm 1 Meta-training Algorithm

**Require:** \( \alpha, \beta \) : learning rates

1. Observe the pre-trained model parameters \( \theta_0 \)
2. for all episodes \( b \) in meta-training dataset do
3. \( \theta^b_0 = \theta_0 \)
4. Calculate spectrogram \( X_b \) and obtain the output \( Y_b = f_{\theta^b_0}(X_b) \)
5. Sample the support set as the time frames \( T^S_b = \{t_1, \ldots, t_K\} \), where \( Y_b \) does not match the ground truth and the rest of the time frames as the query set \( T^Q_b \)
6. Compute the updated parameters \( \theta^b_N \) using \( \theta^b_0 \) with inner-loop optimization as in eq.2
7. Update \( \theta_0 \) using \( \theta^b_b \) by outer-loop optimization as in eq.3
8. end for

2.3. Methodology

There are three stages to applying MAML-based domain adaptation to the melody extraction problem.

In the first stage, we pre-train the model on the source audio dataset \( D^S \). First, the model \( f_{\theta_0} \) is randomly initialized. The last layer of the model is the classifier layer and the rest of the layers are the feature extractor layers. We use gradient descent to optimize the categorical cross-entropy loss.

In the second stage, we meta-train the model on the source audio dataset \( D^S \) itself. An audio segment of 5 seconds is considered an episode. For a particular episode \( b \), the model is initialized as \( \theta^b_0 = \theta_0 \). We calculate the spectrogram \( X_b \) by applying a short-time Fourier transform on the audio segment. The dimension of the calculated spectrogram is \( F \times M \), where \( F \) is the number of frequency bins and \( M \) is the number of time frames. With this spectrogram as the input, the model predicts an output \( Y_b = f_{\theta^b_0}(X_b) \) of dimension \( 442 \times M \). Here, \( M \) corresponds to the respective number of time frames of the spectrogram and 442 is the number of pitch classes. We compute the estimated pitch values using max over the columns of \( Y_b \) and compare them with the ground truth pitch values. From the incorrectly predicted time frames, we randomly select \( K \) time frames as the support set \( T^S_b = \{t_1, t_2, \ldots, t_K\} \) and the remaining \( M - K \) time frames as the query set \( T^Q_b \) of the episode. The inner-loop optimization is carried out by updating the parameters \( \theta^b_0 \) over the support set \( T^S_b \) as in eq.2. With the same spectrogram, \( X_b \), given as an input to the model with updated parameters \( \theta^b_N \), the model predicts an output \( Y_b = f_{\theta^b_N}(X_b) \). We compute the estimated pitch values using max over the columns of \( Y_b \) and compare them with the ground truth pitch values. The outer-loop optimization is carried out by updating \( \theta_0 \) over the query set \( T^Q_b \) as in eq.3. Fig. 1 depicts the entire meta-training framework for a single episode \( b \) as discussed. The inner-loop and outer-loop optimization is an iterative process that is applied to all the episodes present in the meta-training dataset \( D^S \). The de-
| Experiments | MIREX05 | ADC2004 |
|-------------|---------|---------|
|             | RPA     | RCA     | OA     | RPA     | RCA     | OA     |
| CT          | 76.03   | 78.25   | 76.05  | 75.48   | 76.28   | 75.48  |
| FTA         | 77.41   | 78.56   | 77.41  | 76.16   | 77.57   | 76.28  |
| ML($\theta_0$) | 81.20   | 82.76   | 82.42  | 82.90   | 83.50   | 82.88  |
| ML($\theta_N^b$) | 83.40   | 84.31   | 84.57  | 85.39   | 86.88   | 85.95  |

Table 1. Comparison between different adaptation-based methods on MIREX05 and ADC2004 dataset.

In the third stage, we meta-test the model on the target audio dataset $D_T$. After meta-training which is a two-stage optimization process, the updated model parameters $\theta_0$ now act as good initialization parameters for the model that adapts to the audios of different singers or genres. An audio segment of 5 seconds is considered an episode. For a particular episode $b$, the model is initialized as $\theta_0^b = \theta_0$. We calculate the spectrogram $X_b$ by applying a short-time Fourier transform on the audio segment. With this spectrogram as the input, the model predicts an output $Y_b = f_{\theta_0^b}(X_b)$. We compute the estimated pitch values using max over the columns of $Y_b$ and compare them with the ground truth pitch values. From the incorrectly predicted time frames, we randomly select $K$ time frames as the support set $T_S^b = \{t_1, t_2, ..., t_K\}$ of the episode. The inner-loop optimization is carried out by updating the model parameters $\theta_N^b$ over the support set $T_S^b$ as in eq. [2]. In this way, the model adapts to a few support points from the target audio segment. The performance of the model with the updated parameters $\theta_N^b$ is finally evaluated on the same episode. The results are averaged over all episodes in $D_T$.

3. EXPERIMENTS

3.1. Data

For the melody extraction task, we have used MIR1K data (source data $D_S$) as the training data which contains 1000 Chinese karaoke clips. The total training dataset consists of about 22 hours of audio. No data augmentation is performed. We have tested the performance of our model on two datasets (target data $D_T$): ADC2004 and MIREX05. The proposed model is only trained for singing voice melody, so we have selected only those test samples that contained melody sung by humans. As a result, 12 clips in ADC2004 and 9 clips in MIREX05 are selected.

3.2. Experiment setting

In this paper, we employ a basic deep CNN model to carry out the melody extraction task. The deep CNN model consists of 5 convolutional layers having $[32, 64, 128, 64, 32]$ filters respectively, with the first and last layers having a kernel size of $5 \times 5$ and the intermediate layers having a kernel size of $3 \times 3$. These convolutional layers are the feature extractor layers followed by a classifier layer having 442 nodes corresponding to the pitch classes. We use the softmax activation function and categorical cross-entropy loss to update the parameters of the model. An audio segment of 5 seconds is considered an episode. We apply a short-time Fourier transform on the audio segment. The dimension of the calculated spectrogram is $F \times M$, where $F = 512$ and $M = 500$. We perform four different experiments to analyze the performance of different adaptation-based techniques as compared to standard pre-training for polyphonic melody extraction. For the classical training experiment, the weights of all the layers of the model are updated. For the rest of the experiments, the first four layers of the model remain frozen and the weights of the rest of the layers are updated. The experiments are described as follows:

CT: Classical training - We pre-train the deep CNN model on all the audios of MIR1K data. We train our model for 400 epochs with a learning rate of $1 \times 10^{-5}$. No adaptation on the test data is performed. The model with the updated parameters is used to evaluate the performance on the test datasets.

FTA: Fine-tuning based adaptation - We pre-train the deep CNN model with all the audios of MIR1K data and then fine-tune on both the test datasets - MIREX05 and ADC2004. We calculate a spectrogram of dimension $512 \times 500$ by applying a short-time Fourier transform on an audio segment. After computing the estimated pitch values using the max over the 500 columns, we consider $K = 5$ time frames where the estimated pitch values do not match the ground truth pitch values. We fine-tune the model on these frames as in eq. [1]. We take $\alpha = 1 \times 10^{-5}$. After fine-tuning, the model with the updated parameters is used to evaluate the performance on the test datasets.

| Experiments | RPA   | RCA   | OA   |
|-------------|-------|-------|------|
| Patch-based CNN[12] | 81.20 | 82.20 | 73.20 |
| Attention Network[13] | 77.80 | 77.80 | 84.40 |
| SegNet[14] | 78.40 | 79.70 | 78.60 |
| ML($\theta_N^b$)(ours) | 83.40 | 84.31 | 84.57 |

Table 2. Experimental results on MIREX05

| Experiments | RPA   | RCA   | OA   |
|-------------|-------|-------|------|
| Patch-based CNN[12] | 76.70 | 78.40 | 74.30 |
| Attention Network[13] | 76.30 | 76.50 | 77.40 |
| SegNet[14] | 82.70 | 84.90 | 81.60 |
| ML($\theta_N^b$)(ours) | 85.39 | 86.88 | 85.95 |

Table 3. Experimental results on ADC2004
**Fig. 2.** Comparison between the three experiments. The first figure is the prediction after classical training (CT), the second figure represents the prediction after fine-tuning (FTA) and the third figure represents the prediction after meta-testing (ML($\theta^b_N$)). The results are produced on an audio segment of the audio daisy2.wav.

**ML($\theta_0$): Meta-learning without adaptation** - We pre-train the model with the first 800 songs of MIR1K. The remaining 200 songs are used for meta-training. The model learns the episode-specific knowledge by updating the weights on audio episodes.

During meta-training, for the inner-loop optimization, we consider a total of $K = 5$ time frames as the support set. We take $N = 5$ and $\alpha = 1 \times 10^{-5}$ in eq. [2]. For outer-loop optimization, we consider the remaining $M - K = 495$ time frames as the query set. We take $\beta = 1 \times 10^{-5}$. The model with the updated parameters $\theta_0$ is now used to evaluate the performance on the test datasets.

**ML($\theta^b_N$): Meta-learning with adaptation** - We pre-train the model with the first 800 songs of MIR1K. The remaining 200 songs are used for meta-training. We use the test datasets MIREX05 and ADC2004 for meta-testing.

During the meta-training phase, for the inner-loop optimization, we consider a total of $K = 5$ time frames as the support set. We take $N = 5$ and $\alpha = 1 \times 10^{-5}$ in eq. [2]. For outer-loop optimization, we consider the remaining $M - K = 495$ time frames as the query set. We take $\beta = 1 \times 10^{-5}$.

During the meta-testing phase, for the inner-loop optimization, we consider a total of $K = 5$ time frames as the support set. We take $N = 5$ and $\alpha = 1 \times 10^{-5}$ in eq. [2]. The model with updated parameters $\theta^b_N$ is used to evaluate the performance on the test datasets.

The performance metrics considered are raw pitch accuracy (RPA), raw chroma accuracy (RCA), and overall accuracy (OA). All these metrics are computed by using a standard mir-eval library with a pitch detection tolerance of 50 cents.

### 3.3. Results

Results of the experiments performed on the employed model are shown in Table 1. We can see that FTA performs better than CT, but ML($\theta_0$) performs better than FTA. We also see that ML($\theta^b_N$) outperforms all three methods on both test datasets. From this, we can infer that ML($\theta_0$) can generalize well to any audio and improves further with adaptation on little training data as in ML($\theta^b_N$). Table 2 and Table 3 show the comparison of our meta-learning-based adaptation method ML($\theta^b_N$) to baseline methods on both the test datasets and it is observed that our method outperforms the baseline methods. It is to be considered that none of the baseline methods are adapted to the support set of the test datasets. From this work, we infer that meta-learning-based adaptation when applied to a simple deep CNN model, can improve the results significantly. In general, meta-learning-based adaptation is model-agnostic and can be applied to complex models such as [13] to improve the results further. A more vivid representation of different experiments can be seen in fig. 2. We can visually compare the performance of different methods in the estimation of pitch contour. Although the estimation with FTA has improved as compared to that with CT, the performance has degraded on those points which were correctly predicted by CT (between 1 and 2 seconds). However, ML($\theta^b_N$) has not only improved the overall prediction as compared to CT, but it is also able to retain the correct predictions with CT.

### 4. CONCLUSION

In general, domain adaptation improves the performance of the polyphonic melody extraction model. But a specific domain adaptation method, namely model-agnostic meta-learning (MAML), brings in the best performance among all the discussed methods. The MAML algorithm is model-agnostic and can be applied to any model to improve its performance. The methods discussed can be applied to interactive pitch estimation which can help in speeding up the annotation process. Its applications also extend to the field of pedagogy and music synthesis.

### 5. ACKNOWLEDGMENT

This work was supported by an IMPRINT-2C grant from SERB, Government of India, and grant no. PB/EE/2021128B from Prasar Bharti. Project no. IMP/2019/400.
6. REFERENCES

[1] K. Chen, B. Liang, X. Ma, and M. Gu, “Learning audio embeddings with user listening data for content-based music recommendation,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021, pp. 3015–3019.

[2] X. Du, K. Chen, Z. Wang, B. Zhu, and Z. Ma, “Bytecover2: Towards dimensionality reduction of latent embedding for efficient cover song identification,” in ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022, pp. 616–620.

[3] K. Chen, C.-i. Wang, T. Berg-Kirkpatrick, and S. Dubnov, “Music sketchnet: Controllable music generation via factorized representations of pitch and rhythm,” arXiv preprint arXiv:2008.01291, 2020.

[4] Y. Ikemiya, K. Yoshii, and K. Itoyama, “Singing voice analysis and editing based on mutually dependent f0 estimation and source separation,” in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015, pp. 574–578.

[5] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, “Analysis of representations for domain adaptation,” Advances in neural information processing systems, vol. 19, 2006.

[6] K. Bouwmals, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, “Domain separation networks,” Advances in neural information processing systems, vol. 29, 2016.

[7] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra et al., “Matching networks for one shot learning,” Advances in neural information processing systems, vol. 29, 2016.

[8] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” 2016.

[9] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in International conference on machine learning. PMLR, 2017, pp. 1126–1135.

[10] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, “Learning to learn by gradient descent by gradient descent,” Advances in neural information processing systems, vol. 29, 2016.

[11] E. Cetinic, T. Lipic, and S. Grgic, “Fine-tuning convolutional neural networks for fine art classification,” Expert Systems with Applications, vol. 114, pp. 107–118, 2018.

[12] L. Su, “Vocal melody extraction using patch-based cnn,” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 371–375.

[13] S. Yu, X. Sun, Y. Yu, and W. Li, “Frequency-temporal attention network for singing melody extraction,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021, pp. 251–255.

[14] T.-H. Hsieh, L. Su, and Y.-H. Yang, “A streamlined encoder/decoder architecture for melody extraction,” in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 156–160.

[15] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, “mir_eval: A transparent implementation of common mir metrics,” in In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR. Citeseer, 2014.