Real Time Beat Tracking: A Mixed Approach
Category: Music

George Woskob
gwoskob@stanford.edu

March 18, 2020

Abstract

Beats are the fundamental unit of time in music. Finding the beats in music is usually quite easy for many people; many people will actually inadvertently do it as they tap their feet to music. Beat tracking is the term used to describe the task when given to a computer. In this paper I describe a beat tracking system built that works in real time to find beats as they occur in an musical audio signal and predict the location of the next beat. The incoming audio gets converted to a spectrogram image where frequency and amplitude are plotted over a time domain. At a regular frequency, the image representing the last 3 seconds of audio gets passed to a convolutional neural net which predicts the location of the beats within the image. Given the predicted locations of the beat in the image a tempo is inferred as well as the temporal distance since the last beat. This information is passed to a Kalman filter which keeps track of the current belief about tempo and temporal distance from the last beat and updates its belief based on the new predictions.

1 Introduction

Music is periodic (rhythmic) and its speed is measured by beats, a unit of time in music around which musical events (such as notes played by instruments) are organized. Understanding beats is crucial to understanding music. Beats provide a framework for describing relative locations within a piece of music. On top of this temporal framework we can then understand higher level features of music such as harmony and form. The aim of this project is to build a beat tracking system. This system will identify the location within a beat and the tempo (the rate at which beats occur, typically measured in beats per minute) given a stream of musical audio. Being able to determine the location within this rhythmic structure in real time is useful to any sort of system that can accompany or assist live musicians. The challenges of a system like this are that digital audio is a dense format and that there is an enormous variety of musical styles. The input to this system is a stream of audio sampled at rate of 22,050 hz. Each sample represents the relative air density that a speaker would output if the audio were to be played over a speaker. The system described in this paper will output, at any particular moment, its estimate of the tempo of the music being observed, in beats per minute, and the location within the beat, where 0 would be the beginning of a beat, and as the output approaches 1 it approaches the end of a beat and thus the beginning of the next beat.

2 Related Work

The high performing beat tracking systems I came across in my research mainly fell into two categories, those that relied mostly upon signal processing [1][2] and those that relied mostly upon CNNs and RNNs [3][4]. Online (real time) and offline systems fell into both categories. The authors in [2] take a novel approach without any neural network. After applying a short time Fourier transform (STFT) to the audio, they use onset detection to find the beginnings of musical events (notes, chords, percussive moments) and then apply another Fourier transformation on these events to find the local periodicity of the musical events from which tempo and beat locations are inferred. Ideas from this paper could be applied to improving the current design's approach to determining the tempo and location within the beat given the output of the neural

model. All systems listed above transform the audio signal into a representation of frequency and amplitude over the time.

3 Data and Features

For this project I am using the well cited GTZAN dataset[5] with the annotations by Ugo Marchand et al in GTZAN-Rhythm[6] for the task. The GTZAN dataset offers 1,000 30 second audio recordings across a wide varieties of musical genres at a sample rate of 22050 hz and the annotations from GTZAN-Rhythm offer labels for the locations of beats as a list of locations in seconds per audio file in the original GTZAN dataset. All audio recordings are single channel (mono).

The audio files were transformed via STFT by the librosa audio processing library[7] so that every 512 audio samples (at 22,050 hz this represents roughly 1/43 of a second of audio) a vector representing the frequencies present in those samples was generated. The choice to calculate the STFT every 512 samples was due to the fact that larger frequencies are harder to detect with smaller windows and that 1/43 of a second represents an amount of time small enough that rhythmic clarity could still be maintained. Larger windows may produce more accurate representations of the frequencies present in a window but rhythmical accuracy would be diminished in the system. Concatenating 129 windows into a single image provides a representation of 3 seconds worth of audio, which was the input into the neural network. The labels for each image were the locations on the x-axis of each beat.

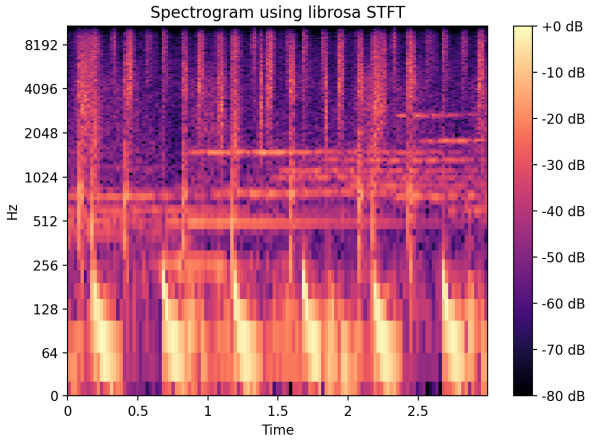

Figure 1: Above is an example of the input to the neural net. The y-axis is the frequency present in the audio at that moment int time which is graphed on a log scale to match our perception of pitch. Brighter colors denote higher amplitudes. Each vertical slice in the image represents a chunk of 512 samples or roughly 1/43 of a second with a total of 3 seconds of audio. The audio represented here is house music, a style of dance music which contains a regular pulsing low end and pronounced rhythmic features.

The entire dataset is comprised of 1,000 30 second audio files. This amounts to about 1,200 labeled images per file. Five percent of the audio files were randomly selected to be a part of the validation set. This amounted to roughly 1,140,000 labeled images in the training set and 60,000 labeled images in the validation set.

4 Methods

In order to determine the location within the beat and the tempo at a given point in time in a clip of musical audio, initial strategies had the model trying to predict these two things. Given an image like the one above, the model was to predict the beats per minute of the audio image and the relative location within a beat at the rightmost edge of the image. However, it was found that neural networks are not reliable linear

predictors, at leat not with the accuracy needed for this application. Thus, attention was turned to trying to make the neural net predict the locations of the beat across the x-axis, which represents time in the image. Inspiration was drawn from the YOLO algorithm[8] for the output of the nueral network in how it locates the beats within the image and for the loss function that optimized that neural network.

Similar to the grid that YOLO uses to search for objects, the audio image that gets fed into the model was divided into 15 vertical bins. The choice of 15 was due to the fact that at a tempo of 300 beats per minute, there would be 5 beats per second, which with three seconds of audio per image would mean 15 beats per image. 300 beats per minute is just above above the range of realistic tempos which nearly guarantees that a bin will not contain more than one beat. For each bin the model outputs two values, a probability estimate of a beat existing in that bin and an estimate of that beat's relative location within that bin, from left to right. The loss function is shown below:

$$\lambda_{loc} \sum_{i=0}^{B} 1_i^{beat} \left| x_i - \hat{x}_i \right| \tag{1}$$

$$+ \lambda_{beat} \sum_{i=0}^{B} 1_i^{beat} \left| C_i - \hat{C}_i \right| \tag{2}$$

$$+ \lambda_{nobeat} \sum_{i=0}^{B} 1_i^{nobeat} \left| C_i - \hat{C}_i \right| \tag{3}$$

where 1_i^{beat} denotes that there was a beat in that bin. During training λ_{loc} was set to 1.0 and λ_{beat} and λ_{nobeat} were set to 0.5. C_i denotes the confidence that there is a beat in that bin and B is the number of bins. The loss function penalizes the model if it does not properly identify a beat as belonging in the bin. It also penalizes for predicting the beat far from where it actually is. The hyperparameters of this loss function were not extensively searched, though setting λ_{beat} equal to λ_{nobeat} was found to work better than many other values. Additionally, squaring the errors of this loss function may be applied in future work to see if it can improve the model or allow it to learn more quickly.

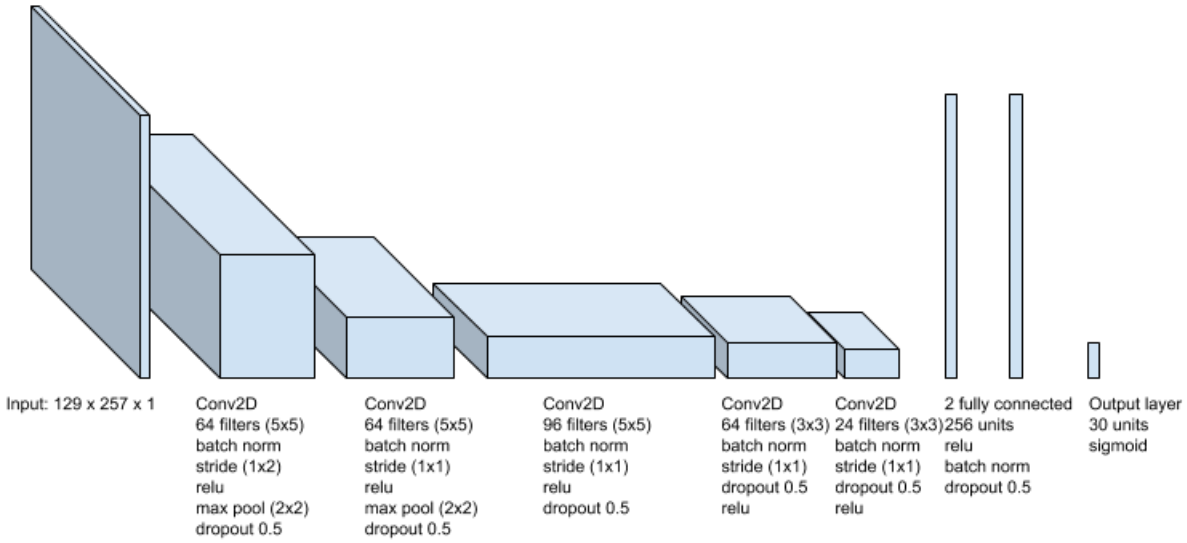

Figure 2: The neural network has 5 convolutional layers followed by two fully connected layers and a fully connected output layer. The first two convolutional layers have max pooling layers following them.

One dimensional convolution was tried before two dimensional convolution. This allowed for fast feedback on outputs to the model that were going to prove useful or not because it was much faster to train a 1D model. Eventually it was found that 2D convolution was predicting beats and tempos more accurately than

1D, though 1D was not a terrible estimator. The neural network was built and trained using the Keras API for tensorflow[9][10] for 36 epochs over the traning set..

The output of the neural net predicts the locations of the beats in the last 3 seconds of audio. With this information, tempo can be inferred by averaging the spans between beats. For instance, if two beats are identified, and the time between them is 0.5 seconds, we could infer that the tempo is 120 beats per minute because a beat of length 0.5 seconds means 2 beats per second and over 60 seconds this is 120. Additionally we can infer our location relative to the last beat by taking the inferred length of a beat and multiplying it by the length of time since last beat.

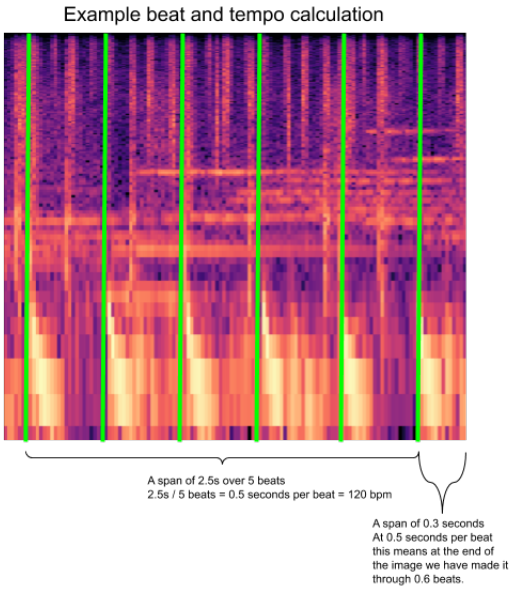

Figure 3: Given an audio image, the model will predict the locations of the beats. These are shown as green vertical lines superimposed on the same audio image from Figure 1. An example of how tempo and the location within the beat at the end of the image are calculated given the model's output is shown above.

Once the tempo and location within the beat are found, this is passed to a Kalman filter as previously implemented in [11]. Based on all the observations passed to it thus far the Kalman filter has its own internal representation of the state of the tempo and location within the beat. Each additional observation allows the Kalman filter to update its state. Using the Kalman filter's internal representation of the state, it can also predict future states even if devoid of new observations. Observations are passed to the Kalman filter roughly eight times a second, allowing the neural network enough time to make predictions between updates to the Kalman filter.

The Kalman filter works by essentially averaging linear distributions. Given that both tempo and location with the beat can be represented a linear distributions, the Kalman filter is an effective tool for smoothing out the noisy predictions that the neural network makes by reconciling these predictions against its own idea of the the state of the world.

5 Results

There is an oft cited paper [12] that defines a number of methods for evaluating beat tracking algorithms and a Python library mir_eval[13] that implements the methods. A common metric is F-measure which takes into account accuracy, precision, and recall with correct identification meaning that the identified start of the beat occurred within a 0.07 second window (0.07 seconds was the default value used by mir_eval library) of the actual start of the beat. This metric is useful in its ubiquity and simplicity. Other useful metrics include cemgil score, which is like F-measure but it uses a gaussian distribution over each beat instead of a hard window. Other scores like P-score are more complicated but still near ubiquitous while several continuity

Common Beat Tracking Scores[12]

AMLt	*AMLc*	*CMLt*	*CMLc*	*P Score*	*F Measure*	*Cemgil*
0.611	0.435	0.582	0.543	0.728	0.694	0.528

Scores calculated as an average for each file across the validation set

scores exist that score on the basis of providing continuously correct predictions over time. Shown is a chart of the average performance on all metrics of the beat tracking system over the entire validation set. The scores presented are on par with the scores of the papers cited earlier, though because the datasets used for evaluation are not identical it is impossible to compare performance.

Generally, the system performs well on audio that contains strong beats such as disco and pop songs as well as some rock songs, with the system perfectly scoring across all metrics on several songs withing the validation set. Other songs where the beats are less pronounced by rhythmic instruments proved more challenging to the system such as songs in the jazz and classical genre. Below is a distribution across the songs in the validation set for one particular metric, F-Measure.

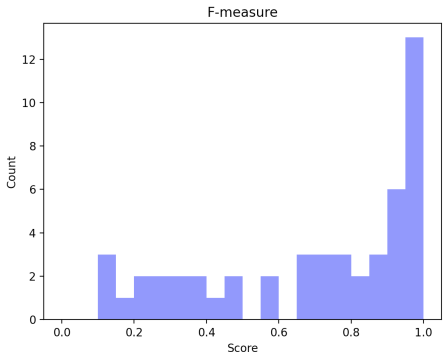

Figure 4: A histogram of the scores for the songs included in the validation set for the F-Measure score, a metric commonly used when evaluating beat trackers and which accounts for accuracy, precision, and recall of the beat predictions. Many songs in the validation set received near perfect scores across all metrics.

6 Conclusion/Future Work

For a first attempt at a beat tracking system I am quite pleased with the results while there are still so many clear avenues to try to improve the system. The system works well for songs that contain heavily emphasized beats like dance music while performing poorly on genres like jazz and classical music. Of the avenues to explore next, one of the options that seems to promise the greatest reward is using RNNs for the neural network to capture time dependencies between audio images. This was briefly explored as a part of this project but was discarded for the sake of simplicity and getting an end to end solution working in the time frame allotted. Also, great improvements can be made in the way that tempo and beat locations are calculated from the output of the neural image. Strategies from [2] will provide vastly more reliable predictions for tempo and beat locations while statistical models may be promising as well. Lastly, improvements can be made to the preprocessing of the audio. Different window sizes and numbers of frequency bins for the STFT can be explored to see which audio representations allow the neural network to make the best estimations of tempo and beat location.

Future work also includes attempting to run the system on an embedded device like a Raspberry PI so that the system can easily be used in a live music setting to control other instruments like a a drum machine that will accompany live musicians.

7 Contributions

This project was completed by a single individual. Given more people it would have helped to try many models in parallel to test different hyperparameters. Also, more members would have allowed for greater parallelization across the whole project from data preprocessing, setting up the model, to building the system that outputs the predictions of beat and tempo.

References

[1] Ali Mottaghi, Kayhan Behdin, Ashkan Esmaeili, Mohammadreza Heydari, and Farokh Marvasti. Obtain: Real-time beat tracking in audio signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017.

[2] Peter Grosche and Meinard Müller. Extracting predominant local pulse information from music recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1688–1701, 2011.

[3] Sebastian Böck, Florian Krebs, Amaury Durand, Sebastian Pöll, and Raminta Balsyte. Robod: a real-time online beat and offbeat drummer. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017.

[4] Sebastian Böck, Matthew E.P. Davies, and Peter Knees. Multi-task learning of tempo and beat: Learning one to improve the other. In *20th International Society for Music Information Retrieval Conference, Delft, The Netherlands*, 2019.

[5] Retrieval Marsyas (Music Analysis and Synthesis for Audio Signals). Gtzan genre collection. `http://marsyas.info/downloads/datasets.html`.

[6] Ugo Marchand, Quentin Fresnel, and Geoffroy Peeters. Gtzan-rhythm: Extending the gtzan test-set with beat, downbeat and swing annotations. hal-01252607, 2015.

[7] Brian McFee, Vincent Lostanlen, Matt McVicar, Alexandros Metsai, Stefan Balke, Carl Thomé, Colin Raffel, Ayoub Malek, Dana Lee, Frank Zalkow, Kyungyun Lee, Oriol Nieto, Jack Mason, Dan Ellis, Ryuichi Yamamoto, Scott Seyfarth, Eric Battenberg, , Rachel Bittner, Keunwoo Choi, Josh Moore, Ziyao Wei, Shunsuke Hidaka, Nullmightybofo, Pius Friesch, Fabian-Robert Stöter, Darío Hereñú, Taewoon Kim, Matt Vollrath, and Adam Weiss. librosa/librosa: 0.7.2, 2020.

[8] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[9] François Chollet et al. Keras. `https://keras.io`, 2015.

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[11] George Woskob. Finding rhythm in music using kalman filters optimized with local search. `https://github.com/aphera/aa228-final-project/blob/master/final.pdf`.

[12] Matthew Davies, Norberto Degara Quintela, and Mark Plumbley. Evaluation methods for musical audio beat tracking algorithms, 10 2009.

[13] Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. mir_eval: A transparent implementation of common mir metrics. In *Proceedings of the 15th International Conference on Music Information Retrieval*, 2014.