Reproduction of Temporal Convolutional Networks for Musical Audio Beat Tracking

Jiajun Wu
University of Rochester
jwu107@u.rochester.edu

Sebastian Xu
University of Rochester
sxu38@ece.rochester.edu

ABSTRACT

The purpose of this reproduction is to check the effectiveness of Temporal Convolutional Networks (TCN) models on beat tracking tasks. The paper reproduced is Temporal Convolutional Networks for Musical Audio Beat Tracking. Through the training process, ease of training for this TCN model compared to RNN based model is confirmed. In the reproduction process, an environment was set up, training data and targets were downloaded from github, pre-processing was successfully carried out, training was nicely implemented, but the model evaluation process was not successful.

1. INTRODUCTION

The method for beat tracking reached the state of art (SOTA) a long time ago. However, beat tracking lies as the basis for Music information retrieval (MIR) tasks. Beat tracking can provide structural insight for models designed for beat related tasks. For example, chord detection might use beat detection because the existence, or onset, of the chord's note can provide useful information for recognising chords. For another example, beat detection will be very useful when it comes to drum beat generation for beatless music audio. The goal of this reproduction is to test if TCN model accuracy is as good as stated in the paper being reproduced.

2. ORIGINAL PAPER SUMMARY

In the paper, the SOTA method by using Bidirectional Long Short-Term Memory is described to compare with the method proposed by the paper. Take one audio date in .wav format as input data, the audio input is pre-processed and converted to mel-spectrogram in numpy format. Three different mel-spectrograms are generated, with hop size of 10ms, window sizes of 23.2 ms, 46.4 ms and 92.9 ms. Three mel-spectrograms are fed to Three layers of BLSTM and the BLSTM generates a beat activation function. Eventually the beat activation function is sent to DBN approximated via hidden Markov Model (HMM). The DBN used at the end is to better track the position of the beat.

Fig. 1 contains the main processing pipeline for both BLSTM model and model proposed by the paper.

The method proposed by the paper is similar to that of the BLSTM method. For pro-processing, one audio in .wav format is converted to mel-spectrogram. However, this time, only one mel-spectrogram, with hop size of 10

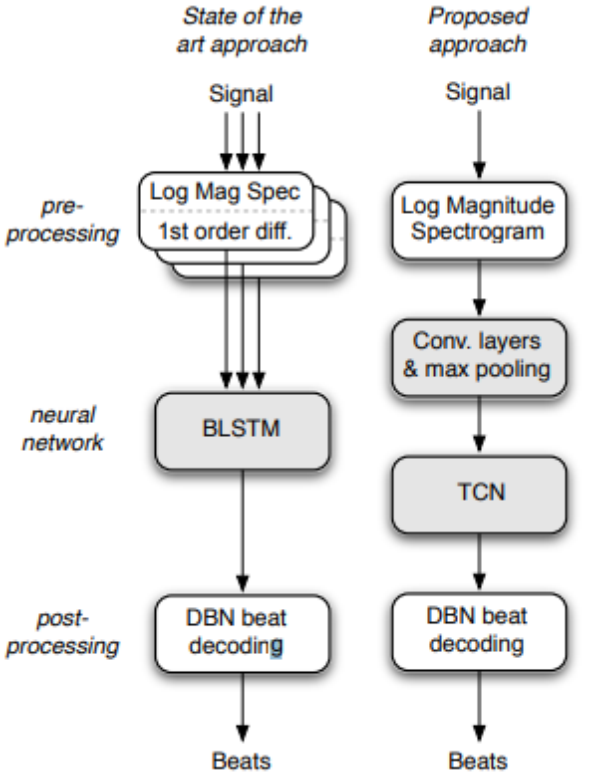

Fig. 1. Comparison between existing state of the art (left) with our proposed approach (right). The neural network blocks are shaded light grey, from [1].

ms, window size of 46.6 ms, is generated. The mel-spectrogram is then fed to the Convolutional Block. The structure of the Convolution Block is as follows: the first two layers have 16 filters of size 3x3 with max pooling over 3 bins in the frequency dimension; the third layer has 16 filters of size 1x8 but this time without pooling. The Exponential Linear Unit (ELU) is used as the activation function in the Convolutional Block. ELU used here is smoother than ReLU. Dropout rate is set to 0.1. Then comes the TCN Block. The TCN has only 1 stack, with dilations from 2^0 to 2^10, 16 filters, filter size of 5, 0.1 dropout rate and ELU activation function. Eventually, DBN is used to decode the beat activation function and yields the final beat position result.

For the implementation part proposed by the paper: as for training: Adam optimizer is used; learning rate is set to 0.001, batch size is 1, sigmoid activation

function is used as the output function; binary cross-entropy is used for loss function.

Table 1 contains all the signal processing and learning parameters.

Signal Conditioning	
Audio sample rate	44.1 kHz
Window shape	*Hann*
Window & FFT size	2048 *samples*
Hop size	10 ms
Filterbank freq. range	30 . . . 17000 Hz
Sub-bands per octave	12
Total number of bands	81
Conv. Block	
Number of filters	16, 16, 16
Filter size	3×3, 3×3, 1×8
Max. pooling size	1×3, 1×3, —
Dropout rate	0.1
Activation function	*ELU*
TCN	
Number of stacks	1
Dilations	$2^{0,\ldots,10}$
Number of filters	16
Filter size	5
Spatial dropout rate	0.1
Activation function	*ELU*
Training	
Optimizer	*Adam*
Learning rate	0.001
Batch size	1
Output activation function	*sigmoid*
Loss function	*binary cross-entropy*

TABLE I. OVERVIEW OF SIGNAL PROCESSING AND LEARNING PARAMETERS, [1]

3. METHODOLOGY

Python3.12.4 is set up for the environment. Virtual environment is supposed to be created for the reproduction, default environment is used instead. Librosa, madmom, mir-eval, numpy, and torch are all installed into the environment. Installed Numpy version is 1.26.4. The versions of Numpy and python are specified because these two versions eventually cause the failure to produce model evaluation.

Source codes are available from github: https://github.com/ben-hayes/beat-tracking-tcn. All source codes are cloned into local directories.

Here is the reproduction implementation.

Dataset: Ballroom dataset is used as the training data. Ballroom dataset, now, contains 698 .wav files. Ballroom dataset is available for downloading on the UPF website. However, the dataset provided contains only the audio data, which is the training data, and no labels are contained. Tempo annotations are available for download but tempo annotations should not be used in the paper-proposed method. .beats file should be the label, the real annotation file, used for ground truth. Eventually, .beats annotation files are downloaded from github: https://github.com/CPJKU/BallroomAnnotations.

pre-processing: all 698 audio files are converted to mel-spectrogram by using librosa module. After conversion, all mel-spectrograms, in format of .npy, as training data, are stored in datasets/ballroom folder.

When the data is ready for training, run the train.py file in VS Code terminal, with the training dataset directory and label directory as input to the train.py file. GPU, 3060 is used for training. However, GPU is used not by specifying train.py's input but by directly modifying code in the train.py file. During training, the task manager is open and the usage of the GPU is confirmed by looking at the GPU usage. The training process stops when the validation error does not update for 50 epochs. The training process is terminated at epoch 99. The model is then stored in a checkpoint file.

Model Evaluation: the model evaluation is not successfully carried out because version incompatibility. The evaluate_model file is the one used for evaluation which uses the madmom module. However the madmom module requires a lower version of numpy. A method to change all np.int and np.float into int and float is used but it did not work out. Downgrade of numpy to version 1.19.5 is implemented but then comes the problem that the lower version of numpy is not compatible with python3.12.4. Downgrade of python to version python 3.8.16 is implemented, but because there is no installer for python 3.8.16, the model evaluation part is counted as failure.

4. RESULTS AND CONCLUSION

As a result, the data downloading, data loading, data pre-processing, training process are all successfully carried out. However, the model evaluation part is not successfully carried out. In conclusion, the reproduction is half successful.

5. FUTURE WORK

Python3.8 was tried to download from the official website, but it is easily accessed in anaconda. make the final model evaluation process successful, python3.8 will be downloaded from anaconda and the downgrad will be implemented. In addition, instead of using a local environment, a virtual environment will be added to the project file and activated during training. Hopefully model evaluation will be successfully implemented after successfully downgrading python and numpy.

Furthermore, 8 fold cross validation will be used for the training data to improve training.

6. REFERENCES

[1] Böck, S., & Davies, M. E. P. (2018). "Temporal Convolutional Networks for Musical Audio Beat Tracking," In *Proceedings of the 29th International Society for Music Information Retrieval Conference (ISMIR)* (pp. 90–97).