Coursework Assignment – Joint Beat and Tempo Tracking for MIDI

Tolly Collins –Queen Mary University, London

1 Introduction

There have been many studies conducted on beat tracking for audio data, with a wide variety network architectures being implemented. A long-running state-of-the-art model was based on a Recurrent Neural Network (RNN) followed by a Dynamic Beysian Network (DBN) post-processor [1]. Other approaches have included Adaptive Frequency Neural Networks (AFNN) [2] which utilise oscillators that resonate at integer multiples of the music's pulse. This approach mirrors prior work using digital signal processing techniques such as the Predominant Local Pulse [3] which use either short Time Fourier Transforms (STFTs) or Autocorrelation to construct a tempogram based on local periodicities. Much like with the AFNN approach, the amplitude and phase of the periodicity at different frequency bands can be used to infer the localised tempo, which in turn informs the beat tracking process.

Recently, Temporal Convolutional Networks (TCNs) have been shown to be able to perform audio beat tracking at state-of-the-art level, while using significantly fewer parameters than RNNs with similar performance levels [4]. TCNs use exponentially increasing dilation factors to achieve a wide receptive field which means that the nodes at the end of the network receive a large amount of contextual information from the musical sequence. However, as these networks do not have a recurrent element, the backpropagation can be parallelised and the problems of vanishing gradients are much less severe [5].

The concept of leveraging tempo information to inform beat tracking has also been applied by Bock et al in the form of a multi-task neural network [6]. In this case, a TCN was used for beat tracking, but this was linked via skip connections to a tempo classifier. They found that the network performed the beat tracking with the additional tempo branch as well as it did without. However, there was no significant improvement in performance. Intuitively, the representation of the tempo information in the latent space of the convolution layers should be able to aid the network in its beat tracking task. However, the structure and balance of the network branches and the loss function will be crucial in determining whether or not a performance improvement can be achieved.

There have been relatively few applications of deep learning for MIDI beat tracking. In 2020, Chuang & Su claimed that they were the first researchers to implement such a model [7], applying three variations of RNN models. This study aims to combine the approaches described above, where we use a multi-task TCN network similar to that of Bock et al, with the application to MIDI data informed by the work of Chuang & Su. We consider two separate networks for the tasks of beat and tempo tracking, and we then investigate combining these into a single multi-task network. Furthermore, we investigate the application of a loss network with learnable parameters.

2 Methodology

2.1 Data

For this study, we use the ASAP dataset [8]. This is a set of time-aligned MIDI and audio recordings of Classical piano performances, originally from the MAESTRO dataset [9], with additional labels. Here, we make use of the annotated beat and downbeat times. Following the approach of Chuang & Su [7], we first transform the MIDI data into a piano roll, with a sample rate of 100 time points per second. We then map the beat times to corresponding labels, with a value of 1 at a beat time and 0.5 at the time points either side. This smoothing of the labels is designed to help the model learn. For the calculation of tempo labels, we primarily follow Bock et al's method [4] of taking a time-based histogram from inter-beat intervals and smoothing with a normalised Hamming window filter of length 15 samples. We then take the smoothed tempo values at the beat positions and apply quadratic interpolation to get beat labels corresponding with the piano roll time points. Our deviation from Bock et al's method is to use single continuous beat labels, rather than one-hot discrete labels. As tempo operates on a continuous scale this approach could allow the model to learn more efficiently. We allow pieces with smoothed tempi in the range [10, 360] and we remove pieces with inconsistent beat annotations.

It should be noted that these choices of input features depart somewhat from those of Chuang & Su [7], who followed a more traditional approach of crafting onset, Inter-Onset-Interval and spectral flux features. Here, we use a deeper neural model, and we supply only the piano roll data. We view the MIDI piano roll as an analogy to a spectrogram representation, where magnitude is represented by velocity and frequency by MIDI notes. Unlike some studies [10], we input all 88 possible note values. We also experimented with representing times with no notes playing as an 89^{th} 'rest' frequency bin.

2.2 Models

The beat tracking model first consists of a 2D convolution feature extraction block with 3x3 kernels, made up of three layers. Max-pooling is applied in the frequency axis, but the time dimension is kept intact [4]. For the TCN block, we use a marginally shallower 10 layers compared to Bock et al's 11, with the assumption that MIDI data inherently contains less information. We did not wish to reduce the number of layers further as this would significantly reduce the receptive field of the final layers. The resulting network had a receptive field of 4039 time points, corresponding with just over 40 seconds of music. This was followed by a dense layer to reduce the number of channels down to 1 for the output, which was passed to a binary cross-entropy loss. There was also an additional optional output branch to track downbeats.

The tempo tracking model was a standard Convolutional Neural Network (CNN). As it received the same input data as the beat tracking model, we used the same architecture for the 2D convolutional block. This was followed by 1D convolution layers with max-pooling, which down-sampled by a factor of 8. This allowed the output to have a coarser time resolution more fitting for tempo tracking. The tempo labels were down-sampled by the same factor. A mean-squared-error loss function was used.

For the multi-task network, the beat tracking structure was kept intact from the individual beat tracking network. For the tempo branch, the 1D convolution section was replaced by a mean-pooling layer over the time dimension which gave the same down-sampling factor as with the

individual network. This was connected to the TCN block via skip-connections, as this has been found to improve performance relative to simply connecting it to the output [6].

The loss function network had three parameters, corresponding to the weightings of the beat, downbeat and tempo elements of the loss function. When used, the weights were learned alongside the training of the main network.

2.3 Sample size and Data Augmentation

As a result of the receptive field requirements described above, we decided to give uniform-length input sequences to the models. These all had 4349 time points, equivalent to 256 receptive fields translated by 1 position. In order to add variety to the input data to help avoid overfitting, the starting position for each example was not fixed. Rather, the pieces were designated intervals within which each sample randomly began.

A further strategy applied was to augment the data in two ways. The first was a random pitch transposition for each sample within a given interval, and the second was a random time stretch (equivalent to a proportional change in tempo) within a permitted interval. It has been shown that using dynamic data augmentation gives a greater improvement in performance for tempo and beat estimation compared with pre-processed data augmentation, as the model is presented with an infinite possible set of data compared with just a larger pre-designated set [11]. This augmentation was therefore calculated as each sample was requested by the Data Loader.

3 Results

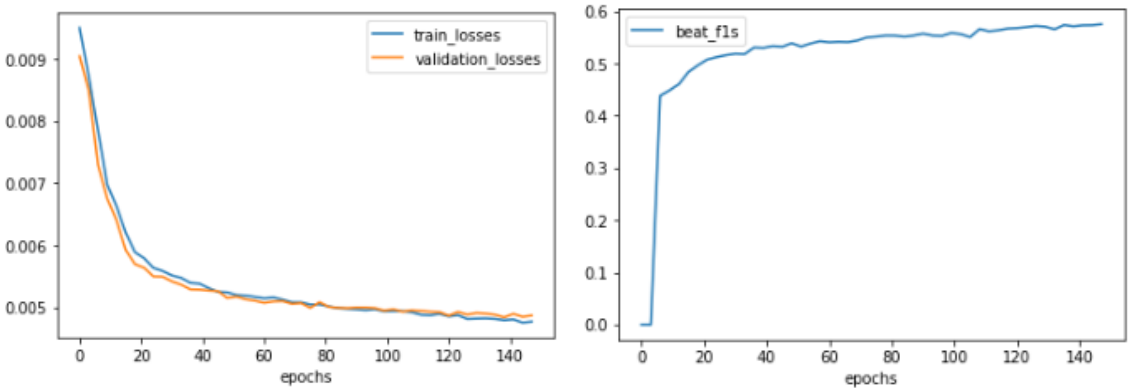
3.1 Beat Tracker

The main metric used to assess the performance of the beat tracking in this study is the F1-measure. We use the standard tolerance window of 0.07 seconds around a beat time. This means that for a piece with tempo of 120 BPM, 14% of the possible time points are within a tolerance window.

As a baseline for comparison, Cheung et al achieved an F1-score of 0.605 on the MusicNet database. It should be noted that no direct comparison can be made with the results in this study as a different database is used, but this can serve as a reference nonetheless.

At the beginning of the training process, the beat tracking network was achieving F1-scores which plateaued in the region of 0.35. We adjusted hyper-parameters such as the learning rate and the post-processing peak picking filter length but there seemed to remain a ceiling on performance.

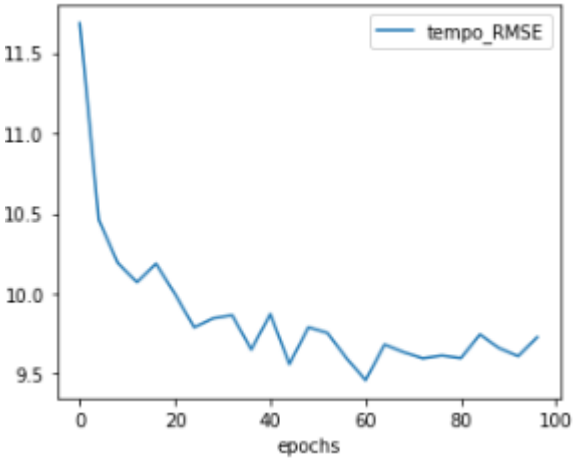
The next approach was to assign a weighting to the positive class in the loss function. This was done to combat the issue that beat data is very imbalanced. At a tempo of 120 BPM and a sample rate of 100, there will be 49 negative targets for every positive target. When using a weighting of 4 for the positive labels, an F1-score of 0.61 was achieved on the validation dataset, which is in line with the state-of-the-art performance on the MusicNet database.



The figure above shows example learning curves from the beat tracker over 150 epochs, and the corresponding F1-scores on the validation set.

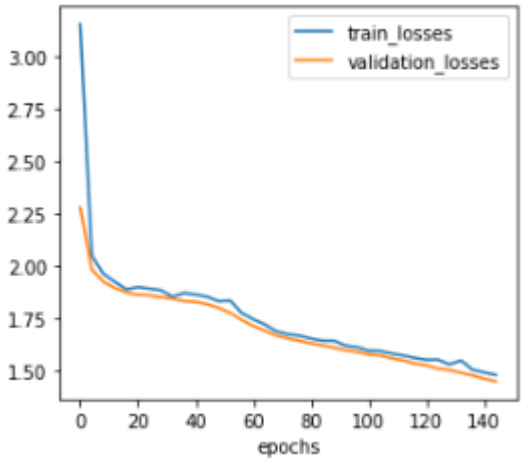
3.2 Tempo Tracker

The performance of the tempo tracker was measured by root mean squared error between target and predicted tempi in BPM. We experimented with different number of convolutional layers, channels and dropout rates and settled with 4 layers with 16 channels each and a dropout rate of 0.1. The best RMSE score achieved on the validation set was 9.5 BPM, shown in the graph below.



3.3 Multi-task Beat and Tempo Tracker

With the multi-task network, the primary challenge was balancing the loss function to allow both branches to learn effectively. The loss of the tempo branch was naturally higher than that of the beat tracking branch, and so the network seemed to prioritise learning the tempo representation to the detriment of the beat tracking representation in the convolutional layers. In this scenario, the F1-score for the beat tracking remained below 0.4. We experimented with different loss weightings to counteract this effect, and settled on a tempo loss weighting of 0.2. This allowed the beat tracking F1-score to rise to just over 0.5, but it could not get as high as it did in the single-task network. Likewise, the performance of the tempo branch was weaker than the performance achieved in the single-task tempo network. However, on inspecting the learning curves, it became apparent that the network was learning more slowly than the single-task networks were. We see in the figure below that the loss is still steadily decreasing in this example run of 150 epochs. In future work it would be worth investigating further hyper-parameter adjustments to make the learning more efficient.

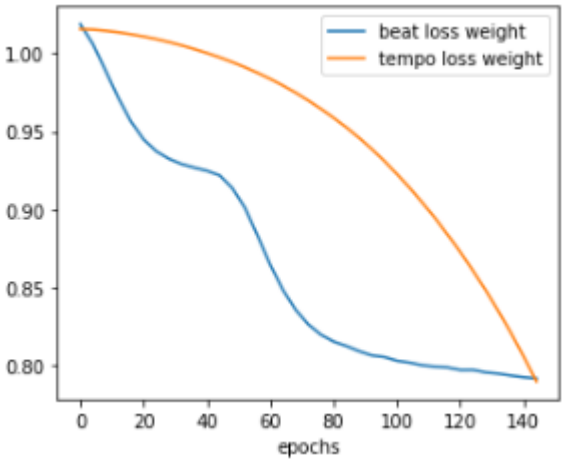


3.4 Learnable loss weights

We experimented with allowing the network to learn weights to balance the losses of the two branches. We began by implementing an approach similar suggested by Liebel & Korner [13], with a loss function given by

$$L_T(y_T, y'_T) = \sum_{t \in T}\left(\frac{1}{2c_t^2} L_t(y_t, y'_t) + \ln(1 + c_t^2)\right)$$

Here, each c_t is a learnable weight parameter and $\ln(1 + c_t^2)$ is a regularisation term to avoid degenerate solutions. However, we discovered that this approach suffered from an initial imbalance where the weights would decrease to reduce the regularisation term, as seen by the graph below:

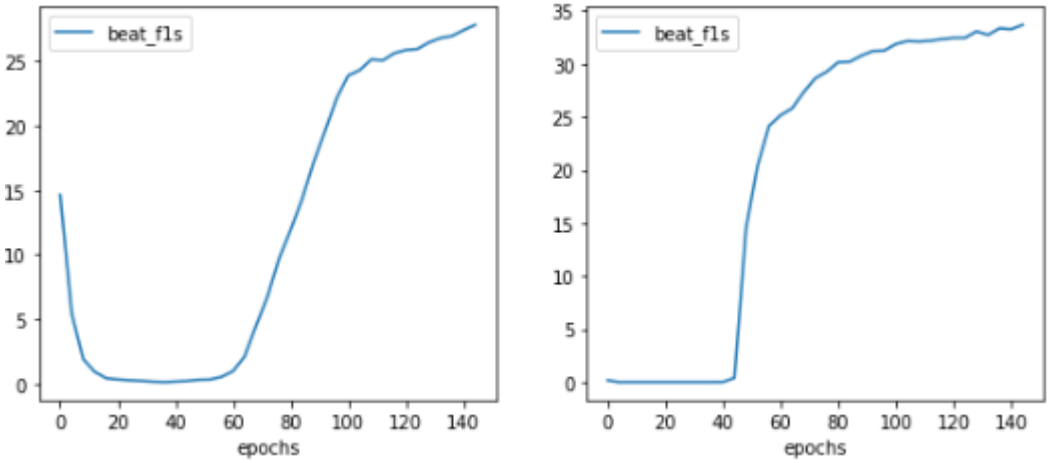


This resulted in much slower learning for the rest of the network. The aim here was for the two loss weights to be regularised to avoid degeneration, but at the same time to be able to vary from each other rather than reaching a common equilibrium. We found that a solution to this was to reduce the level of regularisation on the learned weights, but then to add a second regularisation term for the sum of the two weights. This would then allow one to increase relative to the other, with both not being allowed to vary too far from their initial value of 1. As the weights were being applied as multipliers, the absolute value of a logarithm of the weight value was used so that a proportional increase or decrease from 1 would be penalised by roughly the same degree. This gave:

$$L_T(y_T, y'_T) = \sum_{t \in T}(m_t \cdot |c_t| \cdot L_t(y_t, y'_t) + \lambda_1 |\ln(c_t)|) + \lambda_2 \left|\ln\left(\frac{1}{|T|}\sum_{t \in T} c_t\right)\right|$$

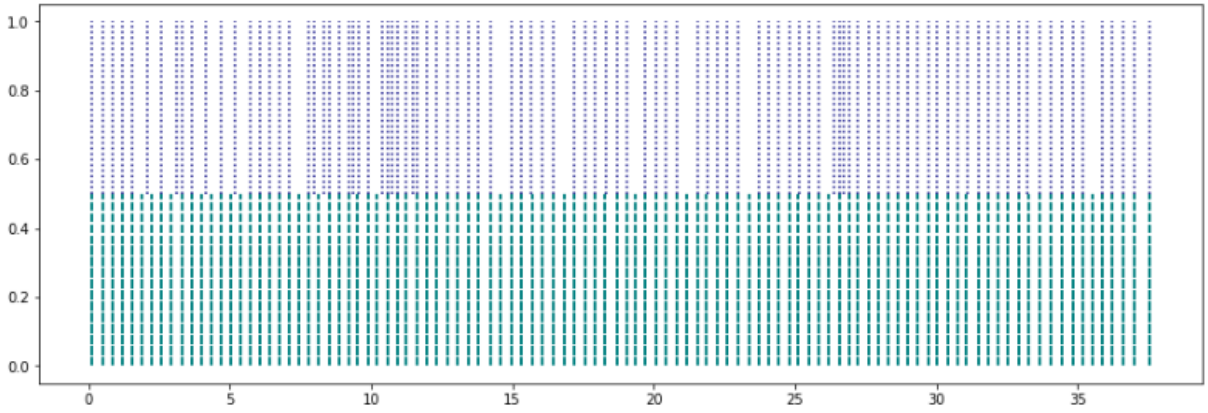
In the above function, m_t are manual weightings which can be given to each loss term, c_t are the learnable weights and λ_i are the regularisation weightings which can be adjusted as hyper-parameters.

We found that the training did generally improve with this loss function, as opposed to a non-learnable loss function. The graphs below demonstrate a comparison in learning for F1-scores with the standard loss function applied on the left and the loss function with learned weights applied on the right. Further investigation is needed to determine if the overall predictions of a fully trained network can be improved with this approach, and whether there is a reliable improvement in training efficiency that can be achieved once hyper-parameters are tuned more thoroughly.



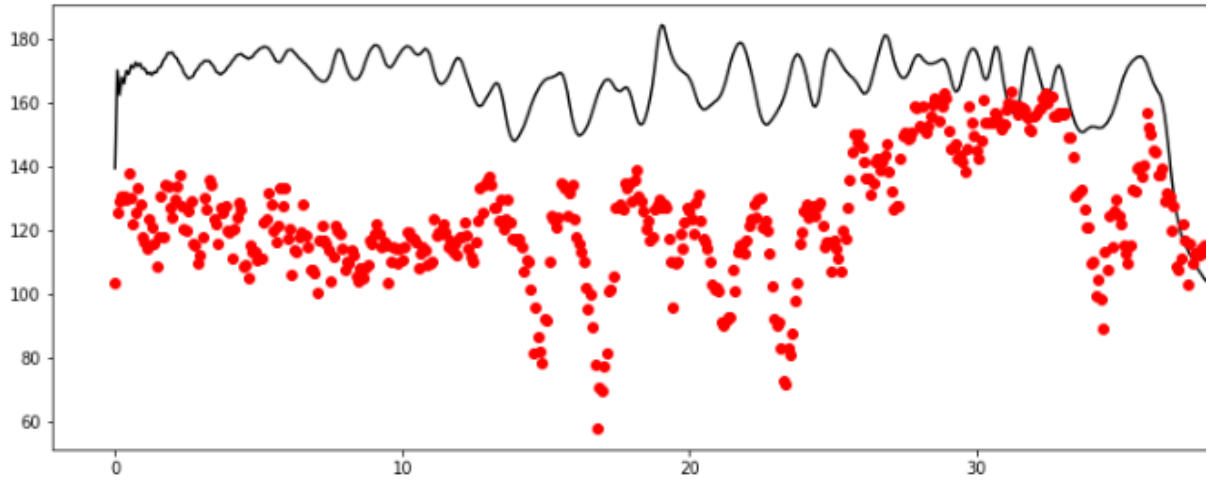
3.5 Case Study

The graph below shows the beat tracker's output predictions (blue lines) relative to the ground truth labels for a performance of Mozart's Sonata No.12, movement 1. This is the same excerpt as the accompanying audio file (a synthesised version of the MIDI file with superimposed beat predictions).



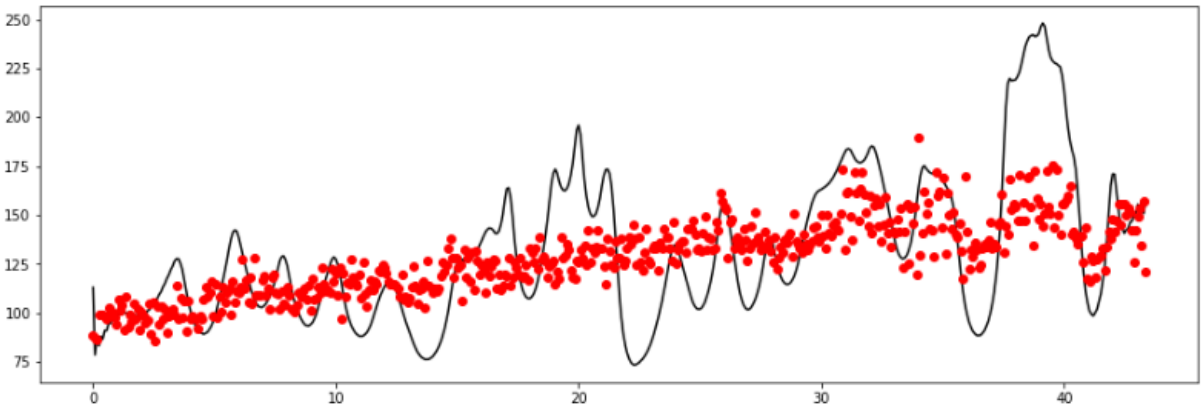
We see that for this piece, the model is able to determine the beats with a good level of accuracy, although there are both missed beats and cases of double beating. The f1-score for this excerpt was 0.86, which was above average for the performance of this model. It should be noted that the model generally performed less well on pieces with more tempo variation, whereas the relatively steady tempo of this Mozart piece may have aided the model's beat tracking.

The tempo tracker did not perform well on this excerpt, as shown below:



We see that it is tracking an incorrect tempo for the majority of the piece, only moving up toward the correct region towards the end.

We note that the performance of the tempo tracker, both individually and part of the multi-task network was generally weaker. Below is a typical example of where the tempo tracker is able to follow a general trend without being able to react to the more localised tempo changes.



4 Discussion

The system described here shows significant promise for both beat and tempo tracking. In particular the f1-scores of the beat tracking part of the system are similar to those shown in other MIDI beat tracking studies. The learning curves demonstrate that the models are all converging towards better performance as they train, but it is clear that further work needs to be conducted to tune the models to bring out better performance.

The tempo results shown in the case study above suggest that the model may currently be under-fitting, as the output curves only react to a general trend, rather than the finer-level detail. In our implementation, we followed the suggestion of Bock et al [6] to sum the outputs of the skip connections for the multi-task network. However, this may result in a loss of the finer-resolution information, and perhaps a concatenation would allow more of the localised sequence information to come through. It should also be noted that the tempo information was down-sampled by a factor of 8 in the time domain, again followed the suggestions of prior work. If we view the tempo branch

of the multi-task network simply as a learning aid for the beat tracking branch in terms of organising the information in the convolution space representation, the general trend may very well be more useful than finer-grain tempo fluctuation information. We suggest the investigation of the effects of different down-sampling ratios for further work. We also suggest investigating the implementation of skip connections between layers in the TCN architecture to aid the learning process.

The implementation of the learnable weights for the multi-task loss function shows potential for the use of weight regularisation by collective mean, rather than simply by individual weight value. The multi-task model's learning was most stable under these conditions.

The use of data augmentation and random example start points did seem to aid the model's learning, but with the resources available we were not able to run the model for enough epochs to see whether or not the final performance ceiling was increased under these conditions.

References

[1] – Krebs et al (2016) - Downbeat Tracking Using Beat-Synchronous Features and Recurrent Neural Networks. *ISMIR*

[2] – Lambert et al (2016) - Adaptive Frequency Neural Networks for Dynamic Pulse and Metre Perception. *ISMIR*

[3] – Grosche (2012) - Signal Processing Methods for Beat Tracking, Music Segmentation, and Audio Retrieval. *Max-Planck-Institut fur Informatik Saarbrucken, Germany*

[4] - Davies & Bock (2019) – Temporal convolutional networks for musical and audio beat tracking. *EUSIPCO*

[5] - Bai et al (2018) - An empirical evaluation of generic convolutional and recurrent networks for sequence modelling. *arXiv*

[6] – Bock et al (2019) - Multi-Task Learning of Tempo and Beat: Learning One to Improve the Other. *ISMIR*

[7] – Chuang & Su (2020) - Beat and Downbeat Tracking of Symbolic Music Data Using Deep Recurrent Neural Networks. *APSIPA*

[8] – ASAP dataset: https://github.com/fosfrancesco/asap-dataset

[9] - MAESTRO dataset: https://magenta.tensorflow.org/datasets/maestro

[10] – Brunner et al. (2018) - Midi-Vae: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer. *ISMIR*

[11] – Bock & Davies (2020) - Deconstruct, Analyse, Reconstruct: How to Improve Tempo, Beat, and Downbeat Estimation. *ISMIR*

[12] – Tony-Y (GitHub, 2021) - https://gist.github.com/Tony-Y/9e3687fbe10e817596d1e1ed58c9f191

[13] – Liebel & Korner (2018) - Auxiliary Tasks in Multi-task Learning. *Technical University of Munich*