Assignment 1: Beat and Downbeat Tracking

Matthew Rice
Queen Mary University of London
m.rice@se22.qmul.ac.uk

1. Introduction

In this assignment, we were tasked with implementing a beat-tracking system for ballroom dance music. Beat tracking is defined as ”determining the time instances in an audio recording, where a human listener is likely to tap his/her foot to the music” [8] In particular, we are looking for the tactus (primary metrical level) of the music. A related task is downbeat tracking, where the goal is to determine the beginning of each musical bar known as the downbeat. In fact, the downbeat times should be a subset of the beat times.

For most of the history of this task, music informatics approaches have been used with some success. Recently, the best results for this task use deep learning techniques. Therefore, I chose to implement a variation of the beat-tracking system outlined in [6], which is explained in short in sections 2 and 3. Unlike in their previous work [2], this approach did not consider downbeat times. Therefore, besides implementing the paper from scratch, I also tried some approaches to obtain the downbeat times. [1]

2. Theory

The before-mentioned paper uses a modified Temporal Convolutional Network (TCN) combined with a dynamic Bayesian network (DBN) to output beat times, but not downbeat times. Before passing through the network, the audio waveform data is transformed into a log spectrogram, then passed through a convolutional front end.

2.1. TCN for Beat Tracking

The foundational paper WaveNet [7] introduced the idea of a TCN, a modification to CNN networks that allows for efficient and powerful training on large sample sizes such as audio. The main idea is to use layered dilation for the convolutional kernels, which enable the CNN layers to have larger receptive fields without increasing the number of layers. This paper uses the TCN as a core component of the system, but makes two key modifications, (1) replace tanh-style activations with ELU activations and (2) replace causal convolutions with non-causal convolutions. The reasoning was the non-causal convolutions would allow the TCN layers to see both forwards and backwards in time, instead of solely backwards in time. This is due to the beat-tracking task being an ”offline” task; for a real-time task, causal convolutions would be needed. Besides the TCN, the approach includes a convolutional front-end to abstract an intermediate representation of the spectrogram, and a DBN to decode the beat energies into beat times.

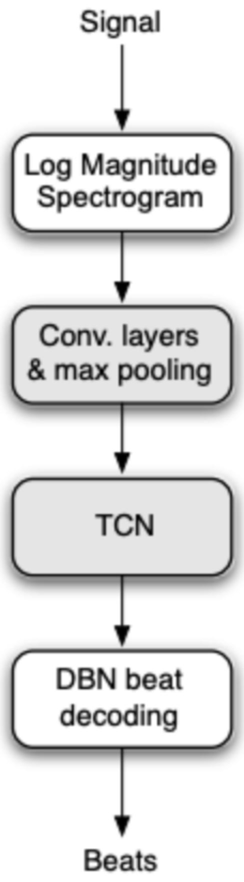

Figure 1: Inference-time forward pass of the system presented in [6]

[1]Code and checkpoints can be found at https://github.com/mhrice/BeatTrack

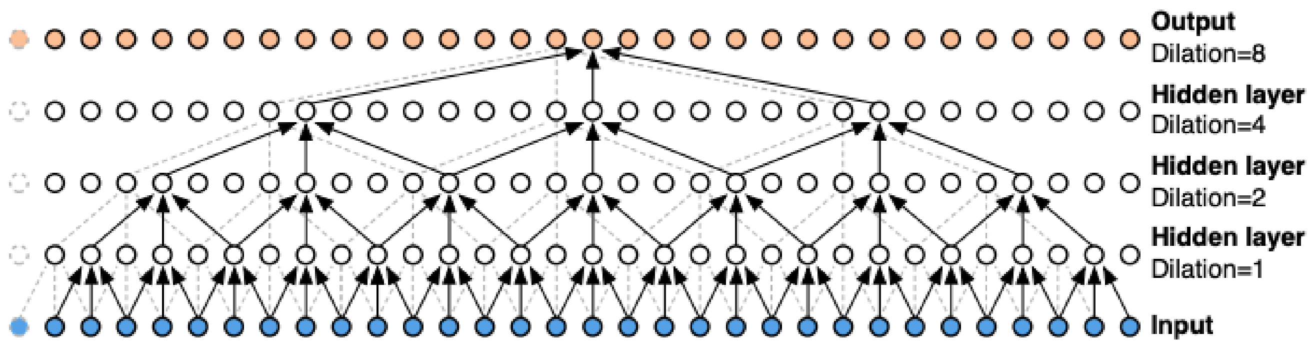

Figure 2: Diagram of non-causal TCN Layer presented in [6]

2.2. DBN Beat decoding

The DBN is similar to a hidden Markov model but is specialized in modeling time series data. [5] In this network, the purpose of the DBN is to perform thresholding and peak-picking from the output of the TCN, which allow the system to be a bit more robust to small variations in predictions. [3]

2.3. Downbeat Tracking

Although this paper did not implement downbeat tracking, a future work [1] by the same authors suggested some ideas on how to obtain downbeats with a similar system. There are two discussed methods: modify the architecture to model the beats and downbeats jointly (*joint*) or output the downbeats as a post-processing step after obtaining the beats (*sequential*). I attempted both approaches, although implemented them differently than what was suggested.

3. Implementation

The entire system was written in python using the pytorch-lightning deep learning framework. Pytorch-lightning is built on top of pytorch that speeds up the implementation and training of deep learning techniques.

3.1. TCN system

The convolutional front-end and TCN were both implemented as standard pytorch nn classes. While the convolutional front-end is straightforward to create, the TCN required some more time but was aided by watching a TCN implementation video [4]. The output of the TCN has a sigmoid unit to ensure the values are between 0 and 1. For the DBN, the author's implementation in the madmom [2] library is used. All the settings used follow [6].

[2]`https://github.com/CPJKU/madmom`

3.2. Downbeat modifications

Two approaches were used to obtain downbeats. For the *joint* approach, the number of filters was increased to 20 as suggested by [1], then a linear layer was added before the final sigmoid, which mapped the 1D output to a 2D output as suggested by [9] For the *sequential* approach, an attempt was made to leverage the knowledge that the data had either 3 or 4 beats-per-measure. Several approaches were tried, but the best-performing one was as follows:

1. Find the beat corresponding to the highest low-frequency energy by comparing the sum the lowest 20 bins of the mel spectrogram at the beat locations
2. Assume this is a downbeat, then calculate the initial beat positions for both 3 and 4 beats-per-measure
3. Sum all of the energies of the beat locations starting from the initial beat position, skipping 3 for the 3 beat-per-measure calculation and 4 for the 4 beat-per-measure calculation.
4. Select the method with the highest energy as the correct beat-per-measure, and return the associated beat times.

3.3. Dataset

The dataset used is the ballroom dataset, which includes ballroom dances in various styles as well as beat time and downbeat number labels. The dataset is split into 80% train, 10% valid, and 10% test. Each audio file is resampled, trimmed or padded to 30 seconds, then passed through a mel spectrogram with the same settings as 3. The shape of the output is (frames, mels) = (3001, 81). The beat labels are a 1D vector with then the same length as number of spectrogram frames. They are processed such that there is a 1 in every frame with a beat, as well as 0.5 in the two adjacent frames to either side. The downbeat labels are processed in the same way as the beat labels. The spectrogram and

Method	Beats						Downbeats					
	FM	CML_c	CML_t	AML_c	AML_t	D	FM	CML_c	CML_t	AML_c	AML_t	D
Davies [6]	0.933	0.864	0.881	0.909	0.929	3.456						
Beats-Only	0.971	0.944	0.844	0.972	0.982	0.689						
Joint	0.944	0.854	0.856	0.985	0.986	0.697	0.556	0.032	0.032	0.631	0.631	0.643
Sequential	0.983	0.972	0.974	0.986	0.972	0.689	0.281	0.069	0.069	0.154	0.177	0.582

Table 1: Performance of beat and downbeat tracking systems

associated beat and downbeat labels are then saved to disk. This preprocessing step is done before training to speed up training on a GPU.

3.4. Training

During training, each batch of the spectrograms is passed through the network, excluding the DBN. For the *joint* case, the output is split into beat predictions and downbeat predictions, following 8. The loss is then calculated as:

$$L_{beats} = BCE(beat\,preds, beat\,labels) \tag{1}$$

$$L_{downbeats} = BCE(db\,preds, db\,labels) \tag{2}$$

$$L = L_{beats} + L_{downbeats} \tag{3}$$

with BCE being binary cross entropy. For the *sequential* case, only the L_{beats} loss is used, and the output is not split. The network is optimized with the Adam optimizer and trained until the validation loss is constant for 75 epochs. Also, near the end of training, the learning rate is reduced by a factor of 10 to improve convergence.

Unlike the original paper, I use a batch size of 32 instead of 1 because I pad/trim all the audio files to 30 seconds. Trained occurred on a single NVIDIA A100 on the QMUL JupyterHub cloud server.

3.5. Inference

During inference, the model first loads the best checkpoint. Then the data is preprocessed in the same manor as the for training. Afterwards, the model takes a forward pass on the data and outputs beat predictions and downbeat predictions. Then a beat DBN and downbeat DBN post-process the raw activations into beat times and downbeat times respectively. Both beat and downbeat DBNs use a transition λ of 100 and 100 fps, but the beat DBN is set to look for beats in a tempo range between 55 and 215 bpm while the downbeat DBN looks for beats in a tempo range between 15 and 80 bpm. Note that for the *sequential* case, only the beat times are output and processed, the downbeats are extracted using a implementation of section 3.2.

4. Results

Following 6, $F\text{-}Measure$ (FM), CML_c, CML_t, AML_c, AML_t, and $InformationGain$ (D) are calculated for both beats and downbeats. These metrics are calculated using the *mir_eval* library and aggregated over one test epoch. I present the results of the baseline from the paper (Davies) along with the beats-only model, joint beat and downbeat tracking models, and the sequential (downbeat post-processing) model. See Table 1 for a listing of all of the results.

5. Discussion

Although the beat tracking results show impressive results even compared to the already remarkable baseline, it is worth noting that there is a difference in the training procedure. Namely, the baseline system was trained on 4 vastly different datasets, while the presented approach only uses the ballroom dataset. There is also some strangeness in the calculation of the information gain metric (D). The large difference between the baseline and the calculated performances suggests that there is some difference in the implementation of this metric than the one from mir_eval. Furthermore, although theoretically the Beats-Only and Sequential should have identical results, there are some minor differences which are likely due to a few hyper-parameters being changed during training.

The downbeat tracking performance has much more room for growth. Although the presented approaches were attempted, they seem significantly worse than some other approaches in newer papers. It is likely due to both the different implementations of joint beat and downbeat tracking as well as the dataset and training schemes. Also, since downbeat tracking was never evaluated by itself, there is a possibility of a bug in the implementation.

Also, as expected, not only were the results impressive, but using the TCN was an efficient method to train, as the models would train in less than 30 minutes, which allowed for substantial experimentation.

6. Conclusion

In this assignment, I successfully implemented a modern deep learning approach to beat tracking from scratch that was able to perform similarly to the reported performances metrics. This method focused on using a TCN as the main network to process the spectrograms. Although the original approach did not produce downbeat times, I reviewed a few approaches with some degree of success.

References

[1] S. Böck and M. E. P. Davies. Deconstruct, Analyze, Reconstruct: How To Improve Tempo, Beat, And Downbeat Estimation.

[2] S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks.

[3] S. Böck, F. Krebs, and G. Widmer. A multi-model approach to beat tracking considering heterogeneous music styles.

[4] DLVU. Lecture 5.4 - CNNs for sequential data.

[5] Z. Ghahramani. An Introduction To Hidden Markov Models And Bayesian Networks.

[6] E. P. MatthewDavies and S. Bock. Temporal convolutional networks for musical audio beat tracking. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE.

[7] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio.

[8] J. Shaikh. Beat tracking | music information retrieval python.

[9] J. Zhao, G. Xia, and Y. Wang. Beat transformer: Demixed beat and downbeat tracking with dilated self-attention.