HUMAN-IN-THE-LOOP ADAPTATION FOR INTERACTIVE MUSICAL BEAT TRACKING

Kazuhiko Yamamoto
YAMAHA Corporation

ABSTRACT

In music information retrieval (MIR), beat-tracking is one of the most fundamental and important task. However, a perfect algorithm is difficult to achieve. In addition, there could be a no unique correct answer because what one interprets as a beat differs for each individual. To address this, we propose a novel human-in-the-loop user interface that allows the system to interactively adapt to a specific user and target music. In our system, the user does not need to correct all errors manually, but rather only a small portion of the errors. The system then adapts the internal neural network model to the target, and automatically corrects remaining errors. This is achieved by a novel adaptive runtime self-attention in which the adaptable parameters are intimately integrated as a part of the user interface. It enables both low-cost training using only a local context of the music piece, and by contrast, highly effective runtime adaptation using the global context. We show our framework dramatically reduces the user's effort of correcting beat tracking errors in our experiments.

1. INTRODUCTION

Musical beat is among the most important factor in music [1]. Many MIR systems first analyze the beats as the starting point, assume the results would be correct, and use them as a unit for further processing [2–4]. However, although the performance of many MIR algorithms, including beat tracking, has improved dramatically through recent deep learning-based approaches [5–10], a perfect algorithm is difficult to achieve in principle, and errors are bound to occur. In addition, what one interprets as a beat differs for each individual. There is thus no unique answer to the question *"What is the correct beat?"* for a music. Many existing music pieces also show that the beats we want vary from time to time [11, 12]. This means that the ideal beat tracking system needs to produce different outputs from a single input depending on the situation. This is difficult to deal with using machine learning systems.

To address this, we propose an interactive beat-tracking interface for adapting to a specific user and target music using a human-in-the-loop approach (Figure 1). In this system, the user provides feedback for the temporal result

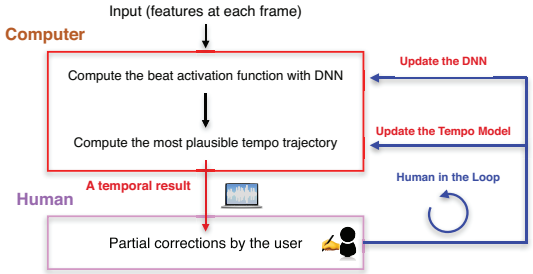

Figure 1. An overview of the system.

of the system by correcting the error, and the system then computes the output again. By iterating this interaction between the user and computer, the system adapts the internal neural network model (DNN) to the user, allowing it to produce more desirable results for the user. The user does not need to correct all errors manually, but rather only a small portion of the errors that are noticed. The system adapts the internal DNN to follow the user's intention and automatically corrects the remaining errors that the user has not corrected directly. This enables a dramatic reduction in the user's effort to obtain the desired result[1] .

To achieve such an online adaptation, we present a novel adaptive runtime self-attention model (ARSA), in which the adaptable parameters are intimately integrated as a part of the interactive user interface. This is a variant of the self-attention mechanism [13], and similarly to other models, it connects the entire input sequence (intermediate feature sequence) globally throughout the target music piece by the attention map. However, the most significant difference in the ARSA is not to be pre-trained at all. We embed the ARSA into an another pre-trained neural network only at runtime and use it for adaptation. This means that our model architecture changes between the training and runtime.

Our model is trained using only a local context for the training dataset and adapts to the user using the global context at runtime. This locally-aware learning reduces the computational cost during training, and the globally-aware runtime adaptation allows the effects of locally modified user feedback to be distributed throughout the entire piece of music. This strategy is based on our assumption that the amount of local feedback that the user can practically provide could be insufficient for adaptation. To assess the feasibility and effectiveness of the proposed system, we validate it through a simulation environment and a user study. In addition, we show the extent to which the proposed system improves the efficiency of error correction in beat tracking.

[1] The supplemental materials: www.yamo-n.org/hilbeats

2. RELATED WORK

2.1 Musical Beat Tracking

The state-of-the-art techniques in beat tracking are machine learning-based, and use a two-stage processing. They first compute the beat activation function by a DNN, which is a probability-like representation of the beat distribution, from which they then determine the actual beats by estimating the most plausible tempo trajectory throughout the target music. Böck and Schedl [6] use a bidirectional long short-tern memory neural network (BLSTM) to compute the beat activation function. Davies and Böck [5] use temporal convolutional networks (TCN) to improve the computational efficiency and the quality of the activations. To determine the actual beat positions from the beat activation function, Krebs et al. [14] use a dynamic Bayesian network (DBN), which is approximated as a hidden Markov model (HMM) solved using the Viterbi algorithm. Some studies have also reported that the combined modeling of the beat, downbeat, and tempo improves the beat tracking quality [7, 9]. The desired beat tends to depend on the genre. Böck et al. [8] use multiple pre-trained models adapted to different types of musical genres, and select one of them at runtime to the most plausible result. However, it is difficult to guarantee whether the classification of the genre would be appropriate.

2.2 Human-in-the-Loop System in MIR

The concept of human-in-the-loop involves humans as a part of the system. Sonic Visualizer [15] and Dixon [16] present interactive editing tools for beat tracking. They visualize the beats, and the user manually corrects the errors. However, such manual corrections are a burden to the user. Driedger et al. [17] propose a hybrid method for annotating beats by manual tapping and through the use of an automatically computed beat activation function. They correct the inaccurate tap to be closer to the activation function. However, this would be difficult to use if the activation is not sufficiently close to the desired result. Nakano et al. [18] attempt to improve the accuracy of the singing voice separation from mixed audio by fine-tuning a trained DNN at runtime. They use the user-corrected F0 curve as the new training data for the DNN. This concept is similar to that of our ours. However, using only a local correction that the user can provide through practical means might be insufficient for proper model adaptation.

Songle [19] provides a web-based error correction tool for several musical factors, including beats, by using crowdsourcing contributions. HumanGAN [20] also uses crowdsourced human evaluations as an evaluation function to determine whether a voice is realistic or not when it trains a singing voice synthesis DNN. However, we can not obtain the desired result for a specific user by applying these approach. Bryan et al. [21] propose an interactive sound source separation method by masking the spectrograms on the GUI. The system uses paintable masks to adapt the internal algorithm. Bazin et al. [22] also present an inpainting-based control on the spectrogram for sound synthesis using token-masked Transformers [13] and VQ-VAE [23]. Zhou et al. [24] use Bayesian optimization to efficiently explore the latent space of the melody generative model and obtain the desired output.

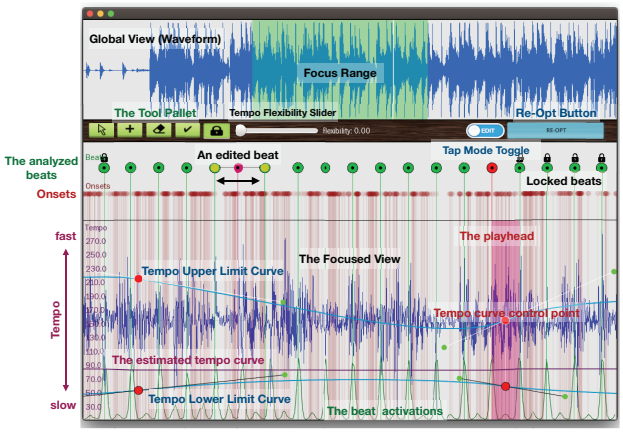

Figure 2. A graphical user interface for the system.

Bongjun and Bryan [25] propose an interactive interface for an effective sound event annotation. When the user makes a selection, several similar candidate positions are presented. The user listens to them and labels them as positive if they are actually the target, and otherwise labels them as negative. The system uses this feedback to update the similarity search model such that it shows more similar candidate locations. Our approach is partly inspired by them, and uses a similar method for treating the beats that are expected to be significantly affected by a user-modification (§.5.3).

3. USER INTERFACE

When the user inputs the target music, the system first displays the result of the beat tracking on the UI screen as the initial state (Figure 2). On the screen, the waveforms of the entire piece of music (top row) and a focused area (bottom row) are displayed. The user can zoom into/out of the arbitrary range by pinching in/out on the trackpad of the PC and check the results by playback. During playback, the system represents the beat by making a clicking sound when the playhead reaches a beat location. The user listens to it, and if the user finds an error or undesired result, the user can modify it. After certain modifications, the user presses the "Re-Opt" button (the blue button), and the system then starts the adaptation. The progress of the optimization process is displayed on the screen and updated at each iteration. The user can stop the process at an arbitrary timing.

To correct the errors by the user, we provide several options. These tools are categorized into three types. The first type is simply editing the beat directory (**Move/Insert/Remove/Lock/Tap**). The move tool allows the movement of the beat position by dragging the mouse. The insert and remove tools are used to add a new beat and remove an existing beat. The lock tool is used to fix the beat positions. By using a tap tool, the user can specify the beat directly during playback by taping a key on the PC keyboard.

The second tool type constrains the tempo trajectory, which affects the distribution of the beats throughout the

entire piece of music (§.5.4). There are two tools. First, the **Tempo Range Limit Setting Tool** allows the upper and lower limits of the tempo search range to be set by a curve editor. The user can add keyframes at arbitrary points and set the tempo range using Bezier curves. The system then excludes tempos that fall outside of this range from the search. Next, the **Tempo Change Flexibility Slider** determines the stability of the tempo tracking. The higher the value is, the more the system allows the tempo to vary widely throughout the music. For example, for rock and pop songs, where the tempo often remains almost constant, this value can be set to a low value, whereas for classical music, where the tempo often changes significantly, a high value can often achieve better results.

The last type is a special compared with the two above. This is the **Label Annotation Tool** that is used to provide the user's feedback to the system and improve the adaptation ability. This tool can be used only after a beat modification using the first tool type. When the user edits a beat, it recommends beats that have the possibility to be significantly affected by the user's modification by applying question marks, as below.

Figure 3. The label annotation tool.

In the above figure, the red beat is the beat currently edited the user. The user listens to some (not necessarily all) of the recommended locations (question marks) and judges whether they really need to be modified or not. If a correction is necessary, the user assigns a positive label to the location;, otherwise the label is negative. This can be achieved by clicking on the label mark with the mouse. Each time the user clicks on the same mark, the positive and negative labels are alternately switched. The user can also directly modify the recommended beat, but does not have to correct it and leave. This is an important aspect. The effort of this labeling is minimal, whereas the effort required to edit a user's beat manually is significant. This is because it only takes a moment to listen to the specified portion and judge whether it needs to be modified. In addition, this also has a merit in that we can fix the location of the negative labeled beat at this point. The system uses these assigned labels to compute the attention map in the ARSA (§.5.3).

4. BEAT TRACKING

Our beat-tracking algorithm basically follows the state-of-the-art algorithm [5]. First, we divide the target music into small frames and obtain the features for each frame. We use the time difference of the mel-frequency cepstrums (MFCCs) for the feature (one can also use alternative features such as CQT [26]). We used 44.1kHz for the sampling rate, $W = 4096$ samples for the Hanning window lengths, and $W/2$ for the hop size. We then input this feature series of T frames into the DNN and compute the beat activation function for each frame. As described in §.1, we train this DNN using only a local context of the music piece. We use three serial connected TCN blocks [5] (five layers with {1, 2, 4, 8, 16} dilations and the gated activations [29]), followed by three linear layers with the ReLU (Figure 4). We note that the figure shows only a single TCN block for simplicity. TCN is a variant of a dilated convolutional neural network that is inspired by WaveNet [29]. It takes the past and future frames around the focused frame. We use the frames for approximately 2.4 seconds for past and future frames. To train this DNN, we used four types of datasets: GTZAN [31–33], the RWC popular and genre dataset [34,35], and an in-house dataset of 400 musical pieces of various genres purchased from Amazon Music [36] and annotated by a vendor. Note that although our DNN architecture is slightly different from [5], we could not observe significant differences in our experiments.

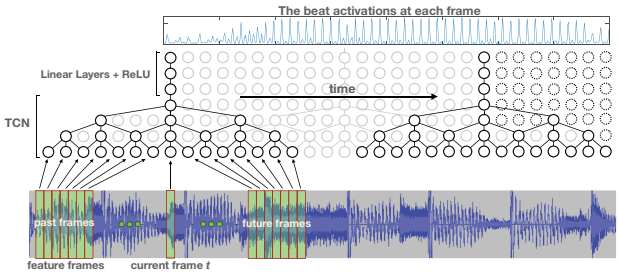

Figure 4. Our DNN is trained with a local manner.

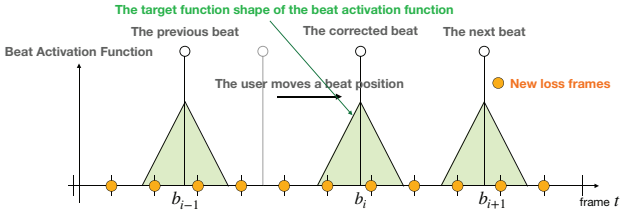

Figure 5. The target function shape of the beat activation.

To determine the actual beat positions from the beat activation function, we use the the hidden semi-Markov model (HSMM) [27, 28], which applies the activation as the observation state and the tempo as the hidden state. It estimates the most plausible trajectory throughout the entire piece of music by solving the Viterbi algorithm. Our method is basically similar to the state-space tempo model [8, 14] that uses the HMM. As a significant difference, our HSMM treats the midway phase of each tempo as the hidden state (Figure 6: Right). This HSMM divides the duration of a beat of each tempo into microphases, and gradually moves through the intermediate states sequentially until it reaches the next beat head. Only when the state reaches the beat head state, it is allowed to transit to a different tempo.

5. ADAPTATION ALGORITHM

5.1 Loss Making from the User's Corrections

We treat the user-modified beat and the two adjacent beats, for a total of three beats, as a unit. We describe the situation in which the user moves a beat position as an example here, but the basic idea is similar to other editing methods. Assuming that the user-modified beat position is b_i and the beat positions before and after it are b_{i-1}, b_{i+1}, we make the DNN output to be close to one at b_i, and zero at the midpoint $b_i^- = \frac{b_i+b_{i-1}}{2}$ and $b_i^+ = \frac{b_i+b_{i+1}}{2}$, (Figure 5).

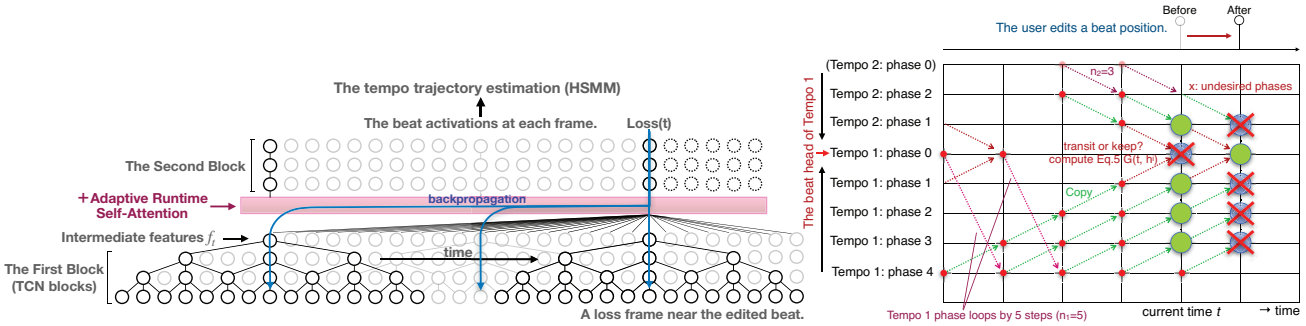

Figure 6. Left: At runtime, we split the pre-trained model into two blocks and insert the adaptive runtime self-attention block between them (§.5.2). Right: Hidden semi-Markov model for estimating the most plausible tempo trajectory (§.5.4).

Therefore, we define the loss function at an orange color frame t (referred to as the loss frame in the figure) as,

$$L(t) = \left| DNN(t) - \left(1 - min(1.0, \frac{|b^c - t|}{W})\right)\right|_2 \quad (1)$$

where b^c and W denote the closest beat from the frame t and the window size, respectively. We minimize the sum of the loss functions at all the loss frames by a gradient descent, such as Adam [37], to adapt the DNN.

5.2 Adaptive Runtime Self-Attention

As described, our DNN is trained using only a local context during pre-training, taking approximately 2.4 seconds of each past and future frame around the interest frame as input, and shifting the frames individual. By contrast, at runtime, we split the DNN into two parts, the first half (three blocks of TCNs) and the second half (three layers of linear layers and ReLU activations), and insert the ARSA between them (Figure 6:Left). This means that the model architecture itself has changed. The ARSA connects all of the intermediate features (the outputs of the first block) throughout the entire piece of the music and outputs the input for the second block at time frame t. This new type of DNN model achieves a better balance between the reduced computational cost during pre-training and an improvement of the runtime adaptation capability that distributes locally modified user feedback to throughout the entire piece of music.

Now supposing that the series of the outputs of the first block (intermediate features) $\{f^1, f^2, ..., f^T\}, f^t \in \mathbb{R}^D$, similar to general attention, the ARSA first computes three parameters at each frame t as: $Query^t = \mathbf{W}_Q \cdot f^t$, $Key^t = \mathbf{W}_K \cdot f^t$, $Value^t = \mathbf{W}_V \cdot f^t$, where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V \in \mathbb{R}^{D\times D}$ are the matrices that are initialized as identity matrices. General attention uses the inner products between queries and keys to construct the attention map. Alternatively, we use the weighted inner product of the $Query^t$ and Key^i as,

$$d(t,i) = (w^t \otimes Query^t) \cdot (P_e(|t-i|) \otimes Key^i), \quad (2)$$

where $w^t \in \mathbb{R}^D$ is the weight that is interactively determined by the user's label annotation (§.3). We describe this in §.5.3. $P_e(\delta t) \in \mathbb{R}^D, P_e(\delta t)_k = sin(\omega_n \delta t)$, if $k = 2n$; otherwise, $cos(\omega_n \delta t)$, where $\omega_n = 1.0/0.0001^{2n/\frac{D}{2}}$, denotes the relative positional encoding [13] that represents how apart the frame i is from the frame t. This $d(t,i)$ becomes the attention map at the frame t toward the frame i in the ARSA. After sorting this attention map $d(t,i)$ with respect to i, we then sample N frames in descending order in a manner of an importance sampling, and store them into $\mathbb{C}_t$ (we use N=512). Then, the output of the ARSA at frame t can be formulated as follows,

$$y_t = (1-\alpha) \cdot Value^t + \alpha \sum_{i\in\mathbb{C}_t} \frac{exp(d(t,i))}{\sum_{j\in\mathbb{C}_t} exp(d(t,j))} \cdot Value^i, \quad (3)$$

where α denotes the weight as a hyperparameter (we set 0.5 in our experiment).

5.3 Interactively Adapting the Attention

The weight w^t in Eq.2 is a unique parameter of ARSA that is computed using the user's interactive feedback. As described in §.3, when the user edits a beat, the label annotation tool first shows the beats that are expected to be significantly affected by the modification (the question marks in the Figure 3). These locations can be derived naturally from the original principles of the attention mechanism. The fact that the attention map $d(t,i)$ is larger means that the frame t is more focused on the frame i. In other words, the amount of back propagation from a given loss in frame t would be distributed more to frames with larger $d(t,i)$. Therefore, we can state that the beats within the vicinity of frames with large attention map values tend to be more susceptible to the user modification. Thus, the system shows these beats to correct the user feedback.

For the recommended beats, the user judges whether they really need to be corrects, and assigns positive/negative labels to them. The assigned label on a beat is redistributed to the neighboring frames, as shown in the following figure.

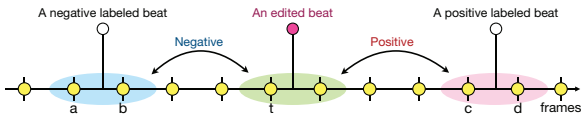

Figure 7. The labels are distributed to the neighbor frames.

These distributed labels are used to determine the relationship between each frame and the interest frame t. For example, in Figure 7, the frames $\{a, b\}$ are negative for t, and $\{c, d\}$ becomes positive for t. Using these relationships of the frames for the interest frame t, we then compute the Fisher's criterion [25, 38] as

$$w^t = \frac{(avg(A^{t,pos}) - avg(A^{t,neg}))^2}{std(A^{t,pos})^2 + std(A^{t,neg})^2}, \quad (4)$$

where $A^{t,pos}$ and $A^{t,neg}$ represent $Query^t \otimes P_e(|t-i|) \otimes Key^i$ in the positive and negative frames i for the interest frame t, respectively. Note that if no feedback from the user is obtained, we set $w^t = \mathbf{1}$. This weight reduces the effects from the user's modification on the frames that the user does not want to be changed, and conversely increases the effects simultaneously on the frames that the user wants to more aggressively modify. It is therefore expected to improve the adaptation speed.

5.4 HSMM with The User's Constraints

We also apply the user's modifications to the HSMM as constraints using two approaches during the forward path in the Viterbi algorithm. The first approach is to simply reduce the likelihoods of the undesired phases directly. For example, in Figure 6:Right, it is undesirable to pass through the phase states other than the beat head states at the destination frame where the user has moved the beat. Therefore, we prevent such undesired paths by reducing the likelihoods on the undesired grids (the blue circles).

The second approach is to weight the likelihoods at each beat-head state, where the tempo transition occurs. We apply a similar idea to that in §.5.1. First, we prepare a weighting array $U \in \mathbb{R}^T$ and initialize it by $\mathbf{1}$. We set $U_t = 1 + (1 - |t - b_i|/W)$ for the frames near the edited beat and its neighbors, and $U_t = 1 - (1 - |t - b_i^{\pm}|/W)$ for the frames around their midpoints. Then, our formulation of the integrated likelihood at the beat-head state h^i of the tempo i at the frame t becomes

$$G(t, h^i) = \max_j\{ G(t-1, e^j) + F(t, j, i) P_{t-n_j} U_{t-n_j} \Gamma_{j,i} P_t U_t \}, \quad (5)$$

where e^j denotes the phase state just before the beat head of the tempo j. $\Gamma_{j,i}$ and P_t are the transition probabilities from tempo j to i, and the observable state (the outputs of the DNN). n_j is the number of phase divisions of tempo j. $F(t, j, i)$ constrains the tempo range and flexibility that are set by the tempo-setting tools in §.3. If the tempo i exceeds the upper tempo limit or falls the lower tempo limit, $F(t, j, i) = 0$; otherwise, $F(t, j, i) = 1$. In addition, if $n_i/n_j > flex_t$ or $n_i/n_j < 1/flex_t$, $F(t, j, i) = 0$; otherwise, $F(t, j, i) = 1$, where $flex_t \in [0, 1]$, is the flexibility of the tempo change (the slider value).

6. EVALUATION

6.1 Validation through Simulation

We first conducted a validation using a simulation environment. We compared the results with and without ARSA. This comparative method without ARSA is equivalent to fine-tuning, which corresponds to Nakano et al. [18]. In this experiment, we used the SMC MIREX dataset [39], which includes 217 musical pieces of various genres with annotations of the beat positions. The lengths of all the pieces were aligned to a length of 40 seconds. We used a laptop PC (CPU: 2.9 GHz 6-Core Intel Core i9, RAM 32GB, GPU: Radeon Pro Vega 20) for equipment. We implemented all the our system including the DNN in C++.

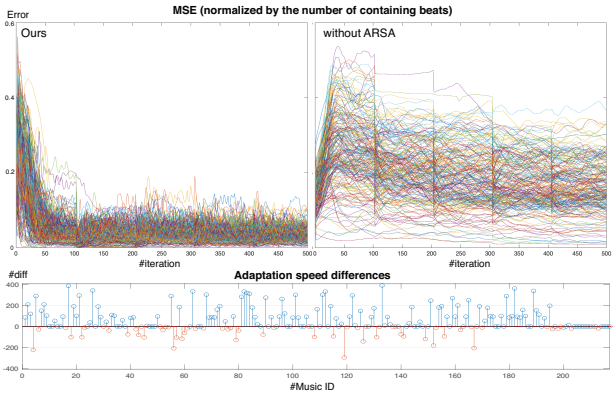

Figure 8. The result in a simulated experiment. The top row shows the transitions of the MSE losses throughout the entire musics. The bottom row shows a comparison of how quickly the adaptation converged in both methods.

The experimental procedure was as follows. We first found the three adjacent beats that have the highest errors (MSE loss), and automatically corrected them (treating three adjacent beats as a single unit, as described in §5.1). The system then iterated the adaptation iterations 100 times. The system repeated this procedure 5 times. We then obtained totally 5 error corrections and 500 adaptation iterations for each piece. For ARSA label annotations, we automatically assigned a positive label if the recommended beat position was sufficiently close to the correct beat annotation, and otherwise assigned a negative label. We added five labels for each error correction. Following to Dixon [40], we considered that an analyzed beat is accurate if it falls within a ±70 ms tolerance window around the correct position. For adaptation, we updated the parameters in the second block and the ARSA while keeping that of the first block in Figure 6:Left. We found this achieves a better balance between the computational cost and the adaptation ability in our experiments.

Figure 8 shows the result. The top row indicates the transitions of the mean squared errors (MSE loss) throughout the entire piece of music We note that these errors are normalized by the number of beats contained in each piece. Clearly, we can see that the errors in our method tend to converge to small values, and that our method has an ability to adapt throughout the entire song from only the local corrections. By contrast, with the comparative method, although the error itself tends to be reduced, but it does not converge well, and we can see that it struggles to adapt to the entire piece of music. The bottom row in figure 8 shows a comparison of how quickly the adaptation progresses with both methods. Suppose that the number of iterations when our method reaches an F1-measure of over 0.8 for the first time is N^A, and for the comparative method is N^B, the figure plots the difference, $N^B - N^A$. Then, a larger the positive value indicates that our method reaches the F1-measure faster than the existing method (blue), whereas a smaller negative value indicates that it achieves it at a slower rate (red). Statistically, in the 156/217 pieces ($p < 0.05$ by chi-squared test), our method reached to the goal significantly faster. For that computational cost, this simulation experiment (a total of

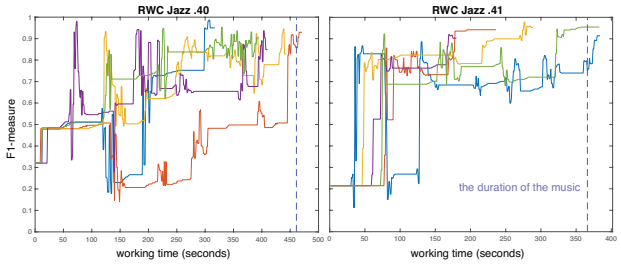

Figure 9. Improvements of the scores by 5 participants.

217 musical pieces) took 5.5 hours for our method and 5 hours for the comparative method. This approximately 8 seconds increases in cost per music on average is a negligible compared to the actual time the user requires for a manual correction in practical.

6.2 User Study

We conducted a user study to evaluate how the proposed interface improves the efficiency of the correction. We measured the working time for correcting the errors contained in the test target music. We set the baseline of this working time as the duration of the music. The reasons for this are as follows: Naturally, the fastest way to annotate the beats is to accurately tap throughout the entire piece of music if it is possible (although it is hard). The ideal working time for this method is equal to the duration. Therefore, our experiment determines whether our interface can reduce the working time to less than this baseline.

We hired 5 participants, all skilled users of sound editing software (e.g., Cubase [41] and Audacity [42]; DAW). The experiment consisted of three parts. The first part was to practice using our interface (45 minutes). We described how to use the interface, and each subject trained the system. The second part was a time trial (45 minutes). Each subject was asked to correct as quickly as possible the errors of the beats in the two pieces of target music using our interface. For the targets, we selected 2 pieces from the RWC Jazz dataset [34]. The first one is No.40, which has a duration of 7 minutes and 41seconds, and contains 481 beats, and the second one is No.41, which has a duration of 6 minutes and 6 seconds, and contains 637 beats. Both pieces were selected because they have appropriate durations and contain many errors from the initial analysis that are impractical to fix manually. Finally, the last part was a survey interview about their experience (10 minutes).

In the second part, each participant processed two pieces of target music. To avoid different interpretations of the beat by different people, we prepared an "answer" music file containing click sounds. We first asked the participants to listen to this answer entirely once through. Next, we asked the subjects to correct the beats as quickly as possible. We monitored the working times and progress of the F1-measure. When the F1 reached above 0.9 stably, we stopped the task even if they had not yet checked the entire piece of musics. Although each subject was allowed to use all tools prepared in §.3, however, we prohibited the use of only the tap tool for more than 5 seconds in a row. We also asked them to press the Re-Opt button as much as possible after each operation.

Figure 9 shows the result of improvements of the F1-measure (vertical axis) for each participants with the passage of the working time (horizontal axis). The vertical dotted lines represents the duration of each music. As the results show, 4/5 and 3/5 of the subjects completed the task in less time than the durations for pieces No. 40 and No. 41, respectively, and the remaining subjects also completed the task at slightly after the target duration. This means that most of the subjects completed the task without realizing it, despite not listening to the entire piece of music. Of course, when we really use our interface, we have to check the entire piece of music at the end. Therefore, the working time can not be less than the duration of the music piece. Therefore, albeit conditionally, we can conclude that our interface improves the efficiency of the error corrections.

Finally, we conducted an interview with each participant. We prepared two predetermined questions. The first question is *"Would you like to use this interface if it is implemented on DAW?"*. For this question, all the participants answered *"Yes"*. The second question is *"Did you have any problems in using the system?"*. For this question, some participants commented that **A.** they were confused about which tool to use for their facing beat error because the system provides many options for modification. Other participants also mentioned that **B.** they were unsure how soon the optimization loop should be stopped.

For the comment **A**, it is expected to be alleviated as the user becomes more proficient with the interface. However, as a better solution, the system could understand the current situation and recommend the most effective tool. This remains an area for a future work. For the comment **B**, because the concept of optimization through iterations is difficult for the common users to understand, we would need to find a criterion for the system to automatically terminate the iteration. However, because our adaptation process does not uniquely converge, it is difficult to find such a criterion. Therefore, this is a challenge for the future. As a similar problem, because of the iterative adaptation, the overall error rate frequently increases even if the user has corrected the error. This is not intuitive and is disconcerting for the user and should be improved.

7. CONCLUSION

In this paper, we proposed an interactive beat-tracking interface for adapting to a specific user and a targeted piece of music using a human-in-the-loop approach. To achieve this, we introduced a novel adaptive runtime self-attention that achieves a better balance between the lower computational cost during training and the high runtime adaptation ability that distributes the local modifications by the user to throughout the entire input sequence globally. We validated the feasibility and effectiveness of our method through several experiments, including a user study with the potential users of our interface. Beyond beat tracking, by training the machine learning model using only a local context and adapting it to a specific target using the global context at runtime, our method is expected to be useful for other broad domains such as chord recognition, singing voice synthesis, and sound source separation.

8. REFERENCES

[1] M. Goto and Y. Muraoka, "A beat tracking system for acoustic signals of music," *ACM Multimedia (ACMMM)*, 1994.

[2] Y. Ueda, Y. Uchiyama, T. Nishimoto, N. Ono, and S. Sagayama, "Hmm-based approach for automatic chord detection using refined acoustic features," *International Society for Music Information Retrieval (ISMIR)*, 2010.

[3] E. J. Humphrey and J. P. Bello, "Four timely insight on automatic chord estimation," *International Society for Music Information Retrieval (ISMIR)*, 2015.

[4] G. Shibata, R. Nishikimi, and K. Yoshi, "Music structure analysis based on an lstm hsmm hybrid model," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[5] M. E. P. Davies and S. Bock, "Temporal convolutional networks for musical audio beat tracking," *European Signal Processing Conference (EUSIPCO)*, 2019.

[6] S. Bock and M. Schedl, "Enhanced beat tracking with context-aware neural networks," *International Conference on Digital Audio Effects (DAFx)*, 2011.

[7] S. Bock, M. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," *International Society for Music Information Retrieval (ISMIR)*, 2016.

[8] S. Bock, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles," *International Society for Music Information Retrieval (ISMIR)*, 2014.

[9] S. Bock and M. E. P. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," *International Society for Music Information Retrieval (ISMIR)*, 2020.

[10] B. D. Giorgi, M. Mauch, and M. Levy, "Downbeat tracking with tempo-invariant convolutional neural networks," *International Society for Music Information Retrieval (ISMIR)*, 2020.

[11] S. Reich, "Clapping music," 1972.

[12] ——, "Piano phase," 1967.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[14] F. Krebs, S. Bock, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking," *International Society for Music Information Retrieval (ISMIR)*, 2015.

[15] C. Cannam, C. Landone, and M. Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," *ACM Multimedia (ACMMM)*, 1910.

[16] S. Dixon, "An interactive beat tracking and visualisation system," *International Computer Music Conference (ICMC)*, 2001.

[17] J. Driedger and M. M. Hendrik Schreiberand W. Haas, "Towards automatically correcting tapped beat annotations for music recordings," *International Society for Music Information Retrieval (ISMIR)*, 2019.

[18] T. Nakano, Y. Koyama, M. Hamasaki, and M. Goto, "Interactive deep singing-voice separation based on human-in-the-loop adaptation," *Intelligent User Interface (IUI)*, 2020.

[19] M. Goto, K. Yoshii, H. Fujihara, M. Mauch, and T. Nakano, "Songle: A web service for active music listening improved by user contributions," *International Society for Music Information Retrieval (ISMIR)*, 2011.

[20] K. Fujii, Y. Saito, S. Takamichi, Y. Baba, and H. Saruwatari, "Humangan: Generative adversarial network with human-based discriminator and its evaluation in speech perception modeling," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[21] N. J. Bryan and G. J. Mysore, "Interactive refinement of supervised and semi-supervised sound source separation estimates," *IEEE Acoustics, Speech, and Signal Processing*, 2013.

[22] T. Bazin, G. Hadjeres, P. Esling, and M. Malt, "Spectrogram inpainting for interactive generation of instrument sounds," *Joint Conference on AI Music Creativity*, 2020.

[23] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *Advances in Neural Information Processing Systems (NeurIPS)*, no. 32, 2019.

[24] Y. Zhou, Y. Koyama, M. Goto, and T. Igarashi, "Interactive exploration-exploitation balancing for generative melody," *Intelligent User Interface (IUI)*, 2021.

[25] B. Kim and B. Pardo, "A human-in-the-loop system for sound event detection and annotation," *Intelligent User Interface (IUI)*, 2018.

[26] J. C. Brown, "Calculation of a constant q spectral transform," *Journal of Acoustics in America*, 1991.

[27] R. Chen, W. Shen, A. Srinivasamurthy, and P. Chordia, "Chord recognition using duration-explicit hidden markov models," *International Society for Music Information Retrieval (ISMIR)*, 2012.

[28] K. Shibata, R. Nishikimi, S. Fukayama, M. Goto, E. Nakamura, K. Itoyama, and K. Yoshii, "Joint transcription of lead, base, and rhythm guters based on a factorial hidden semi-markov model," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[29] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "A. van den oord and s. dieleman and h. zen and k. simonyan and o. vinyals and a. graves and n. kalchbrenner and a. senior and k. kavukcuogluwavenet: A generative model for raw audio," *CoRR abs/1609.03499*, 2016.

[30] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *International. Conference on Learning Representations*, 2016.

[31] G. Tzanetakis and P. Coo, "Musical genre classification of audio signals," *"IEEE Transactions on Audio and Speech Processing*, 2002.

[32] U. Marchand, Q. Fresnel, and G. Peeters, "Gtzan-rhythm: Extending the gtzan test-set with beat, downbeat and swing annotations," *International Society for Music Information Retrieval (ISMIR)*, 2015.

[33] U. Marchand and G. Peeter, "Swing ratio estimation," *International Conference on Digital Audio Effects (DAFx)*, 2015.

[34] M. Goto, "Masataka goto and hiroki hashiguchi and takuichi nishimura and ryuichi oka," *International Society for Music Information Retrieval (ISMIR)*, 2002.

[35] ——, "Aist annotation for the rwc music database," *International Society for Music Information Retrieval (ISMIR)*, 2006.

[36] Amazon, "Amazon music," *https://music.amazon.co.jp*.

[37] J. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," *CoRR abs/1412.6980*, 2014.

[38] E. Wu and A. Zhang, "A feature re-weighting approach for relevance feedback in image retrieval," *International Conference on Image Processing*, 2002.

[39] A. Holzapfel, M. E. P. Davies, J. R. Zapata, and J. L. Oliveira, "Selective sampling for beat tracking evaluation," *IEEE Transactions on Audio, Speech, and Language Processing*, 2012.

[40] S. Dixon, "Evaluation of the audiobbeat ttacking system beatroot," *New Music Res*, 2007.

[41] Steinberg, "Cubase," *https://new.steinberg.net/cubase/*.

[42] Audacity, "Audacity," *https://www.audacityteam.org*.