BEAT THIS!
ACCURATE BEAT TRACKING WITHOUT DBN POSTPROCESSING

Francesco Foscarin[*1,2] **Jan Schlüter**[*1] **Gerhard Widmer**[1,2]
[1] Johannes Kepler University, Linz, Austria
[2] LIT AI Lab, Linz Institute of Technology, Austria
firstname.lastname@jku.at

ABSTRACT

We propose a system for tracking beats and downbeats with two objectives: generality across a diverse music range, and high accuracy. We achieve generality by training on multiple datasets – including solo instrument recordings, pieces with time signature changes, and classical music with high tempo variations – and by removing the commonly used Dynamic Bayesian Network (DBN) postprocessing, which introduces constraints on the meter and tempo. For high accuracy, among other improvements, we develop a loss function tolerant to small time shifts of annotations, and an architecture alternating convolutions with transformers either over frequency or time. Our system surpasses the current state of the art in F1 score despite using no DBN. However, it can still fail, especially for difficult and underrepresented genres, and performs worse on continuity metrics, so we publish our model, code, and preprocessed datasets, and invite others to beat this.

1. INTRODUCTION

Beat tracking is the task of estimating the temporal locations of musical beats in an audio signal. It is often combined with the downbeat tracking task, which targets a higher metrical level: tracking the beginning of each measure. Despite being one of the long-standing problems in the Music Information Retrieval (MIR) field, it still attracts attention and several approaches were proposed in recent years [1–8]. Most recent work follows a common pipeline: the audio files are transformed into some spectrogram-like representation, then a deep neural network predicts frame-wise beat and downbeat probabilities, which are postprocessed to obtain the final predictions. The most widely used postprocessing technique is the Dynamic Bayesian Network (DBN) in the form proposed by Böck et al. [9]. It addresses four tasks: variable threshold peak-picking, forcing the tempo to stay in a certain range (i.e., limiting the allowed distance between beats), limiting sudden tempo changes, and (for downbeat tracking) ensuring that the downbeat falls every n beats, where n is constant for a piece of music and is selected from a limited list of values.

We argue that the DBN is a problematic component because it is inherently bound to fail for several music pieces: pieces with time signature changes, pieces whose tempo falls outside of the tempo range (or that slow down/speed up outside the tempo range), and pieces whose number of beats per measure are not included in the list of supported values. Moreover, it has a fixed parameter controlling allowed tempo variations, although we can expect, for example, classical music to have bigger tempo variability than rock music. Finally, even the hypotheses of having periodic beats and downbeats may be invalid, for example, for songs where the players make mistakes or audio tracks containing multiple concatenated songs.

Still, the DBN performs well on most pieces commonly used to train and evaluate beat tracking systems: music with a constant time signature of 3/4 or 4/4 and a stable, medium tempo. This can be seen from the default DBN parameters which most systems use,[1] i.e., tempo range $[55, 215]$ BPM, beats per measure $[3, 4]$, and a tempo variability optimised on pop, rock and dance datasets. Pieces outside these specifications are likely to be mispredicted by the system, but form a minority in typical datasets. Therefore, in terms of evaluation metrics, it usually does not pay to remove the DBN. However, working in these "simplified" conditions blocks research from solving corner cases in existing datasets and targeting more challenging or diverse data.

Our first goal is thus to replace the DBN with minimal postprocessing, free of the aforementioned musical assumptions. A recent attempt to remove the DBN was made by Chen and Su [2]. However, their system may not look appealing to practitioners requiring beat tracking for downstream tasks, or researchers seeking a system to improve, as its accuracy falls clearly behind DBN-based ones.

Our second goal is to provide a powerful basis for practitioners and researchers. The current best-performing system (which uses a DBN) from Hung et al. [10] falls short in this regard, as its code or a pretrained model is not public, its architecture is very complex, and (to the best of our knowledge) the results could not be reproduced by others.

* Equal contribution.

[1] We could verify that [3, 4, 8] use these parameters, since their code is publicly available, and we assume [1, 10] do as well, since they do not mention any details in their paper.

Attribution: F. Foscarin, J. Schlüter, and G. Widmer, "Beat this! Accurate beat tracking without DBN postprocessing", in *Proc. of the 25th Int. Society for Music Information Retrieval Conf.,* San Francisco, United States, 2024.

In this paper, we present an open-source system that obtains new state-of-the-art F1 scores without a DBN. It is based on a rotary transformer [11] applied on spectrograms, with the following novelties: (1) We design a frontend alternating convolutions with a transformer variant by Lu et al. [12] that attends alternatively over frequency bins or time frames. (2) We train with a shift-tolerant binary cross-entropy (BCE) that can cope with small deviations in the beat/downbeat annotations, and with weights on beat/downbeat frames to balance their relative scarcity. (3) We propose an approach that encourages downbeat predictions to be a subset of beat predictions, and (4) a data augmentation masking input segments to encourage the network to consider a longer context. Our code, pretrained models, and preprocessed datasets are openly available. [2]

2. RELATED WORK

The currently best-performing model (on the GTZAN dataset [13] commonly used for evaluation) is by Hung et al. [10] and serves as a point of comparison. It uses a complex neural network architecture named SpecTNT which alternates computing frequency-related features with a frequency-oriented transformer, and processing a virtual extra frequency band with a time-oriented transformer. This runs in parallel with a more widely used Temporal Convolutional Network (TCN, a fully convolutional network with dilated convolutions), and the outputs of the two networks are merged for the final predictions. Unfortunately, the approach is not open source, and to the best of our knowledge, no other research group has managed to reproduce its results. Moreover, it still uses the DBN, which, as argued in the introduction, limits the system's generality.

Although no other work could reach the accuracy reported by Hung et at., two other recent beat tracking papers brought new interesting ideas [3, 4]. Both perform instrument separation (with a pretrained network) and feed the separate stems (bass, drums, vocals, other, for [3] also piano) into the model, mixing their information in cross-instrument attention blocks. While this approach is very reasonable from a music perception standpoint, it reduces the generality of the system, since it assumes that the input pieces will contain such instruments, at least to some extent. Another proposal of [3, 4] is the use of dilated attention, following the successful use of dilated convolutions in beat-tracking architectures to increase the receptive field without adding computations. We find that flash attention [14] enables us to train with dense attention over a satisfactorily large input size, and leave experiments with dilated attention for future work.

Chen and Su [2] try to remove the DBN and propose a set of improvements. The most impactful is to replace the BCE with the Dice [15] and Focal [16] loss, inspired by their common usage for medical image segmentation. While these losses improve results, possibly due to their inherent ability to handle unbalanced classes, we found that a BCE with weights on the positive (beat and downbeat) classes outperforms them. We suspect this is because, in contrast to medical image segmentation, our positive examples are single frames, and the Dice and Focal loss perform better when the area of positive predictions is larger [17]. Another proposal by Chen and Su is to predict the phase of the beat/downbeat instead of a single binary value, following [18]. Although this seems promising, the results on both papers (and ours) do not show any consistent improvement.

Many recent approaches [2, 3, 18, 19] use the additional task of tempo prediction (a single tempo target for each input excerpt) in a multi-task setting. While this improves their results, it goes against our goals of generality, since it assumes an (almost) constant tempo in the file excerpt, which is not the case for many kinds of music.

Other recent papers do not align with the goal of this paper: [1] explores the usage of different time resolutions between the input audio and output predictions (only addressing beats); [8] performs unsupervised beat tracking; [20] focuses on online beat tracking; [5] notices the problems of the DBN for music with tempo changes, and proposes a different postprocessing method targeted specifically to classical music; [7] focuses on fine-tuning existing systems, and changing the DBN parameters for targeting specific underrepresented genres.

3. METHOD

Our beat tracker is based on a neural network with $\sim$20 M parameters. It starts from 30 seconds of mono audio sampled at 22.05 kHz and converts it to a 128-bin mel spectrogram from 30 Hz to 10 kHz, with a window size of 1024 and hop size of 441 samples (yielding 50 frames per second), and magnitudes scaled via $\ln(1 + 1000x)$ (similar to $\ln(\max(10^{-3}, x))$, but maps silence to zero). These hyperparameters were optimised in preliminary experiments. Our model processes this into frame-wise beat and downbeat probabilities, followed by minimal post-processing to derive beat and downbeat locations.

3.1 Model

Our model (Figure 1) processes a $T \times 128$ spectrogram into $T \times 2$ probabilities; T being the number of input frames (1500 in case of 30 s). It consists of three components: a frontend converting the spectrogram into a sequence of feature vectors, a transformer processing these vectors, and two task heads computing the output probabilities.

3.1.1 Frontend

The frontend's role is to integrate information across the 128 frequency bands into feature vectors. Typically, this is done via 2d convolutions gradually reducing the number of bands to 1 while increasing the number of channels [3, 10]. We adopt this, but found it helps to interleave convolutions with *Partial Transformers*, which treat the time and frequency axis independently. Overall, our frontend consists of a stem, three identical blocks, and a linear projection.

The stem (Figure 1, top right) starts with a batch normalisation that processes each frequency band separately

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

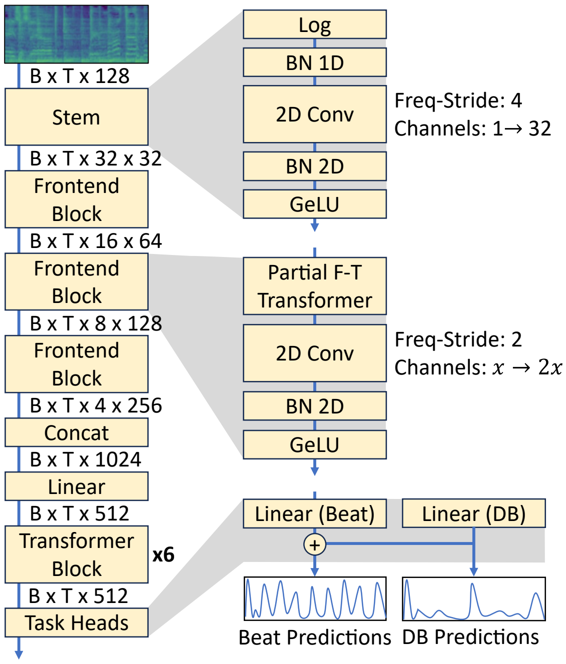

Figure 1: Full model architecture.

to homogenise them, followed by a 2d convolution of 3×4 kernels, regular batch normalisation, and GeLU nonlinearity. The convolution is strided to reduce the number of frequency bands to a fourth and creates 32 channels.

Each block (Figure 1, middle right) consists of two partial transformers, a strided 2d convolution halving the number of bands while doubling the number of channels, batch normalisation and GeLU. The first partial transformer is *frequency-directed*, i.e., it processes the $T \times F \times C$ tensor by treating each time frame as a sequence of length F (the number of bands), the second one is *time directed* and treats each frequency band as a sequence of length T (the number of frames), an idea adopted from the Band Split RoFormer [12]. Each transformer has a head size of 32 (one head in the first frontend block, two in the second, four in the third), rotary positional embedding [11], a sigmoid gate per head [21, Sec. 4.2], and includes a usual pointwise feedforward network with a hidden size of four times the channel count.

After three frontend blocks, the resulting $T \times 4 \times 256$ tensor (4 bands, 256 channels) is reshaped to a $T \times 1024$ tensor and linearly projected to 512 features.

3.1.2 Transformer

The transformer makes up the bulk of our model's parameters and compute. It consists of 6 stacked transformer blocks processing the 512-dimensional vectors with 16 heads of size 32, rotary positional embedding, sigmoid gating, and a pointwise feedforward network of 2048 hidden units. This matches the configuration in the frontend transformer blocks, but as it processes a single sequence of 512-dimensional feature vectors, it is a regular transformer over time without separately considering a frequency dimension. Its goal is to map the 512 input features to a space that relates to beats and downbeats. Due to the attention mechanism, the transformer's receptive field covers the full sequence, and it could therefore produce an output that has characteristics that we want in the beat predictions, for example, regularity.

3.1.3 Task Heads

The output of the final transformer block is processed by two linear layers, one for beats and one for downbeats. Initially, we used the common approach of passing their output into 2 sigmoid functions to produce a probability for each input frame, and then threshold this probability at 0.5 to obtain "hard" beat predictions. However, we observe that this sometimes produces downbeat predictions not coinciding with a beat prediction, which is allowed under the evaluation metrics but is a musically invalid and unusable output. This problem is solved when using a DBN to jointly process beats and downbeats. However, we noticed that several works, e.g., [3,6], use two independent DBNs to predict beats and downbeats (and others [2, 10] do not specify). To our surprise, this leads to better metrics, but it severely limits practical use.

To mitigate this problem, we propose a *Sum Head* that sums the output of the beat and downbeat layers, and treats this as the beat logits (for prediction and training). This is a very simple way of helping the network produce a beat when there is a downbeat (though it does not enforce that; a highly negative output of the beat layer can still counter the downbeat layer). We explored other ways of aggregating the beat and downbeat logits, like taking their maximum, but this hampered training due to the sparser gradients. On the GTZAN dataset, the sum head almost halves the percentage of downbeats that are more than 70 ms away from the closest beat, from 1.1% to 0.62%, compared to directly using the output of the linear layers. We observe that the remaining unmatched downbeats are in pieces with very erratic predictions that would be incorrect anyway.

Some systems circumvent the problem by using a 3-way classifier (beat vs. downbeat vs. none) instead of the two binary classifiers (beat vs. none, and downbeat vs. none). However, to be able to train on datasets that do not include downbeat annotations, we stick to binary classifiers.

3.2 Postprocessing

To obtain beat/downbeat locations, we pick all frames assigned the highest beat/downbeat probability within a neighborhood of ± 3 frames (± 70 ms), and probability > 0.5. In case adjacent frames are assigned the same probability, we report their center. Finally, we move all downbeat predictions to the closest beat prediction to correct the remaining mismatches described in the previous section. For music pieces longer than 30 s, we concatenate predictions over non-overlapping 30-second excerpts.

3.3 Loss

The model is trained by gradient descent on a loss function that compares the frame-wise beat and downbeat predictions with frame-wise binary annotations. The usual loss for binary classification is Binary Cross-Entropy,

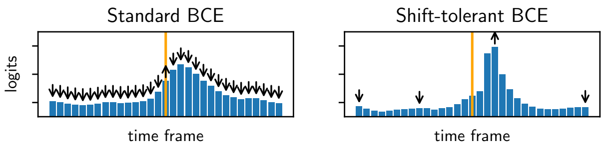

Figure 2: The standard binary cross-entropy loss (left plot) encourages high network outputs (upward arrow) at beat annotations (vertical line), and low outputs for all other frames (downward arrows). Max-pooling the predictions over time redistributes gradients to local maxima (right plot). This way, slightly shifted annotations do not affect learning, and the network produces confident sharp peaks.

$L_{\text{bce}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_t y_t \log(\hat{y}_t) + (1-y_t)\log(1-\hat{y}_t)$. Training with BCE leads to unconfident predictions since the problem is heavily imbalanced. To counter this, we can weight positive examples by a factor w as $L_{\text{wbce}}(\mathbf{y}, \hat{\mathbf{y}}, w) = -\sum_t w y_t \log(\hat{y}_t) + (1-y_t)\log(1-\hat{y}_t)$. We found that setting w to the number of negative examples divided by the number of positive examples (over the training set) yields the best result and is crucial when not using a DBN.

Another problem persists: annotations are not precise down to our spectrogram resolution, due to annotators' impreciseness, players' asyncronicity, or simply the limits of human perception. This is taken into account during the evaluation, e.g., the typically used F1 score accepts predictions in a ± 70 ms window around labels. During training, the BCE loss punishes close positive predictions (Figure 2, left) even though they may be correct, thus creating two problems: training is slower and the network learns to predict wide "blurred" peaks. This is commonly addressed by adding two extra positive labels around each annotation weighted by 0.5, but this only mitigates the former problem without helping with the latter. Instead, we max-pool predictions over time (7 frames, stride 1) before comparing them to the labels. In this way, only the largest positive prediction ± 3 frames from each label is considered (Figure 2, right). The loss for negative examples is ignored ± 6 frames from each label, as this is how far a max-pooled prediction 3 frames away from a label spreads. Denoting max-pooling of k frames with $m_k(\cdot)$, we can formalise our *Shift-tolerant weighted BCE* as: $L_{st}(\mathbf{y}, \hat{\mathbf{y}}, w) = -\sum_t w y_t \log(m_7(\hat{\mathbf{y}})_t) + (1 - m_{13}(\mathbf{y})_t)\log(1 - m_7(\hat{\mathbf{y}})_t)$.

3.4 Data Augmentation

Masking. To encourage the model to not only rely on local information for its predictions, we mask 0 to 6 areas of 0.5 to 2 s. Each masked area is randomly divided into 5 to 10 parts which are randomly reordered. This destroys local correspondence between audio and beats without changing local input statistics, and works better than zero masking as employed in SpecAugment [22]. We assume our approach makes it harder for the network to learn a dedicated behaviour for masked areas.

Pitch and time. We speed up and slow down every song by 20, 16, 12, 8, and 4%, and transpose by at most +6 and -5 semitones. We precompute these augmentations (using Pedalboard [23]) so experiments become reproducible without access to the original audio. To limit storage use, tempo and pitch augmentations are not combined, giving 22 variations for each song. We verified that our limited tempo augmentation gives comparable results to the commonly used approach by Böck and Davies [19] of performing on-the-fly augmentations by randomly changing the hop size of the STFT, at the advantage of not requiring audio access.

4. EXPERIMENTS

We perform 8-fold cross-validation experiments on multiple datasets, compute results on the test-only GTZAN dataset, and do an ablation study.

We use the standard beat-tracking metrics: F1, CMLt, and AMLt and compute them using the `mir_eval` package [24] with default parameters. [3] CMLt and AMLt are called continuity metrics, and only consider a beat/downbeat as correct if both it and the previous beat/downbeat are correct; AMLt also accepts different metrical levels such as half or double time, and offbeats [25]. The metrics and their settings match those used by Hung et al. [10]; this enables a comparison with their reported results, though it is unclear which 8-fold datasplit they use, and we cannot run any statistical significance tests since their code is not reproducible. Therefore, any comparison needs to be taken only as an indication.

4.1 Datasets

We train and validate with several datasets: Simac [26], SMC [27], Hainsworth [28], Ballroom [26, 29], HJDB [30], Beatles [31], Harmonix [32], RWC [33, 34] (classical, pop, royalty-free, and jazz), TapCorrect [35], JAAH [36], Filosax [37], ASAP [38], Groove MIDI [39], GuitarSet [40], Candombe [41]. The first two datasets contain only beat annotations, all others both beat and downbeats. We discard one Beatles piece which does not contain downbeat annotations and one with empty beat annotations, resulting in a total of 4556 tracks. For comparison, Hung et al. [10] train with only the first 7 datasets reported above (Simac to Harmonix), plus RWC pop, for a total of 3144 pieces (when assuming the same handling of missing annotations). We use the GTZAN [13] dataset (993 pieces discarding one unannotated track and 6 tracks that miss downbeat annotations) for testing only.

We use only the backing tracks of Filosax without the saxophone solos. For ASAP, we discard the tracks that contain the "rubato" beat annotations. In Groove MIDI, we keep all pieces that are longer than 20 seconds and use the provided audio renderings. We only use the comping tracks of GuitarSet, discarding the solos.

We employ the *8-fold cross-validation splits* published by Böck and Davies [19] for the datasets they used and produce new ones for our added datasets, ensuring different versions of the same piece are not spread across folds, and

[3] This includes "trim_beats" of 5s that discards the first 5 seconds during the evaluation, which is a choice we do not necessarily approve of, but we use it to be consistent with what seems the standard way of evaluating.

stratifying by metadata when possible. We also produce a new *single split* with $\sim 15\%$ of the pieces on each dataset as validation (again taking care of different versions of the same piece).

4.2 Training

We train for 100 epochs with gradient accumulation over 8 batches of size 8,[4] AdamW optimizer [42], weight decay of 0.01 (excluding biases and learned norms), learning rate warm-up [43] of 1000 steps to a maximum of 0.0008, and cosine annealing. During each epoch, we randomly sample 30 seconds of each piece, and pad if the total piece duration l is less than 30 seconds. We draw k samples from pieces that are longer than 30 seconds, following the equation $k = \mathit{round}(\alpha \cdot l/30)$ with $\alpha = 0.65$, since we observe it leads to faster convergence than using one random sample per piece, or $l/30$ non-overlapping samples. On average, this yields ~ 3 samples per piece. During training, every time a sample is drawn, we randomly select a precomputed augmentation described in Section 3.4, and apply masking. The full training takes around 8 hours on a single NVIDIA RTX 2080 Ti, 6 hours on A40, and 4 hours on A100.

During our experimentation, we found that to achieve good results without a DBN, we need our network to be overconfident in its predictions. This may seem to violate usual deep learning practice, but can be explained by a closer look at the beat tracker's desirable output. We do not want our network to produce probabilities close to 0.5 when unsure, since this will lead to random oscillations between positive and negative predictions, and thus erratic beats. Instead, we want it to give steady, high-probability predictions even when unsure, exactly like the DBN would. To achieve this, we keep training even after the validation loss starts increasing, which would typically indicate overfitting. Indeed, we see that the validation F1 score continues to improve even with increasing validation loss. This means that even with our modifications, the BCE loss is not a good indicator of the F1 score, and further research into alternative losses may be valuable.

The reader may wonder why, once we obtain our well-performing network, we do not use the DBN to increase the metrics even more. By having overconfident predictions, we reduce the benefits of such a postprocessing method. With a degree of simplification, we can imagine the DBN as using a model's most high-confident predictions to infer beats in low-confidence areas. By avoiding the low-confidence predictions, we are disrupting this mechanism.

4.3 Main Results

We report the results on our 18 datasets in the commonly used 8-fold cross-validation setting: each dataset is split into 8 parts, we jointly train on $18 \cdot 7$ parts (all but one per dataset) and predict on the remaining 18, after 8 such runs we covered all pieces and average metrics over pieces by dataset. We observe that our model outperforms Hung et al. [10], except for Harmonix and RWC Pop (downbeat). In our results, the lowest downbeat performance is obtained in the ASAP and RWC Classical datasets, confirming the well-known difficulty of beat-tracking classical music [2, 5]. Performance on SMC (where only beat annotations are accessible) is also very low, consistent with the outcomes of other systems, highlighting the substantial room for improvement that beat tracking systems continue to hold.

	Beat F1		Downbeat F1	
	Our	Hung	Our	Hung
ASAP	76.3	-	61.2	-
Ballroom	**97.5**	96.2	**95.3**	93.7
Beatles	**94.5**	94.3	**88.8**	87.0
Candombe	99.7	-	99.7	-
Filosax	99.5	-	98.5	-
Groove MIDI	93.7	-	82.1	-
GuitarSet	92.0	-	88.1	-
Hainsworth	**91.9**	87.7	**80.0**	74.8
Harmonix	**95.8**	95.3	90.7	**90.8**
HJDB	98.2	-	96.6	-
JAAH	95.1	-	85.0	-
RWC Classical	77.1	-	66.3	-
RWC Jazz	83.3	-	80.7	-
RWC Pop	**96.1**	95.0	93.7	**94.5**
RWC RF	94.5	-	91.9	-
Simac	77.9	-	-	-
SMC	**62.7**	60.5	-	-
TapCorrect	93.0	-	86.4	-

Table 1: Results with 8-fold cross-validation.

We also report the results on the GTZAN dataset in Table 2. We compute these results with a single model trained on the entirety of our training-val dataset (note that we do not perform any early stopping or other techniques for which the validation dataset may still be necessary). All our runs are computed 3 times with different random seeds, and we report the means and standard deviations of the computed metrics over the 3 seeds. We notice that even when training on the reduced collection of datasets by Hung et al. (third row in the table), we still outperform their F1 score without a DBN, proving the effectiveness of our design choices. Our main model has ~20 M parameters, 5 times more than Hung et al. with 4 M, so we also show the results for a smaller model with the hidden dimension of the main transformer blocks reduced from 512 to 128, and the number of heads from 16 to 4. This small model has ~2 M parameters and still gives SOTA F1 scores.

Disappointingly, we notice that the continuity metrics (CMLt and AMLt) are lower than those of Hung et al. From qualitative inspections of the results, we notice that for complex or underrepresented pieces, our network introduces non-periodic beats, which drastically lower the continuity metrics. We are then brought to wonder why our network cannot learn a supposedly obvious behaviour, such as only producing periodic-like output, and we can propose two explanations. Firstly, our loss does not specifically penalise non-periodic predictions, but treats each beat individually.

[4] This enables training with under 8 GiB of GPU memory.

	Beat			Downbeat		
	F1	CMLt	AMLt	F1	CMLt	AMLt
Hung et al. [10]	88.7	**81.2**	**92.0**	75.6	71.5	**88.1**
Our system	**89.1 $\pm$ 0.3**	79.8 $\pm$ 0.6	89.8 $\pm$ 0.4	**78.3 $\pm$ 0.4**	67.3 $\pm$ 0.8	79.1 $\pm$ 0.6
– limited to data of [10]	88.9 $\pm$ 0.1	79.9 $\pm$ 0.4	89.4 $\pm$ 0.2	75.5 $\pm$ 0.5	60.8 $\pm$ 1.2	75.5 $\pm$ 0.5
– smaller model	88.8 $\pm$ 0.2	79.4 $\pm$ 0.4	89.0 $\pm$ 0.4	77.2 $\pm$ 0.2	65.3 $\pm$ 0.3	78.0 $\pm$ 0.3
– with DBN	88.1 $\pm$ 0.3	80.5 $\pm$ 0.4	91.1 $\pm$ 0.2	77.4 $\pm$ 0.2	**73.3 $\pm$ 0.2**	87.8 $\pm$ 0.5

Table 2: Evaluation on the test dataset (GTZAN). The results for Hung et al. [10] are taken from their paper.

	Beat F1	Downbeat F1
Our system	**92.6 $\pm$ 0.1**	**85.4 $\pm$ 0.1**
No sum head	**92.6 $\pm$ 0.1**	85.0 $\pm$ 0.1
No tempo augmentation	92.5 $\pm$ 0.1	84.9 $\pm$ 0.1
No mask augmentation	92.2 $\pm$ 0.0	84.5 $\pm$ 0.3
No partial transformers	92.2 $\pm$ 0.1	83.9 $\pm$ 0.2
No shift tolerance	91.2 $\pm$ 0.2	82.2 $\pm$ 0.4
No pitch augmentation	88.3 $\pm$ 0.4	80.8 $\pm$ 0.5
No shift tol., no weights	79.5 $\pm$ 0.7	68.7 $\pm$ 0.8

Table 3: Ablation studies on the single split validation dataset, ordered by decreasing downbeat F1.

This results in a discrepancy between what is preferred by continuity metrics and what the network learns to predict in difficult parts to minimise the loss. Secondly, our datasets contain several non-periodic annotations, some due to quality issues (see Section 4.5), some in correctly annotated pieces such as tapcorrect_10 or beatles_Wild-Honey-Pie, where a 2/4 measure in the middle of a 4/4 piece disrupts the assumption of periodicity for downbeats. Finally, one could also question the generality of the AMLt metric as a tool to quantify double/half-time errors, since the computations of different metrical levels assume that the time signature and the tempo do not change and that a measure can always be divided into 2 or 3 parts.

Using a DBN increases our CMLt downbeat performance by correcting some of the (wrongly) non-periodic outputs, but it reduces our F1 performance, by changing other otherwise correct predictions that fall outside the DBN assumptions. The AMLt score does not increase since our network is overconfident in its predictions, and the DBN cannot easily switch to another metrical level.

4.4 Ablation Studies

We ablate multiple components of our model on the single split described in Section 4.1. We perform every experiment 3 times with different seeds and report the mean and standard deviation on the validation set in Table 3. The usage of our Sum Head shows little impact, but we use it to have a musically valid output, rather than to increase the F1 score. Pitch, mask, and tempo augmentations help, in this order of importance. The usage of partial transformers in our frontend proves more effective than only having convolutions. Our most impactful design choice is the weighted shift-tolerant loss. Using a normal BCE with positive example weights results in decreased performance, which decreases even further when the weights are removed.

4.5 Notes on Dataset Quality

While exploring the datasets, we found multiple problems in the annotations, and we think this hinders the development of better models, especially for downbeat predictions. Even the GTZAN dataset, which is commonly used for evaluation, is not immune to quality problems. Some of them are evident and not debatable, like jazz_00000, jazz_00002, jazz_00083, blues_00015, reggae_00095, classical_00077, rock_00067. Furthermore, there are pieces where even for experts it would be hard to agree on a unique beat and downbeat annotation, like metal_00081 or classical_00056, and multiple annotations would be necessary. Finally, some pieces question the primary assumption of beat tracking, i.e., that there is a beat/downbeat to track, like pop_00064, and jazz_00003.

5. CONCLUSIONS AND OUTLOOK

In this paper, we presented a new beat tracking system which obtained a state-of-the-art F1 score on a very diverse set of music, with minimal assumptions about the tempo, time signature, and their changes over time. Remarkably, we do not use the DBN postprocessing, which was employed by all recent high-accuracy models. However, removing the DBN hurts the CMLt and AMLt metrics. A study on how this trade-off affects human perception, alternative metrics, and a direct comparison with DBN-based models on complex pieces is left for future work.

We emphasise that beat tracking is not a solved problem, even for commonly targeted genres such as rock or electronic music, especially for the downbeat tracking task. We provide an open-sourced model that can be used as a starting point, and we invite future researchers to improve it. Potential directions are: reducing the model parameters, developing new losses that enforce periodicity during training, using other data augmentation techniques to make the system more robust to multiple sound conditions, fine-tuning it on specific genres, and training on larger datasets. The contribution of people with musical expertise will also be essential, as we think that correcting the existing commonly used datasets, and producing new annotated data for under-represented genres is a crucial step for further development.

6. ACKNOWLEDGEMENTS

This work was supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No. 101019375 (*Whither Music?*), and the Federal State of Upper Austria (LIT AI Lab). The computational results presented were achieved in part using the Vienna Scientific Cluster (VSC).

7. REFERENCES

[1] T. Cheng and M. Goto, "Transformer-based beat tracking with low-resolution encoder and high-resolution decoder," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2023.

[2] T.-P. Chen and L. Su, "Toward postprocessing-free neural networks for joint beat and downbeat estimation," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

[3] J. Zhao, G. G. Xia, and Y. Wang, "Beat transformer: Demixed beat and downbeat tracking with dilated self-attention," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

[4] T. Kim and J. Nam, "All-in-one metrical and functional structure analysis with neighborhood attentions on demixed audio," in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2023.

[5] C.-Y. Chiu, M. Müller, M. E. Davies, A. W.-Y. Su, and Y.-H. Yang, "Local periodicity-based beat tracking for expressive classical piano music," *Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 2922–2934, 2023.

[6] L. Maia, M. Rocamora, L. W. P. Biscainho, and M. Fuentes, "Adapting meter tracking models to latin american music," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2023.

[7] A. S. Pinto and G. Bernardes, "Bridging the rhythmic gap: A user-centric approach to beat tracking in challenging music signals," in *Proceedings of the International Symposium on Computer Music Interdisciplinary Research (CMMR)*, 2023.

[8] D. Desblancs, V. Lostanlen, and R. Hennequin, "Zero-note samba: Self-supervised beat tracking," *Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 2922–2934, 2023.

[9] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[10] Y.-N. Hung, J.-C. Wang, X. Song, W.-T. Lu, and M. Won, "Modeling beats and downbeats with a time-frequency transformer," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[11] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "RoFormer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, 2024.

[12] W.-T. Lu, J.-C. Wang, Q. Kong, and Y.-N. Hung, "Music source separation with band-split RoPE transformer," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024.

[13] U. Marchand and G. Peeters, "Swing ratio estimation," in *Digital Audio Effects (Dafx)*, 2015.

[14] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[15] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proceedings of the International Conference on 3D vision (3DV)*, 2016.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the International Conference on 3D vision (3DV)*, 2017.

[17] N. Abraham and N. M. Khan, "A novel focal Tversky loss function with improved attention U-Net for lesion segmentation," in *Proceedings of the International Symposium on Biomedical Imaging (ISBI)*, 2019.

[18] T. Oyama, R. Ishizuka, and K. Yoshii, "Phase-aware joint beat and downbeat estimation based on periodicity of metrical structure." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[19] S. Böck and M. E. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[20] C.-C. Chang and L. Su, "BEAST: Online joint beat and downbeat tracking based on streaming transformer," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[21] Y. Bondarenko, M. Nagel, and T. Blankevoort, "Quantizable transformers: Removing outliers by helping attention heads do nothing," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[22] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proocedings of the Interspeech Conference*, 2019.

[23] P. Sobot, "Pedalboard," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.7817838

[24] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common MIR metrics," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[25] M. E. Davies, S. Böck, and M. Fuentes, *Tempo, Beat and Downbeat Estimation*, 2021. [Online]. Available: https://tempobeatdownbeat.github.io/tutorial/intro.html

[26] F. Gouyon, "A computational approach to rhythm description — Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing," Ph.D. dissertation, Universitat Pompeu Fabra, 2006.

[27] A. Holzapfel, M. E. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon, "Selective sampling for beat tracking evaluation," *Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 9, pp. 2539–2548, 2012.

[28] S. W. Hainsworth and M. D. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, pp. 1–11, 2004.

[29] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2013.

[30] J. Hockman, M. E. Davies, and I. Fujinaga, "One in the jungle: Downbeat detection in hardcore, jungle, and drum and bass." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[31] M. E. Davies, N. Degara, and M. D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," *Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06*, 2009.

[32] O. Nieto, M. C. McCallum, M. E. Davies, A. Robertson, A. M. Stark, and E. Egozy, "The Harmonix set: Beats, downbeats, and functional segment annotations of western popular music." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[33] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical and jazz music databases," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2002.

[34] M. Goto, "AIST annotation for the RWC music database," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2006.

[35] J. Driedger, H. Schreiber, W. B. de Haas, and M. Müller, "Towards automatically correcting tapped beat annotations for music recordings." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[36] V. Eremenko, E. Demirel, B. Bozkurt, and X. Serra, "Audio-aligned jazz harmony dataset for automatic chord transcription and corpus-based research," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[37] D. Foster and S. Dixon, "Filosax: A dataset of annotated jazz saxophone recordings," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[38] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: a dataset of aligned scores and performances for piano transcription," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[39] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman, "Learning to groove with inverse sequence transformations," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[40] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "GuitarSet: A dataset for guitar transcription." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[41] M. Rocamora, L. Jure, B. Marenco, M. Fuentes, F. Lanzaro, and A. Gómez, "An audio-visual database of Candombe performances for computational musicological studies," in *Congreso Internacional de Ciencia y Tecnología Musical (CICTeM)*, 2015.

[42] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[43] X. S. Huang, F. Perez, J. Ba, and M. Volkovs, "Improving transformer optimization through better initialization," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.