Deformable CNN and Imbalance-Aware Feature Learning for Singing Technique Classification

Yuya Yamamoto¹, Juhan Nam², Hiroko Terasawa¹

¹Doctoral Program in Informatics, University of Tsukuba, Japan
²Graduate School of Culture Technology, KAIST, South Korea

s2130507@s.tsukuba.ac.jp, juhan.nam@kaist.ac.kr, terasawa@slis.tsukuba.ac.jp

Abstract

Singing techniques are used for expressive vocal performances by employing temporal fluctuations of the timbre, the pitch, and other components of the voice. Their classification is a challenging task, because of mainly two factors: 1) the fluctuations in singing techniques have a wide variety and are affected by many factors and 2) existing datasets are imbalanced. To deal with these problems, we developed a novel audio feature learning method based on deformable convolution with decoupled training of the feature extractor and the classifier using a class-weighted loss function. The experimental results show the following: 1) the deformable convolution improves the classification results, particularly when it is applied to the last two convolutional layers, and 2) both re-training the classifier and weighting the cross-entropy loss function by a smoothed inverse frequency enhance the classification performance.

1. Introduction

Professional singers express their characteristics and emotions by various singing techniques such as vibrato and breathy voice effects. At the signal level, singing techniques are observed as time–frequency textures, e.g., strong temporal modulation related to harmonics (“vibrato”) and highly noisy components over broad frequency bands (“breathy voice”). Automatic classification of singing techniques is an emerging research topic in singing voice analysis [1].

One of the main problems in this task is extracting features from highly dynamic time–frequency textures of singing techniques. Convolutional neural networks (CNNs) have been recently used as effective methods to capture audio features for singing technique classification [2, 3, 4] as well as similar objectives such as musical playing technique recognition [5]. Although square-shaped kernels, e.g., 3×3 and 5×5, are commonly used in CNNs, it has been shown that customizing the kernel shape improves the classification performance. For example, in a study, oblong-shaped kernels outperformed square-shaped ones for singing technique classification [3]. Similar results have been found in music auto-tagging [6] and musical instrument classification [7]. The above findings suggest that more customized kernels may further improve the performance. However, a brute-force search toward the best kernel shape will be burdensome, and thus, a systemic approach is required.

Another critical problem in singing technique classification is data imbalance, which is mainly attributed to the nature of voice production and musical usage. For example, “vocal fry” and “trillo” are difficult to produce for a long time, and thus, the lengths of such audio samples tend to be relatively short. In addition, “belting” is obtained in only certain musical contexts. Thus, collecting well-balanced samples is problematic.

In this study, we deal with the above two problems by deformable convolution and classifier re-training (cRT) using a class-weighted loss, respectively. Deformable convolution allows the convolution kernel to have a flexible shape [8]. It extends the capability of a CNN by modeling geometric transformation, which can be beneficial in capturing dynamic time–frequency features in singing techniques. cRT decouples the feature extractor and the classifier in training a deep neural network model. It was reported as a simple yet powerful method when the class distribution of the training data has a long tail [9].

The contributions of this study are as follows: 1) We investigate different setups of deformable convolution and show that it improves the singing technique classification performance. 2) We show the effectiveness of cRT for an imbalanced dataset, comparing with joint training of the feature extractor and the classifier. 3) Finally, we present that smoothed weighting the loss function further enhances the effect of decoupled training.

2. Related Work

2.1. Deformable Convolution

Deformable convolution was introduced for image processing to enhance the transformation modeling capability of a CNN [8, 10]. It allows CNN models to only focus on what they are interested in and makes the output feature maps more representative. It has been effective in several tasks involving variations in the temporal context, such as action recognition [11], sign language recognition [12], and video captioning [13]. In the audio domain, deformable convolution has been employed in speaker verification [14] and speech recognition [15], to deal with the variable temporal dynamics of speech. In this study, we apply deformable convolution to singing technique classification.

2.2. Data Imbalance

Data imbalance is a common issue in classification tasks. There are two well-known approaches for solving this problem: sampling and cost-sensitive learning [16]. Sampling manipulates the class representations in an original dataset by either over-sampling the minority classes (over-sampling) or under-sampling the majority classes (under-sampling). In the context of deep learning, neither over-sampling nor under-sampling is efficient; over-sampling degrades the training and may cause overfitting, whereas under-sampling may discard informative majority examples [17]. Cost-sensitive learning is a type of learning that considers the misclassification costs. A simple approach of cost-sensitive learning is reweighting the loss function using inverse class frequency values [18]. However, this strategy may perform poorly when applied to real-world
3. Method

Figure 1 shows an overview of our proposed method, and this section describes the details of each of its parts.

3.1. Deformable Convolution

Deformable convolution (DC) facilitates trainable offset parameters of each kernel to deform the convolutional kernel grid. Deformable convolution consists of the following steps: 1) obtain the offset field, 2) output deformable feature maps by the offsets, 3) perform regular convolution on the deformable feature maps. The middle of Figure 1 illustrates the operation of deformable convolution.

1. The offset field is obtained by applying a convolutional layer over the input feature map with channel dimension $N$. The offset field has the same spatial resolution as the input feature map, and the channel dimension is $2N$. Horizontal offset $\Delta x$ and vertical offset $\Delta y$ correspond to each point of the input feature map.

2. Because the offset parameters ($\Delta x$ and $\Delta y$) are typically fractional, the values of offset location are interpolated around the value of closest four points by bilinear interpolation [8].

3. The output feature maps are obtained by operating a regular convolution using the deformable feature maps. For each location $p_0$ on the input feature map $x$, and the output feature map $y$,

$$ y(p_0) = \sum_{m \in R} w(p_0) \cdot x(p_0 + p_m + \Delta p_m) $$

where $w$, $R$, and $p_m$ denote the weight of the sampled values, kernel grid with size $M$, and interpolated offset value, respectively.

We choose a four-oblong-shaped convolution layer CNN with a multi-resolution spectrogram input, which was the best performing model in [3], for the base architecture. The model consists of four convolutional blocks, a global average pooling layer [21], and two fully connected layers.

Note that although its kernel shapes are unidirectional (i.e., $\{Vertical, Horizontal\} = \{(4 \times 1), (16 \times 1), (1 \times 4), (1 \times 16)\}$), we consider both vertical and horizontal offsets as same, similar to conventional studies [8, 10, 14], to preserve flexibility.

3.2. Weighting Loss Function

We apply a smoothed weighting to the cross-entropy loss function during training, to deal with the data imbalance problem.

$$ L(x, y) = -W \log \frac{\exp(x_{n,c,y})}{\sum_{c=1}^{C} \exp(x_{n,c})} \quad (1) $$

where $x$ is the input, $y$ is the target, and $W$ is the weight of the loss function. We determine the loss weight of each class $w_c$ by the power of the inverse frequency of the training sample as follows:

$$ w_c = \frac{1}{(n_c)\alpha} \quad (2) $$

where $n_c$ is the number of training samples in class $c$, and $\alpha$ is the smoothing factor, controlling smoothing of the loss weights. Note that $\alpha = 0$ corresponds to the value of 1 (i.e., no weighting) and $\alpha = 1$ corresponds to a reciprocal number (i.e., weighting by the inverse class frequency).

3.3. Decoupling Feature Extractor and Classifier

We also investigate decoupling the feature extractor and the classifier from the CNN model, following the method of Kang

\[\text{In the original study, a flatten layer was used in the top part of the feature extractor. However, in singing technique classification, we confirmed that the global average pooling layer generally outperforms the flatten layer.}\]
et al. [9]. They proposed two different approaches of the de-
coupled training method: cRT and normalizing the weights
of the classifier by its own norms scaled by a hyperparameter
(τ-normalized classifier), called learnable weight scaling, and
showed that both outperformed joint training of the classifica-
tion model. We choose to employ cRT, which was reported as
a simple but effective training strategy for an imbalanced dataset.
First, the layers of the model are divided into two parts—the
feature extractor and the classifier—between the first and sec-
two fully connected layers. In the training stage, first the model
is trained regularly and subsequently the classifier is re-trained
after fixing the weights of the feature extractor part. The right
panel of Figure 1 illustrates the training strategy.

4. Experiments

4.1. Dataset

We use VocalSet [2], which is the only publicly available dataset
for studies on singing techniques. The dataset contains singing
voices of 20 different professional singers (9 female and 11
male) performing 17 different singing techniques in various
contexts, such as arpeggio, scale, and long tones. For the
classification experiments, we select the samples corresponding
to ten different singing techniques (“belt,” “breathy,” “inhaled
singing,” “lip trill,” “spoken excerpt,” “straight tone,” “trill,”
“trillo,” “vibrato,” and “vocal fry”). Figure 2 shows the total
length of each singing technique. The distribution of the dataset
has a long-tail shape, i.e., it is imbalanced.

During the learning process, we split the dataset into a train-
ing set of 15 singers and a test set of 5 singers. Subsequently,
we segment the audio signals in each file into 3-second audio
clips and nonoverlapping parts at a sample rate of 44.1 kHz. We
evaluate each model using five metrics: macro-F1 score (F1),
balanced accuracy (B-Acc.), accuracy (Acc.), top-2 accuracy,
and top-3 accuracy.

4.2. Model

We set up four types of deformable convolution model (DC)
with weighting and two models without deformable convolution
(w/o DC) with or without weighting. As a result, we compare
six conditions in total. For all of these six conditions, the model
input and structure are common as follows. The model input is
multi-resolution spectrogram (i.e., stacking three spectrograms
with different time-frequency resolutions along the channel di-
mension.) We obtain them by short-time Fourier Transform
(STFT), and each spectrogram is obtained by the three window
sizes of (2048, 1024, 512 samples) with the same hop length
512 samples and the STFT length 2048 samples with zero-
padding. We employed a four-oblong-shaped convolution layer
CNN [3] for the model structure. Each convolution block cons-
ists of a convolution layer (Conv), a batch normalization layer,
a Rectified Linear Unit (ReLU), a max pooling (MP) layer, and
a dropout of 0.3. They are followed by a global average pool-
ing (Global AP) layer and two fully-connected layers (FC). We
trained our model using the Adam optimizer with a learning rate
of 1e-4 and a batch size of 64.

The four DC conditions are denoted as All, Early, Late, and
Last and their components are listed in Table 1. DC is applied
to different layers. All DC models are trained with the weighted
loss function. For the non-DC models, we considered two w/o
DC conditions with or without weighting, they are referred to
as w/o DC weighted and w/o DC plain.

4.3. Experiment 1: Effect of Deformable Convolution

We investigate the effect of deformable convolution by replac-
ing standard convolution layers of the model with deformable
convolution layers. We tested the six conditions as described
in Section 4.2. As baselines, we use one-dimensional CNN
(1DCNN) [2] and oblong-CNN feature learning with a random
forest classifier (Oblong) [3]. We re-implement the models to
investigate the effect of weighting the loss function. For both
1DCNN and Oblong, we tested both weighted and plain (with-
out weighting) conditions. The number of parameter of each
conditions are as follows; w/o DC: 337.5k, All: 463.3k, Early:
362.2k, Late: 438.7k, and Last: 435.7k, respectively.

4.4. Experiment 2: Comparative analysis of training strategy

and α

We compare three training strategies with a set of smoothing
factors α (0, 0.2, 0.5, and 1) in Eq. 2 seeking the best DC setup.

- **Joint training**: without classifier retraining.
- **cRT-WFC**: weights are applied during both feature
  representation training and cRT phases.
- **cRT-WC**: weights are applied only during the cRT
  phase. (i.e., weights are not applied in the feature rep-
  resentation training phase)

These training strategies were tested upon the **Late** model
because it was the best model in experiment 1 as described in
Section 5.1. For reasonable comparison, the sum of the num-
ber of training epochs is set equal in all conditions. We set 200

<table>
<thead>
<tr>
<th>Layer Configuration</th>
<th>Ch</th>
<th>Deformable Conv</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>w/o DC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All Early Late</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv(4 × 1), MP(4 × 4)</td>
<td>32</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Conv(16 × 1), MP(4 × 4)</td>
<td>64</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Conv(1 × 4), MP(3 × 3)</td>
<td>128</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Conv(1 × 16), MP(2 × 2)</td>
<td>128</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td>Global AP</td>
<td>128</td>
<td>–</td>
</tr>
<tr>
<td>FC (Feature)</td>
<td>30</td>
<td>–</td>
</tr>
<tr>
<td>FC (Softmax)</td>
<td>10</td>
<td>–</td>
</tr>
</tbody>
</table>
Table 2: The results of experiment 1.

<table>
<thead>
<tr>
<th>Models</th>
<th>F1</th>
<th>Acc.</th>
<th>B-Acc.</th>
<th>Top-2</th>
<th>Top-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1DCNN [2] plain</td>
<td>0.488</td>
<td>0.584</td>
<td>0.484</td>
<td>0.764</td>
<td>0.863</td>
</tr>
<tr>
<td>1DCNN [2] weighted</td>
<td>0.306</td>
<td>0.439</td>
<td>0.352</td>
<td>0.643</td>
<td>0.753</td>
</tr>
<tr>
<td>Oblong [3] plain</td>
<td>0.540</td>
<td>0.600</td>
<td>0.597</td>
<td>0.757</td>
<td>0.838</td>
</tr>
<tr>
<td>Oblong [3] weighted</td>
<td>0.548</td>
<td>0.590</td>
<td>0.613</td>
<td>0.759</td>
<td>0.852</td>
</tr>
<tr>
<td>w/o DC plain</td>
<td>0.404</td>
<td>0.402</td>
<td>0.472</td>
<td>0.686</td>
<td>0.805</td>
</tr>
<tr>
<td>w/o DC weighted</td>
<td>0.513</td>
<td>0.554</td>
<td>0.575</td>
<td>0.743</td>
<td>0.858</td>
</tr>
<tr>
<td>All (1,2,3,4)</td>
<td>0.553</td>
<td>0.604</td>
<td>0.59</td>
<td>0.799</td>
<td>0.896</td>
</tr>
<tr>
<td>Early (1,2)</td>
<td>0.554</td>
<td>0.593</td>
<td>0.598</td>
<td>0.776</td>
<td>0.862</td>
</tr>
<tr>
<td>Late (3,4)</td>
<td>0.582</td>
<td>0.623</td>
<td>0.641</td>
<td>0.806</td>
<td>0.894</td>
</tr>
<tr>
<td>Last (4)</td>
<td>0.517</td>
<td>0.572</td>
<td>0.607</td>
<td>0.764</td>
<td>0.846</td>
</tr>
</tbody>
</table>

epochs for the entire training time. For all cRT-based methods, we assign 100 epochs for the joint training of the feature extractor and the classifier, and the remaining 100 epochs for the cRT.

5. Results and Discussions

5.1. Effect of Deformable Convolution

The results of experiment 1 are listed in Table 2. They show that DC models significantly improve the classification performance compared to w/o DC models. Among the four DC setups, the Late model achieves the best. This agrees with the results from previous works that applying DC to several late convolution layers is effective [10, 15]. Compared to the Last model where DC is applied only to the last convolution layer, the accuracy of the Late model becomes much higher. Class-wise accuracy may explain this gap: With the Late model we observed large accuracy increments on the discrimination of “lip trill” and “vocal fry,” which have fine temporal modulation in amplitude, frequency, and breathiness.

This indicates that the small kernel size of the 3rd DC layer plays an important role when the dynamic offset adapts the fine modulation of singing voice. The baseline model with Oblong kernel-shapes achieves higher accuracy than the model without DC, as it uses a random forest classifier on a similar configuration of CNN feature extractor. However, the Late model extracts the features more effectively with DC and outperforms the baseline model.

5.2. Effect of cRT

Table 3 shows the results for the training strategies comparison, summarizing the output with the smoothing factor $\alpha = 0.2$ (as discussed in Section 5.3.) Both cRT methods outperform the joint-training method. Between two cRT methods, cRT-WC significantly improves the classification performance. This suggests that the weighting loss-function is only effective in cRT and so it is better to apply the weighting only during the retraining phase. A similar result was also reported in [9].

Table 3: The results of comparison between joint-training, cRT-WC, and cRT-WFC, under $\alpha = 0.2$.

<table>
<thead>
<tr>
<th>Methods</th>
<th>F1</th>
<th>Acc.</th>
<th>B-Acc.</th>
<th>Top-2</th>
<th>Top-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint-training</td>
<td>0.559</td>
<td>0.610</td>
<td>0.635</td>
<td>0.774</td>
<td>0.874</td>
</tr>
<tr>
<td>cRT-WF</td>
<td>0.582</td>
<td>0.623</td>
<td>0.641</td>
<td>0.806</td>
<td>0.894</td>
</tr>
<tr>
<td>cRT-WC</td>
<td>0.620</td>
<td>0.656</td>
<td>0.655</td>
<td>0.815</td>
<td>0.887</td>
</tr>
</tbody>
</table>

5.3. Effect of Smoothing Factor $\alpha$

We conducted experiment 2 with four different values of the smoothing factor $\alpha$: 0, 0.2, 0.5, and 1. Figure 3 plots Macro-F1 over the smoothing factor. The best performing condition is cRT-WC with an $\alpha$ value of 0.2. As $\alpha$ increases, the performance keeps decreasing in all three conditions and reaches the worst accuracy at an $\alpha$ value of 1 (i.e., inverse-frequency weight).

Increasing $\alpha$ has the expected effect of improving performance of minority classes while hurting majority classes. However, when we vary $\alpha$ from 0.2 to 1, the class-wise F1 scores decreased for both minority (e.g., “inhaled” 0.293 → 0.268, “trill” 0.544 → 0.495) and majority (e.g., “straight” 0.69 → 0.645, “vibrato” 0.648 → 0.623.) It corresponds to the result of conventional works [19, 17] that inverse frequency weight decreased the performance in large-scale long-tail classification problems.

This indicates that classification difficulty comes from not only data imbalance but also similarity between class samples, e.g., “vibrato” (majority class) and “trill”, “trillo” (minority classes). These techniques exhibit both frequency and amplitude modulations, while vibrato and trill mainly rely on frequency modulation and trillo on amplitude modulation. However, close observation of trillo spectrogram also shows some frequency modulation [2]. Detecting these subtle balance of amplitude and frequency modulations was the difficulty in this task.

6. Conclusion

In this paper, we proposed audio feature learning by deformable convolution and imbalance-aware learning based on classifier decoupling and a weighted inverse frequency loss, for singing technique classification. The experiments showed that applying deformable convolution in the last two layers and cRT with smoothed inverse frequency weights improve the classification performance. Future study can explore more complex weighting-based loss functions (e.g., [17]) and evaluating our concept on real-world singing performances, in which the problems of this study (i.e., feature learning and label sparseness [22]) are more serious.

7. Acknowledgements

This work was supported by JST SPRING, Grant Number JP-MJSP2124.
8. References


