# Zero-Note Samba: Self-Supervised Beat Tracking

Dorian Desblancs, Vincent Lostanlen, Romain Hennequin

▶ **To cite this version:**

Dorian Desblancs, Vincent Lostanlen, Romain Hennequin. Zero-Note Samba: Self-Supervised Beat Tracking. IEEE/ACM Transactions on Audio, Speech and Language Processing, 2023, pp.1-13. 10.1109/TASLP.2023.3297963 . hal-03669865v2

**HAL Id: hal-03669865**
**https://hal.science/hal-03669865v2**

Submitted on 22 Jul 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

# Zero-Note Samba: Self-Supervised Beat Tracking

Dorian Desblancs, Vincent Lostanlen, and Romain Hennequin

***Abstract*—Supervised machine learning for music information retrieval requires a large annotated training set, and is thus an expensive and time-consuming process. To circumvent this problem, we propose to train deep neural networks to perceive beats in musical recordings despite having little or no access to human annotations. The key idea is to train two fully convolutional networks in parallel, which we name "Zero-Note Samba" (ZeroNS): the first analyzes the percussive part of a musical piece whilst the second analyzes its non-percussive part. These networks learn a self-supervised pretext task of synchrony prediction (sync-pred), which simulates the ability of musicians to groove together when playing in the same band. Sync-pred encourages the two networks to return similar outputs if the underlying musical parts are synchronized, yet dissimilar outputs if the parts are out of sync. In practice, we obtain the instrumental parts from commercial recordings via an off-the-shelf source separation system: Spleeter. After self-supervised learning with sync-pred, ZeroNS produces a sparse output that resembles a beat detection function. When used in conjunction with a dynamic Bayesian network, ZeroNS surpasses the state of the art in unsupervised beat tracking. Furthermore, fine-tuning ZeroNS to a small set of labeled data (of the order of one to ten songs) matches the performance of a fully supervised network on 96 songs. Lastly, we show that pre-training a supervised model with sync-pred mitigates dataset bias and thus improves cross-dataset generalization, at no extra annotation cost.**

***Index Terms*—Blind source separation, Multi-layer neural network, Music information retrieval, Unsupervised learning.**

## I. Introduction

HUMAN listeners have an intuitive understanding of rhythm [1]. When exposed to music, our auditory system seeks to produce an internal representation of the current "pace" *(tempo)* of musical events such as notes. This internal representation, known as *meter*, aids coordination between musicians in the same orchestra as well as with non-audible gestures, such as dancing or marching [2]. Crucially, musical meter is most often periodic or varies predictably, even though the underlying auditory stimulus is never twice the same. Thus, meter may be encoded efficiently by defining a *pulse*; that is, a sequence of sparse and evenly spaced activations [3].

### A. Beat Induction in Humans versus Machines

Meter is not reducible to a single pulse frequency but, rather, follows a hierarchical division into subsets [4]. In this regard, the most salient level of pulse is called tactus or *beat*: its rate typically ranges between 60 and 180 beats per minute (BPM) and matches the clapping of one's hands or the stomping of one's foot [5]. Admittedly, several musicians may judge the beat rate to belong to unequal levels of the pulse hierarchy: for example, one might tap twice or three times faster than the other [6]. Yet, both will internalize musical meter in a comparable way, and thus will remain able to play together. Indeed, they will notice if one's musical part ever falls "ahead" or "behind" the collective pulse.

This cognitive ability for mapping sound to coordinated gestures is widespread, as it does not require musical expertise. Recent findings have shown that beat induction is active in non-musician adults and even in sleeping newborns [7]. Besides humans, the spontaneous sensorimotor synchronization to a musical beat has been observed in some animals; e.g., parakeets [8] and a California sea lion [9].

The situation is different in machine listening. There is a long-standing effort towards developing audio processing systems which analyze a musical stream so as to predict the most probable beat sequence a human listener would perceive—a task known as *beat tracking* [10, chapter 6]. As of today, all state-of-the-art (SOTA) methods for beat tracking in music information retrieval (MIR) rely on artificial neural networks: either convolutional (CNN) [11], recurrent (RNN) [12], or both (CRNN) [13]. These deep learning models consist of nonlinear units with linear connections, and hence share some structural similarities with neurons and synapses in the brain [14].

However, the analogy only goes so far: while living organisms learn by direct interaction with the real world, deep learning systems require a long preliminary stage of supervised training before deployment [15]. The case of beat tracking is exemplary in that regard: in recent studies, the training set typically consists of $10^2 \sim 10^3$ songs, each containing $10^2 \sim 10^3$ humanly annotated timestamps [16]. Collecting these timestamps requires a cognitive effort that is costly, time-consuming, and disconnected from musical practice as such.

### B. Problem Statement

Our article addresses the problem of training a deep neural network for beat tracking with little or no access to annotated audio data. This problem is interesting for at least three reasons. First, from the standpoints of artificial intelligence and music cognition, it reflects the learning process underlying beat induction in humans better than the current task design, which is fully supervised. Second, from the standpoint of MIR, solving this problem raises the opportunity to scale up to massive online music corpora: i.e., $10^6 \sim 10^7$ songs [17]. Third, from the standpoint of digital (ethno)musicology, unsupervised learning might help to "precondition" the beat tracking system towards accommodating the rhythmic peculiarities of a given genre, even in the absence of human annotation. This third motivation

D. Desblancs and R. Hennequin are with Deezer Research, Paris, 75009, France. V. Lostanlen is with Laboratoire des Sciences du Numérique de Nantes (LS2N) at the Centre National de la Recherche Scientifique (CNRS), Nantes, 44300, France.

The source code to reproduce figure and experiments is available at: https://www.github.com/deezer/zeroNoteSamba.

connects with the overarching goal of diversifying MIR beyond its historical scope of applicability [18].

What makes the problem of unsupervised beat tracking difficult resides in the design of an adequate prior for numerical optimization. Previous publications have formulated this prior in terms of internal properties of the audio stream such as periodicity [19], harmonic homogeneity [20], and the presence of strong percussive onsets at beat locations [21]. Although these properties are indeed verified in certain genres (e.g., electronic dance music), we note that beat induction may also happen without them [22]. Hence, a new methodological perspective is needed to meet the challenge of unsupervised beat tracking—especially in audio streams that are aperiodic, deprived of tonal harmonic progressions, or highly syncopated.

### C. Key Idea: Self-Supervised Synchrony Prediction

In this article, we introduce a method for training deep neural networks on the beat tracking task with little or no human effort in terms of audio annotation. Our key idea is to separate the audio stream into two parts: percussive sources (e.g., drums) and non-percussive sources (e.g., voice or wind instruments). Then, we rely upon a simple musical observation: regardless of the rhythmic pattern in the parts they play, musicians listen to each other so as to seek synchrony. This observation suggests that, in the presence of multiple instruments, beat induction arises not only from listening to the orchestra in full but also from listening to isolated parts. Hence, in order to mimic human listening, the machine must return the same beat sequence for both the percussive and non-percussive parts of a piece. We encourage alignment between instrument-specific beat patterns via a new "pretext" task for self-supervised learning (SSL), named *synchrony prediction* or sync-pred, which is formulated as a contrastive learning task.

### D. Scaling Up the Pretext Task with Source Separation

Training a self-supervised beat tracker via sync-pred is conceptually simple; yet, it would probably have been unfeasible just five years ago. Indeed, the formulation of the sync-pred task requires a massive dataset of real-world music in multitrack format, in which percussive and non-percussive parts are mapped to separate tracks. On the contrary, recorded music is most often distributed in stereo format, with all instruments being mixed. Although the MIR community has released multitrack datasets (e.g., The Open Multitrack Testbed [23], MedleyDB [24]), their size is only in the order of $10^2$ songs; i.e., they are too small to justify self-supervised learning.

Fortunately, the recent progress of (supervised) deep learning for audio source separation now allows researchers to retrieve the vocal and instrumental parts of stereo mixtures. Of course, the process of source separation is imperfect and occasionally presents audible artifacts. However, we postulate that these artifacts make the task of synchrony prediction between percussive and non-percussive parts neither unsolvable nor trivial. We propose to run an off-the-shelf source separation system (namely, Spleeter [25]) as a pre-processing step to beat tracking, both during training and deployment.

Some recent publications have done so in a supervised context, with various applications: sound event detection in domestic environments [26] and verification of vocal note-event annotations in polyphonic music [27]. However, to the best of our knowledge, only one publication so far proposes to combine source separation with contrastive learning [28]. A noteworthy difference is that [28] proposes a task of coincidence prediction, which operates as a "slowness prior"; whereas, on the contrary, our task of sync-pred operates as a "transientness prior" for the learned beat detection function.

### E. Contributions

Our paper proposes the first self-supervised approach to beat tracking in MIR. Its main originality consists in learning not one but *two* representations for every audio excerpt: one for its percussive part and another for its non-percussive part. These representations, hereafter called *embeddings*, result from two encoders which have the same architecture but independent synaptic weights. The hypothesis of our article is that training these encoders via synchrony prediction (sync-pred, see Section I-C) reinforces their sensitivity to the musical pulse.

We test this hypothesis in practice by introducing a new deep learning model named "Zero-Note Samba"[1], or ZeroNS for short. Figure 1 presents the functional diagram of ZeroNS.

Our experiments lead to six new insights:

1) Our network learns appropriately on the sync-pred task: after convergence, ZeroNS is able to predict synchrony between percussive and non-percussive parts beyond its training set. Musical recordings on which the network is unable to learn either contain no drums or contain nothing but drums; these cases are easily detected in advance. See Section IV-A.
2) Thanks to sync-pred, ZeroNS embeddings tend to be sparse and periodic in the time domain. This finding suggests the emergence of a rudimentary form of beat induction: see Section IV-B.
3) ZeroNS matches or surpasses previous SOTA unsupervised methods once coupled with the peak picker found in [29] or the SOTA dynamic Bayesian network (DBN) for peak-picking, found in [30]: see Section IV-C.
4) Sync-pred can serve as self-supervised pre-training before supervised learning. In a $k$-fold cross-validation setting, this pre-training has little effect on downstream beat tracking performance, and the effect is inconsistent (beneficial or detrimental) across datasets: see Section IV-D.
5) Sync-pred is consistently beneficial under the "low-data" regime; that is, with limited labeled data. Our results demonstrate that a fully supervised model with random initialization requires 32–96 songs to learn beat tracking, while ZeroNS reaches competitive performance after supervised fine-tuning on 1–24 songs: see Section IV-E.

[1]We choose the name "Zero-Note Samba" in homage to the 1960 song "One Note Samba" ("Samba De Uma Nota Só"), with music by Antonio Carlos Jobim, Portuguese lyrics by Newton Mendoça, and English lyrics by Jon Hendricks. The name should not be taken too literally: of course, ZeroNS is unable to learn in complete silence. A more accurate name, albeit a far less memorable one, would have been: Zero-*Annotation* Samba.

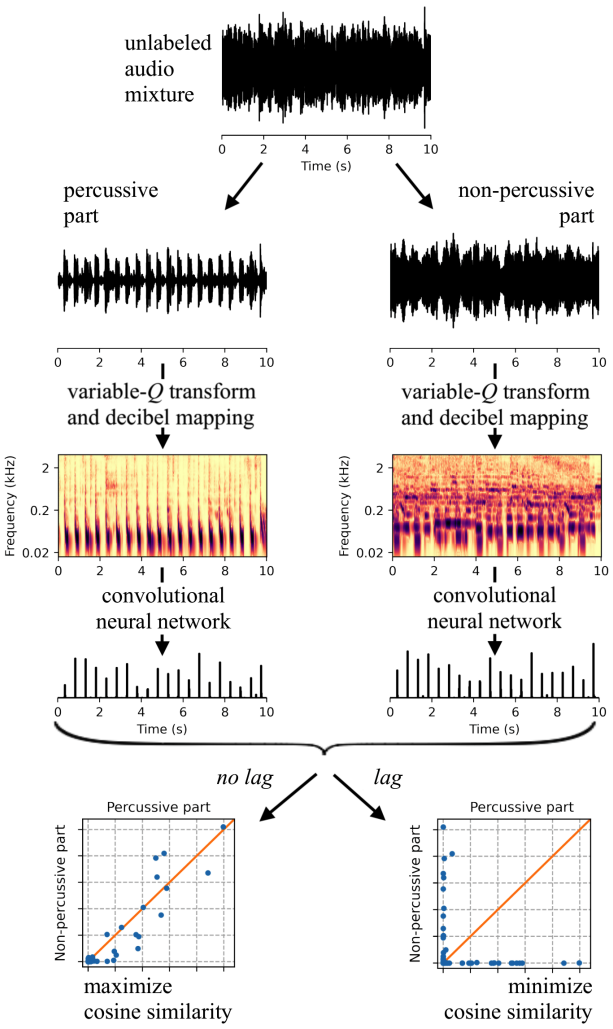



Fig. 1: Functional diagram of ZeroNS. The waveform on the top is an unlabeled music recording from the FMA dataset. Darker colors in the time–frequency representation reflect higher values of energy. The last row displays scatter plots between the percussive (x-axis) and non-percussive (y-axis) embedding. The task of synchrony prediction consists in maximizing cosine similarity in the synchronized case (left), i.e., bringing the scattered dots close to the orange diagonal line; and minimizing cosine similarity in the non-synchronized case (right).

6) Pre-training with sync-pred improves generalization under a cross-dataset evaluation setting: see Section IV-F.

## II. Related Work

In this section, we present a historical overview of four research topics: unsupervised beat tracking; feature learning in the time–frequency domain; self-supervised MIR; and learning from multitrack audio data. Although they have long been studied in machine learning and signal processing, the novelty of our work is to address them in conjunction.

### A. Unsupervised Beat Tracking

The earliest approaches to beat tracking were actually unsupervised: they did not rely on any annotated data and only had a few hyperparameters, which the authors adjusted *ad hoc* [31]. These approaches assumed the timestamps of note onsets to be available in symbolic format (e.g., MIDI) or via motion capture. Besides, they were only applied to a few melodic sequences that usually came from a single instrument [32]. Although they enhanced the real-time interaction between musician and computer [33], these methods were unsuited for the analysis of polyphonic audio. They did however lay a foundation for the computational modeling of musical rhythm.

An important milestone in the history of beat tracking resides in the development of dedicated connectionist models [34]. These models aimed to implement "entrainment"; that is, the phase-locking and frequency-locking of artificial neurons to periodic components of incoming rhythmic patterns. This line of research culminated with the adaptive-frequency neural network (AFNN), a nonlinear model in which the tempo may vary according to a Hebbian learning rule [35].

### B. From Feature Engineering to Feature Learning

Around the year 2000, MIR sprung as an autonomous field of research and beat tracking emerged as one of its well-established tasks [36]. At the time, the SOTA method [37] involved non-trainable modules only: spectral flux, median filtering with half-wave rectification, windowed autocorrelation, and cross-correlation with an artificial pulse train at the predicted tempo. By design, this approach returns a beat sequence that is globally periodic. As such, it is unfit for analyzing musical sections with time-varying tempo.

In 2007, the assumption of global periodicity was relaxed by [29], who proposed a dynamic programming formulation for beat tracking. The key idea is to build a transition cost function between candidate beats which penalizes deviations from the globally estimated tempo. In this way, the dynamic programming algorithm seeks a compromise between picking timestamps of high onset strength and ensuring that all inter-beat intervals match the tempo prior approximately. Thus, assuming that the initial tempo estimate is correct, the method in [29] can accommodate a small "stretch" in the pulse grid, as induced by expressive performance. Despite its limited accuracy, this method remains widely used today because it is conceptually simple. Moreover, it is implemented by librosa [38], a free and open-source software library for Python, as:

```
tempo, beats = librosa.beat.beat_track(
    y, sr=16000, hop_length=256, start_bpm=120)
```

In this article, we use the implementation above as a point of comparison with our proposed method. We regard [29] as the current state of the art in *unsupervised* beat tracking. All algorithmic parameters were tuned to our audios' specifications, such as sample rate, to maximize performance.

The emergence of deep learning in MIR overturned the formulation of the beat tracking task. Rather than predict the tempo globally and adjust beat times accordingly, [39] proposed to train a machine learning system that detects beat times directly from audio data. Specifically, they trained a long short-term memory network (LSTM) on a multiscale variant of the short-term Fourier transform, involving per-channel temporal differentiation and rectified median filtering.

The LSTM produces a beat detection function (BDF); that is, a univariate signal (sampled at 100 Hz) which represents the degree of confidence in the presence of a beat. In [39], the LSTM optimizes its BDF by fitting it to a "ground truth" sequence in which values of 1 and 0 respectively correspond to "beat" and "no beat" events. This is a form of *supervised* beat tracking, because such a ground truth must be provided by a human annotator.

One may improve the accuracy of beat tracking by interfacing the BDF with a *bar-pointer* model; that is, "a probabilistic model of temporal structure in music which allows joint inference of tempo, meter and rhythmic pattern" [40]. The bar-pointer model was initially proposed as a Gaussian process and later revisited as a neural network, under the name of *Dynamic Bayesian Network* or DBN [41]. This model is comprised of: an observation model, which converts the BDF into probabilities; a transition model between hidden variables of tempo, beat, and meter; and an initial distribution for these variables. Crucially, the DBN is unsupervised: all probability distributions are estimated from unlabeled BDF observations and some domain-specific knowledge about music, such as BPM range and the number of states per second. In this paper, we post-process our self-supervised BDF with the most recent DBN provided in [30], as provided by the Python package "madmom" [42]. In [43], the authors (Davies and Böck) use the same DBN in conjunction with their supervised temporal convolutional neural network (CNN). We consider their method as the current state of the art in *supervised* beat tracking. Note that updated beat tracking results can be found in [44] and involve joint beat and downbeat estimation using a transformer architecture.

### C. Self-Supervised Learning in Music Information Retrieval

With a number of publications doubling every year since 2015, the field of self-supervised learning is expanding rapidly [45]. Thus, a comprehensive review of all pretext tasks for self-supervised learning in audio processing is beyond the scope of this paper. Instead, we limit our review to a few publications which play a key role in the inspiration of ZeroNS.

Broadly speaking, we may define SSL as a paradigm in which the machine learns to accomplish a task whose ground truth is trivially available. This task, known as "pretext", does not necessarily have any practical interest per se; however, its resolution supposedly requires some intelligent auditory processing on the part of the machine. Hence, SSL aims at a "best-of-both-worlds" approach between supervised and unsupervised learning. On one hand, SSL is unsupervised in the sense that it operates without human annotation: one may collect real-world unlabeled data and alter them at random to produce a ground truth. On the other hand, since SSL is typically formulated as classification or regression, it can be implemented like supervised learning; namely, as empirical risk minimization with stochastic gradient descent.

A simple but powerful family of pretext tasks for SSL is found in *contrastive learning* [46]. Generally, contrastive learning operates over "batches" of size $B$, each of them containing three kinds of data: one anchor, one positive, and several negatives. Intuitively, the pretext task consists in matching the positive with the anchor while ruling out all negatives. In practice, this is achieved in a differentiable way by contrasting anchor–positive similarity versus anchor–negative similarities. The main difference between contrastive learning tasks resides in their definitions of what constitutes "positive" and "negative" samples with respect to the anchor. In [47], the anchor and positive (resp. negative) are drawn from the same (resp. from different) musical recordings: this task is called similarity maximization. In [48], the authors perform random digital audio effects (e.g., pitch shifting, reverb, equalization) to all samples before similarity maximization, thus encouraging self-supervised invariance to these perturbations.

Besides contrastive learning, our work is inspired by self-supervised pitch estimation (SPICE) [49]. The key idea behind SPICE is to train a convolutional autoencoder in the time–frequency domain to discover relative pitch. During training, two copies of this autoencoder analyze artificially transposed versions of a same musical recording. The pretext task consists in regressing the amount of artificial pitch shift between both versions. After limited supervision, relative pitch is mapped to absolute pitch; the embedding of SPICE becomes a fundamental frequency estimator with almost SOTA performance.

### D. Learning from Multimodal and Multitrack Data

Recent publications have extended SSL to multimodal data, such as videos with audio [50]. Under a contrastive learning formulation, the anchor is a video frame; the positive sample, the matching audio excerpt; and the negative sample, a random audio excerpt from a different music video. This method assumes *audiovisual correspondence*; i.e., that whichever instruments appear in the frame match those being heard. In the absence of a side-channel modality such as video, it remains possible to exploit the *coincidence* between multiple sources within the same modality. In [28], the authors propose to apply a pre-trained model for "universal sound separation" [51] to unlabeled polyphonic mixtures; and to train an encoder to predict the coincidence between a separated source and the mixture to which it belongs, as a binary classification problem.

In the realm of MIR, the authors of [52] create musical mash-ups using instrumental and vocal stems. Then, they train a network to distinguish singers from each other using triplet learning, each mashup being used to strengthen the network's ability to recognize vocals within a song. This work is enhanced by Spleeter in [53]. By extracting vocals in an automatic fashion, the triplet learning task can be conducted on more data, and thus extract higher-performing vocal representations for the vocal-related downstream tasks.

Finally, in the realm of source separation in beat tracking, the following two works inspired this paper. In [54], the authors leverage source separation in the same way as us: they create percussive and non-percussive tracks for each song. From there, three beat tracking systems are trained on three different types of audio: mixtures, percussive tracks, and non-percussive tracks. Their outputs are then fused to enhance performance on the beat tracking and downbeat estimation tasks. In [55], percussive and non-percussive tracks are used to augment the training set and improve the network's performance for the same tasks.

## III. METHODS

In this section, we present: the automatic curation of our unlabeled training set for self-supervised learning; our choice of time–frequency representation; the fully convolutional architecture of the model, named "Zero-Note Samba"; and beat tracking, our supervised downstream task.

### A. Automatic Curation

As noted in Section I-C, we train ZeroNS by predicting the synchrony between percussive and non-percussive parts in the same song. To this end, we need to curate a large-scale training set in which every song contains both percussive instruments and non-percussive instruments. Yet, in a digital music archive, some songs may contain no percussive parts (e.g., *a cappella* choir) while others may contain only percussive parts (e.g., drum ensemble). Since sync-pred would become ill-posed on those songs, we exclude them from our training set beforehand.

In practice, we use music data from the Free Music Archive. The FMA is one of the largest audio catalogs of royalty-free music, and was dumped to an immutable web repository in 2017 [56]. This dump, known as "FMA dataset", has frequently been used in MIR research for audio tagging; e.g., [57]. Yet, our publication is the first to use the dataset for beat tracking. We download the "large" version of the FMA dataset[2]. This version contains 107k song excerpts, each of them lasting 30 seconds, and spans a 161, mostly Western, genre hierarchy.

In order to verify the presence of percussive and non-percussive sources, we process each of these song excerpts with Spleeter[3]. Spleeter is a Python library for audio source separation, involving several pre-trained deep learning models. Spleeter is a supervised model because it was trained with separated sources as a ground truth. However, this ground truth is not equivalent to a human annotation of musical beats. Thus, we may still regard our pipeline as unsupervised with respect to the beat tracking task. Among the models that are present in Spleeter, we select `4stems`: i.e., a four-way separation between *drums*, *vocals*, *bass*, and *other*. For *drums*, the source-to-interference ratio (SIR) [58] of Spleeter is of the order of 12 dB on the MUSDB18 test set [59]; i.e., within one decibel of the state of the art [25]. In other words, the energy of the *drums* track as extracted by Spleeter consists of $(10^{1.2})/(1+10^{1.2}) \approx 94\%$ actual drums and about 6% other sources, on average. Hence, we judge that the *drums* track is sufficiently decorrelated from other tracks to make the sync-pred task nontrivial.

Given an unlabeled song excerpt $\boldsymbol{x}$, let us denote by $\boldsymbol{\phi}_p\boldsymbol{x}$ the *drums* track returned by Spleeter; hereafter called "the percussive part" of $\boldsymbol{x}$. Similarly, we denote by $\boldsymbol{\phi}_{\neg p}\boldsymbol{x}$ the mixture of *vocals*, *bass* and *other* tracks returned by Spleeter; hereafter called "the non-percussive part" of $\boldsymbol{x}$.

We compute the root-mean-square (RMS) value of both $\boldsymbol{\phi}_p\boldsymbol{x}$ and $\boldsymbol{\phi}_{\neg p}\boldsymbol{x}$ over non-overlapping rectangular windows of duration $46\,\mathrm{ms}$; i.e., 2048 samples at a rate of $44.1\,\mathrm{kHz}$. Then, we define a temporal criterion of simultaneous presence of percussive and non-percussive sources via a double inequality on the ratio between $\mathrm{RMS}(\boldsymbol{\phi}_p\boldsymbol{x})$ and $\mathrm{RMS}(\boldsymbol{\phi}_{\neg p}\boldsymbol{x})$:

$$\mathrm{Criterion}(\boldsymbol{x})(t) = \mathbb{1}\left(\frac{1}{2} < \frac{\mathrm{RMS}(\boldsymbol{\phi}_p\boldsymbol{x})(t)}{\mathrm{RMS}(\boldsymbol{\phi}_{\neg p}\boldsymbol{x})(t)} < 4\right), \quad (1)$$

where the symbol $\mathbb{1}$ represents the indicator function. The lower and upper bound ($\frac{1}{2}$ and 4) in the equation above are chosen ad hoc after a process of trial and error. If the above is satisfied for over 30% of the RMS frames $t$, we extract a 10-second musical clip within the 30-second excerpt. This 10 second duration was selected due to computational constraints, in order to later scale our pretext task to a batch size of 16 (see Section III-D). In this way, we extract 35,200 pairs $(\boldsymbol{\phi}_p\boldsymbol{x}_i, \boldsymbol{\phi}_{\neg p}\boldsymbol{x}_i)_i$ from the `fma_large` dataset; i.e., around 98 hours of audio.

### B. Time–Frequency Representation

Musical onsets are typically easier to detect in the time–frequency domain than in the time domain [60]. Indeed, musical notes exhibit a sharp pattern of amplitude modulation at their onset [61]. For this reason, nearly all deep learning systems for beat tracking rely on a time–frequency representation, such as mel-frequency spectrogram [39], chromagram [16], or constant-$Q$ transform (CQT) [11].

In this paper, we use a variable-$Q$ transform (VQT); i.e., a variant of the CQT in which the time resolution is improved in the low-frequency range [62]. This is especially important for our embeddings, because crucial information about musical pulse arises below 100 Hz. That being said, we note that sync-pred has a general definition and could, in principle, apply to any feature map that is approximately equivariant with time shifts; not solely the VQT but other time–frequency representations as well, or even the raw waveform $\boldsymbol{x}$.

To compute the VQT, we design band-pass filters $\boldsymbol{\psi}_\lambda$ indexed by a log-frequency variable $\lambda$ in octaves which ranges between 0 and $(J-1)$. Specifically, we design each $\boldsymbol{\psi}_\lambda$ with a center frequency of $2^\lambda\xi$ and an effective bandwidth of $B(\lambda) = \gamma + 2^\lambda\xi/Q$ where $\gamma \geq 0$ is a constant offset in Hertz, $\xi$ is the center frequency of the bottom-most filter ($\lambda = 0$) in Hertz, and $Q$ is a dimensionless number. We use the VQT implementation of Librosa v0.8.0 [38]. In this implementation, $\gamma$ is adjusted via domain-specific knowledge about psychoacoustics. The bandwidths $B(\lambda)$ are made proportional to the equivalent rectangular bandwidths (ERB) of the human cochlea [63]: $\mathrm{ERB}(\lambda) = 2^\lambda\xi\alpha + \beta$ where $\alpha = 0.108$ and $\beta = 24.7$ Hz. Solving $B(\lambda) \propto \mathrm{ERB}(\lambda)$ with $Q = 12$ yields $\gamma = \beta/(\alpha Q) = 19.1$ Hz.

Once the filterbank $(\boldsymbol{\psi}_\lambda)_\lambda$ is built, we compute the VQT of an audio signal $\boldsymbol{x}$ as its convolutions with every $\boldsymbol{\psi}_\lambda$ followed by the application of pointwise complex modulus and pointwise logarithm, with a constant numerical offset $\varepsilon$:

$$\mathbf{U}\boldsymbol{x}(t,\lambda) = \log\left(\varepsilon + \left|\boldsymbol{x} * \boldsymbol{\psi}_\lambda\right|(t)\right). \quad (2)$$

Within a discrete-time setting, we take $Q = 12$ evenly spaced values of $\lambda$ per octave. In this paper, we set the center frequency of the bottom-most filter to $\xi = 16.35$ Hz and the number of octaves to $J = 8$; hence $QJ = 96$ values of $\lambda$ in total, up to $2^8\xi = 4185.6$ Hz. We set the hop length of the VQT to $16\,\mathrm{ms}$; i.e., 256 samples at $16\,\mathrm{kHz}$. We set $\varepsilon = 10^{-10}$.

[2]FMA repository link: https://github.com/mdeff/fma

[3]Source code of Spleeter: https://github.com/deezer/spleeter

### C. Fully Convolutional Neural Network

The ZeroNS model comprises two branches which analyze separate musical parts. Let us denote by $f_p$ (resp. $f_{\neg p}$) the branch which operates on the percussive (resp. non-percussive) part; i.e., $\phi_p \boldsymbol{x}$ (resp. $\phi_{\neg p}\boldsymbol{x}$), see Section III-A. We build $f_p$ and $f_{\neg p}$ as *fully convolutional* neural networks. This design choice is inspired by the state of the art in supervised beat tracking [43]. Although $f_p$ and $f_{\neg p}$ have identical architectures, their synaptic weights are not shared.

Each branch contains five convolutional layers. The "channel" dimension grows up to 256 in the third layer before returning to 1 in the last layer. The first three layers are followed by maximum pooling over the log-frequency dimension $\lambda$. In this way, they operate at multiple scales: one, three, and twelve musical semitones respectively. However, there is no pooling on the time dimension. Thus, the hop size of the output of $f_p$ is the same as that of $\mathbf{U}$; i.e., $16\,\mathrm{ms}$. The output length is also the same as the size of the input's VQT time dimension. Again, this choice of time resolution is of the same order as the state of the art in supervised beat tracking [43]. Each convolutional layer is followed by a ReLU activation function and dropout layer with value $0.1$. The last convolutional layer passes through a sigmoid activation.

The *fully convolutional* nature of our model makes it approximately equivariant to translation [64]. Once the boundary effects have been neglected, shifting the input of $f_p$ by some time lag $\tau$ is tantamount to shifting its output by $\tau$; and likewise for $f_{\neg p}$. Let us denote by $L_\tau$ the time shift operator with lag parameter $\tau$. We extend this definition to two dimensions: for every matrix $\boldsymbol{u}$, $L_\tau \boldsymbol{u}(t,\lambda) = \boldsymbol{u}(t-\tau,\lambda)$. We state the equivariance of $f_p$ in terms of commutation with $L_\tau$: for all $t$ whose distance is at least $(N+\tau)$ from both signal boundaries,

$$f_p(L_\tau \boldsymbol{u})(t) = L_\tau f_p(\boldsymbol{u})(t) = f_p(\boldsymbol{u})(t-\tau). \tag{3}$$

Now, we may observe that $\boldsymbol{u}$ is the output of a VQT, which is itself a convolutional operator followed by an operation of pointwise complex modulus. This operation demodulates oscillations and thus produces a form of approximate equivariance to time shifts, even despite setting the hop length to 256 samples (see Section III-B):

$$\mathbf{U} L_\tau \boldsymbol{x}(t,\lambda) \approx L_\tau \mathbf{U}\boldsymbol{x}(t,\lambda) = \mathbf{U}\boldsymbol{x}(t-\tau,\lambda). \tag{4}$$

Lastly, the operator $\phi_p$ proceeds from the Spleeter U-Net [65] which, again, satisfies an approximate property of equivariance:

$$\phi_p L_\tau \boldsymbol{x}(t) \approx L_\tau \phi_p \boldsymbol{x}(t) = \phi_p \boldsymbol{x}(t-\tau). \tag{5}$$

A mathematical idealization of the three equations above may be expressed in terms of the commutative diagram below:

$$\begin{array}{ccccccc} \boldsymbol{x} & \xrightarrow{\phi_p} & \phi_p \boldsymbol{x} & \xrightarrow{\mathbf{U}} & \mathbf{U}\phi_p \boldsymbol{x} & \xrightarrow{f_p} & f_p \mathbf{U}\phi_p \boldsymbol{x} \\ \downarrow{\scriptstyle L_\tau} & & \downarrow{\scriptstyle L_\tau} & & \downarrow{\scriptstyle L_\tau} & & \downarrow{\scriptstyle L_\tau} \\ L_\tau \boldsymbol{x} & \xrightarrow{\phi_p} & \phi_p L_\tau \boldsymbol{x} & \xrightarrow{\mathbf{U}} & \mathbf{U}\phi_p L_\tau \boldsymbol{x} & \xrightarrow{f_p} & f_p \mathbf{U}\phi_p L_\tau \boldsymbol{x}, \end{array} \tag{6}$$

and likewise with $\phi_{\neg p}$ and $f_{\neg p}$.

Receptive fields grow over the temporal dimension from 11 frames at the first layer to 25 at the last layer. Over the frequency dimension, their size is as large as 9 rows (27 musical semitones) in the fourth layer. As a result, ZeroNS has a relatively large number of trainable parameters: 13.4M, compared to 21.8k in [43]. This is, in part, because [43] resorts to dilated convolutions so as to increase receptive field size while maintaining a small parameter budget. We leave as future work the optimized design of receptive fields in ZeroNS.

### D. Pretext Task: Synchrony Prediction (Sync-Pred)

We train the ZeroNS model in a self-supervised way to predict the synchrony between percussive and non-percussive parts in a ten-second signal $\boldsymbol{x}$ of polyphonic music. We begin by drawing a "batch" of $B = 16$ independent time lags $\tau_1 \dots \tau_B$ uniformly at random between zero and $\Delta = 5$ seconds. For every lag $\tau_i$ with $1 \leq i \leq B$, we extract the five-second clip in $\boldsymbol{x}$ starting at time $\tau_i$, which we denote by $L_{\tau_i}\boldsymbol{x}$. Then, we apply source separation with $\phi_p$ and $\phi_{\neg p}$ followed by VQT with $\mathbf{U}$. We feed the batch $\mathbf{U}\phi_p L_{\tau_i}\boldsymbol{x}$ to the "percussive network" of ZeroNS ($f_p$) and the batch $\mathbf{U}\phi_{\neg p} L_{\tau_i}\boldsymbol{x}$ to its "non-percussive network" ($f_{\neg p}$). We apply $\ell^2$ normalization over the time dimension; let us denote by $\boldsymbol{y}_{i,p}$ and $\boldsymbol{y}_{i,\neg p}$ the resulting embeddings. Given $i$, the diagram in Equation 6 allows to interpret $\boldsymbol{y}_{i,p}$ as a local renormalization of $f_p(\mathbf{U}\phi_p\boldsymbol{x})$ over the interval $[\tau_i, \tau_i + \Delta]$:

$$\boldsymbol{y}_{i,p}(t) = \frac{f_p\big(\mathbf{U}\phi_p L_{\tau_i}\boldsymbol{x}\big)(t)}{\Big\| f_p\big(\mathbf{U}\phi_p L_{\tau_i}\boldsymbol{x}\big)\Big\|_2} \tag{7}$$

and likewise with every $\boldsymbol{y}_{i,\neg p}$ after having replaced $f_p$ and $\phi_p$ by $f_{\neg p}$ and $\phi_{\neg p}$ respectively. Intuitively, sync-pred consists in enhancing the cosine similarity between $\boldsymbol{y}_{i,\neg p}$ and $\boldsymbol{y}_{i,p}$ while inhibiting that between $\boldsymbol{y}_{i,\neg p}$ and $\boldsymbol{y}_{j,p}$ for $j \neq i$.

Let us adopt the shorthand $f$ for the tuple $(f_p, f_{\neg p})$. We propose to formulate sync-pred as the minimization of the following loss function, named normalized temperature-scaled cross-entropy or NT-Xent [46] and used in [48][47]:

$$\mathcal{L}_f(\boldsymbol{x}) = -\frac{1}{B}\sum_{i=1}^{B} \log\left(\frac{\exp(\frac{1}{T}\langle \boldsymbol{y}_{i,\neg p} | \boldsymbol{y}_{i,p}\rangle)}{\sum_{j=1}^{B}\exp(\frac{1}{T}\langle \boldsymbol{y}_{i,\neg p} | \boldsymbol{y}_{j,p}\rangle)}\right), \tag{8}$$

on average over unlabeled signals $\boldsymbol{x}$. In the equation above, the bracket notation $\langle \boldsymbol{y}_{i,\neg p} | \boldsymbol{y}_{j,p}\rangle$ denotes the scalar product between $\boldsymbol{y}_{i,\neg p}$ and $\boldsymbol{y}_{j,p}$, which corresponds to a cosine similarity between the outputs of $f_{\neg p}$ and $f_p$ after lagging by $\tau_i$ and $\tau_j$ and trimming to $\Delta$. The constant $T$ is a "temperature" hyperparameter, which we set equal to $0.25$.

In practice, we train the ZeroNS model $f$ in PyTorch[4] with the Adam optimizer [66] and a learning rate of $10^{-6}$. We split our unlabeled dataset (see Section III-A) into a training subset of size 28,800 and a validation subset of size 6,400. We monitor the validation loss and stop training once it reaches a minimum; i.e., after 35 epochs.

### E. Downstream Task: Beat Tracking

We evaluate a total of seven systems for the beat tracking task. The first four of these are unsupervised, or self-supervised, and correspond to:

[4]Official website: https://pytorch.org

i *Random+DBN*: a randomly initialized fully convolutional network of Section III-C with the unsupervised DBN of [30] (see Section II-B);
ii *Spectral Flux+DP*: feature engineering (spectral flux) and dynamic programming [29] (see Section II-B);
iii *ZeroNS+DP*: fully convolutional network of Section III-C trained with sync-pred on percussive and non-percussive parts of the FMA dataset (see Section III-D), followed by maximum pooling between the resulting embeddings (see below) and dynamic programming;
iv *ZeroNS+DBN*: same as above, dynamic programming replaced by the unsupervised DBN;

The results of the above can be found in Section IV-C. The next three rely on human supervision, in part or in full. Their results can be found in Section IV-D; we outline the beat tracking training procedure below:

v *Random+DBN*: same as (i), trained on each beat tracking dataset;
vi *ZeroNS+DBN*: same as (iv), fine-tuned on each beat tracking dataset;
vii *TCN+DBN*: temporal convolutional network with supervised training on multiple datasets, followed by the same DBN used in our experiments [43].

In systems (iii), (iv), and (vi), we merge the two branches of ZeroNS via maximum pooling between unnormalized percussive ($f_p$) and non-percussive embeddings ($f_{\neg p}$):

$$\boldsymbol{y}(t) = \max\left(f_p(\mathbf{U}\phi_p\boldsymbol{x})(t), f_{\neg p}(\mathbf{U}\phi_{\neg p}\boldsymbol{x})(t)\right). \quad (9)$$

This allows our system to perform beat tracking when either the percussive or non-percussive sources are silent; i.e. when $x$ contains no drums ($\phi_p\boldsymbol{x} = 0$) or nothing but drums ($\phi_{\neg p}\boldsymbol{x} = 0$). Conversely, in (i) and (v), the model comprises a single branch $f$ which operates on the VQT of the mixture $\mathbf{U}\boldsymbol{x}$; thus, no merging is necessary.

In (iii), we pass $\boldsymbol{y}$ as observation to the dynamic program of [29] and retain its peaks as predicted beats. In (iv), we do the same with the DBN of [30]. We set its BPM range to $[55, 215]$. Both of these approaches (*ZeroNS+DP* and *ZeroNS+DBN*) combine a supervised source separation model (Spleeter), a self-supervised acoustic model (ZeroNS), and an unsupervised sequence model (DBN or DP): hence, they may be regarded as unsupervised with respect to the beat tracking task.

In (v) and (vi), we convert the human annotation into a signal $\boldsymbol{y}^*$ with the same hop length as $\boldsymbol{y}$. Following [43], the value of $\boldsymbol{y}^*(t)$ equals 1 if $t$ is nearest to an annotated beat, 0.5 if $t$ is adjacent to an annotated beat, and 0 otherwise.

We train the network $f$ to minimize the binary cross-entropy (BCE) between its prediction $\boldsymbol{y}$ and the ground truth $\boldsymbol{y}^*$:

$$-\frac{1}{N_{\boldsymbol{y}}}\sum_{t=1}^{N_{\boldsymbol{y}}} \boldsymbol{y}^*(t)\log(\boldsymbol{y}(t)) + (1-\boldsymbol{y}^*(t))\log(1-\boldsymbol{y}(t)), \quad (10)$$

where $N_{\boldsymbol{y}}$ is the duration of $\boldsymbol{y}$, on average for every $\boldsymbol{y}$. Because $N_{\boldsymbol{y}}$ varies across songs $\boldsymbol{y}$, we use a batch size of 1.

In (vi), we pre-train $f = (f_p, f_{\neg p})$ with sync-pred and fine-tune it on Equation 10 with the Adam optimizer and a small learning rate of $5 \times 10^{-8}$ to prevent catastrophic forgetting. However, in (v), we initialize $f$ at random with i.i.d. Gaussian weights. We train $f$ on the beat tracking task in a supervised way, with a larger learning rate of $10^{-5}$. These learning rates were established by trial-and-error.

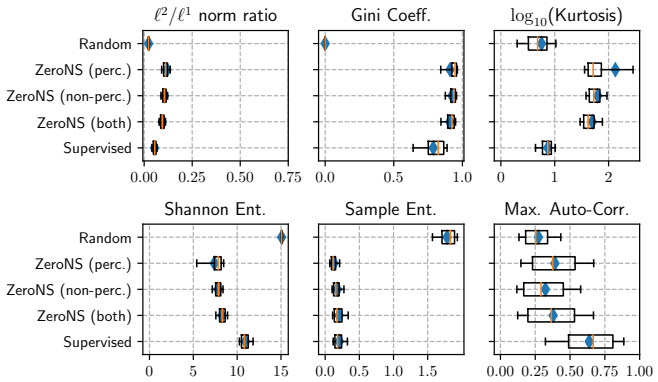



Fig. 2: Comparison of learned embeddings in terms of information-theoretic measures. Blue diamonds and orange bars denote means and medians respectively. Box and whisker edges denote quartiles and deciles respectively.

## IV. Evaluation and Results

In this section, we evaluate our models, both on the pretext task (sync-pred) and the downstream task (beat tracking). We demonstrate that pre-training with ZeroNS benefits unsupervised learning, low-data learning, and cross-dataset generalization, in comparison with supervised learning from a random initialization.

### A. Convergence on the Pretext Task

The ZeroNS neural network is able to train appropriately on the curated dataset. During the first few iterations on the training set, the cosine similarities between all percussive and non-percussive embeddings are very close to one, suggesting that the networks are unable to recognize synchronized and non-synchronized outputs. After 25 epochs, the ZeroNS model reaches a loss value of $0.634 \pm 0.30$. Synchronized embeddings in the validation set have a mean cosine similarity value of $0.804 \pm 0.13$ while non-synchronized embeddings have a mean cosine similarity value of $0.053 \pm 0.01$.

These results show that both our data curation process and pretext task definition enable the model to converge appropriately. This probably would not have been the case if we had not filtered out songs with very few drums, for example. In the end, we stopped training after 35 epochs. By then, the loss on the validation set had not diminished for 10 epochs. We therefore opted to stop training early. The network that achieved the lowest validation loss during the 35 epochs of training was selected for our downstream tasks. For a more comprehensive view of the network's evolution during training, we encourage the reader to consult the figures in the supplementary material of this paper.

### B. Emergence of Sparsity and Periodicity

When looking at an ideal beat tracking activation function, one can immediately notice that it is both very sparse and

resembles a periodic function. Its values are equal to 0 at all points except when a beat occurs. In this case, the value is equal to 1. These beat values are cyclical, and usually occur at almost constant time intervals. We therefore use six information-theoretic measures in order to better understand the embeddings generated by our pretext task, and their relevance for beat tracking. Maximizing performance on these measures is not crucial: in the event of tempo-varying music, the ground truth beat activation function is not exactly periodic. However, these cases only apply to a small subset of songs.

For sparsity, we use the following measures: the $\ell^2/\ell^1$ norm ratio, which is often used to encourage sparsity for non-convex problems such as non-negative matrix factorization [67]; the Gini coefficient, which is a common measure of statistical inequality; the kurtosis, a measure of the amount of outliers present in the distribution of a real-valued random variable; and the Shannon entropy, which quantifies the amount of information contained in our embeddings, the amount of surprise or uncertainty. A comprehensive overview of each measure can be found in [68]. We also report the sample entropy of our embeddings. Originally proposed for physiological time-series [69], this measure estimates the amount of regularity present in a signal. Finally, we compute the maximum autocorrelation value obtained for shifts of 0.25 to 4 seconds. Each embedding mean is subtracted before measuring the above. All values are also divided by the unlagged autocorrelation value. We compute each measure for five embeddings: a randomly initialized CNN; the percussive branch $f_p$ of ZeroNS; the non-percussive branch $f_{\neg p}$ of ZeroNS; the maximum of the two branches, $f$ (see Equation 9); and a supervised CNN trained on the Ballroom dataset (see Section IV-D). All measures are computed on the GTZAN dataset.

Figure 2 displays our results for each network. One can notice that most measures point towards our embeddings being sparser than the randomly initialized network's outputs. This is even more apparent for the percussive network. The Gini coefficient values of the ZeroNS networks are very close to one, for example, suggesting that a few values in our embeddings are large whilst the rest are close to zero. The kurtosis and $\ell^2/\ell^1$ norm ratio are also larger, suggesting a greater amount of outliers and sparsity. On the other hand, our pre-training leads to smaller entropy values. The sample entropy values of the ZeroNS network are especially similar to those of the supervised network. This suggests that our embeddings are both sparser and more regular than the ones generated by the randomly initialized network. The maximum normalized autocorrelation values obtained by each of ZeroNS' branches are also much closer to those of a supervised beat tracking CNN, and hence much more periodic than those of the randomly initialized network. Visualizing these embeddings seems to support this claim: we notice that our embeddings mostly have values that are close to zero, except for peaks that occur sporadically. These peaks seem to be synchronized with periodic musical onsets, and especially drum sounds such as kicks and claps. More work needs to be done in order to validate this hypothesis.

When comparing the ZeroNS outputs to a regular beat tracking network's outputs, we notice that for the $\ell^2/\ell^1$ norm ratio, Gini coefficient, and kurtosis, we obtain noticeably smaller values using the beat tracking network compared to our ZeroNS embeddings. The entropy values obtained using the former, on the other hand, are larger than the former's. Finally, we observe some overlap between the autocorrelation values found in each set of embeddings. Although these measures must not be interpreted as proof that our pretext task is relevant to our downstream task, they do show a trend: the generated outputs tend to be sparse and periodic, just like a supervised network's outputs. We invite the reader to consult the supplemental material of this paper in order to better visualize our embeddings.

| Dataset | # files | length |
|---|---|---|
| Ballroom | 685 | 5h57m |
| Hainsworth | 222 | 3h19m |
| GTZAN | 1000 | 8h20m |
| SMC | 217 | 2h25m |

TABLE I: Datasets used for Beat Tracking

### C. Unsupervised Representations

We evaluated our models on four annotated datasets: the Ballroom dataset[5] from [70], [71] and annotated using [41][6] , the Hainsworth dataset [72], the GTZAN dataset from [73] and annotated in [74], and the SMC dataset [75]. Unlike in [43], [12], [76], [39][7], we report results on each dataset separately. This was done in order to better study the effects of pre-training on small datasets. Each is annotated differently and spans a variety of musical genres. Table I further describes the datasets we used according to the number of files they contain and the duration of these files. We adopt the following evaluation metrics: F1-score, AMLc, AMLt, CMLc, and CMLt. The CMLc and CMLt metrics evaluate how continuously correct a beat tracking estimation is (use of the maximum length of correct predictions). The AMLt and AMLc metrics are similar but allow offbeat variations of an annotated beat sequence to be matched with detected beats. One can read more about each metric in [77].

Let us now explore the effects of pre-training on downstream task performance. In our first evaluation setting, we test our models in an unsupervised fashion. Our results can be found in Figure 3. When looking at the figure, one notices that the randomly initialized network is greatly outperformed by the *Spectral Flux+DP*, *ZeroNS+DP*, and *ZeroNS+DBN* methods. The randomly-initialized network does not surpass a mean CMLc score of 0.01 or an F1-score of 0.37 on each dataset. This was to be expected: it is neither trained or optimized to learn a form of musical meter, tempo, or synchrony to a pulse.

In contrast, both ZeroNS methods perform just as well, if not better, than the method in [29]. When used in conjunction with the latter's dynamic program, our ZeroNS method outperforms the *Spectral Flux+DP* method on 13/15 metrics for the SMC

[5]Online repository: https://github.com/CPJKU/BallroomAnnotations

[6]The duplicates in the Ballroom dataset identified in https://highnoongmt.wordpress.com/2014/01/23/ballroom_dataset/ were removed.

[7]In these papers, the SMC, Ballroom, and Hainsworth datasets are mixed during 8-fold cross validation. They use the GTZAN dataset as a test set.

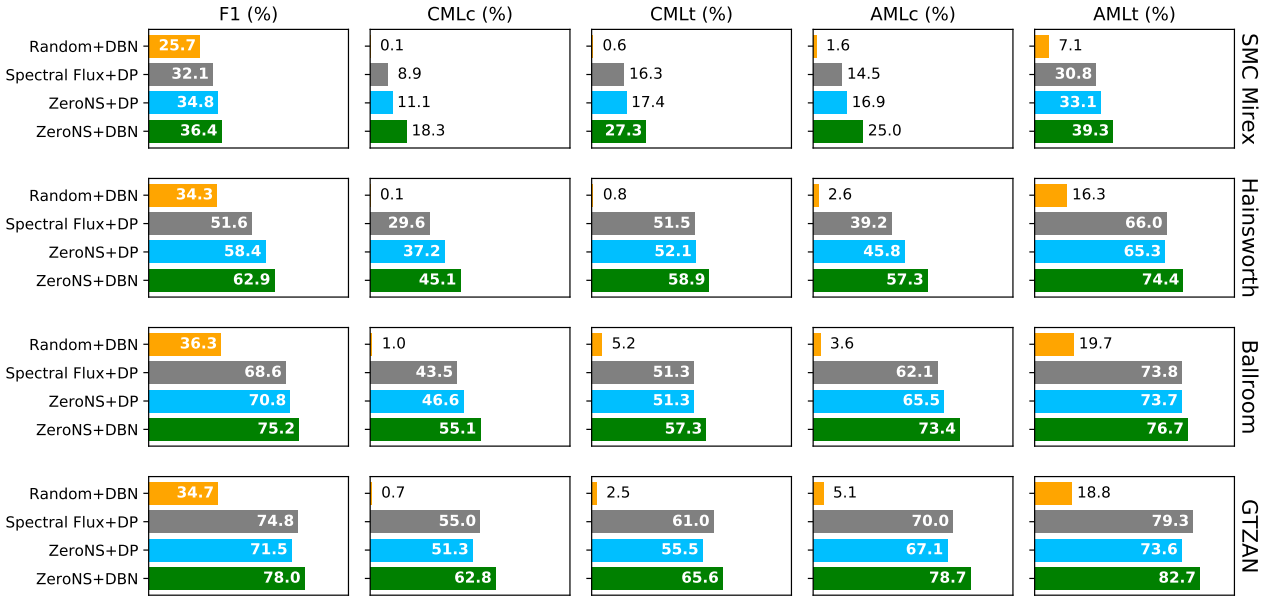



Fig. 3: Benchmark of four different unsupervised models on the beat tracking task: our randomly initialized fully-convolutional network followed by the DBN in [30]; Librosa's spectral flux and dynamic programming (DP) algorithm [29]; ZeroNS followed by DP; and ZeroNS followed by the DBN. Each row corresponds to a different dataset (indicated on the right) while each column corresponds to a different evaluation metric (at the top). Reported figures result from the mean scores for each track.

Mirex, Hainsworth, and Ballroom datasets. It is however consistently outperformed by the *Spectral Flux+DP* method on the GTZAN dataset. We leave as future work to determine where the ZeroNS method fails compared to it Spectral Flux counterpart. These results do suggest that the onset functions output by the ZeroNS network are at the very least on par with the ones generated by Spectral Flux [29] for beat tracking.

When used in conjunction with the SOTA DBN from [30], our ZeroNS model outperforms *Spectral Flux+DP* on all datasets for every single metric. When using an F1-score metric, the *ZeroNS+DBN* method beats the *Spectral Flux+DP* method by at least four percentage points. The method also achieves 70% (SMC Mirex), 80.3% (Hainsworth), 82.5% (Ballroom), and 92.1% (GTZAN) of the performance of its fully-supervised counterpart (*Random+DBN* in next section). When using an AMLc metric, the *ZeroNS+DBN* method beats the *Spectral Flux+DP* method by at least eight percentage points on every datset. The method also achieves 56.7%, 76.2%, 82.8%, and 92.3% of the performance of its fully-supervised counterpart. The same effect can be observed for the other three metrics. These results are, to the best of our knowledge, SOTA for unsupervised beat tracking.

### *D. Supervised Learning*

For the supervised models, we split our datasets into eight randomly selected folds. Each of these are then used as a test set during cross-validation. During each cross-validation iteration, the remaining folds' data is used for either our training or validation sets. These are selected randomly using a $\frac{6}{7}$ and $\frac{1}{7}$ split. For each 8-fold cross-validation iteration, training is stopped when the model's F1-score on the validation set has not increased for 20 epochs. The randomly initialized network and the fine-tuned ZeroNS network are all evaluated using the aforementioned methodology. Figure 4 displays our results on each dataset according to the metrics introduced previously. We also report the results from [43], obtained using a temporal convolutional network (TCN) and the the DBN from [30].

Fine-tuning our ZeroNS model on each beat tracking dataset does not yield the same promising results as the unsupervised learning setting. On all datasets and metrics, the *ZeroNS+DBN* network's performance is roughly on par with the *Random+DBN* network's performance. On the SMC and GTZAN datasets, for example, the fine-tuned network outperforms the randomly initialized network by a slight margin (maximum $\sim$ 0.03) on all metrics. We observe the opposite phenomenon on the Ballroom and Hainsworth datasets. This leads us to the following conclusion: in an 8-fold cross-validation setting, there is enough annotated training data for the randomly initialized and ZeroNS model to perform similarly.

When comparing our models' performances with the *TCN+DBN* network, we notice that the latter's performance on the SMC, Ballroom, and Hainsworth datasets is generally better for every metric. This is particularly striking on the smaller Hainsworth dataset, where their mean F1-score is 0.874 compared to our maximum of 0.783. On the GTZAN dataset, however, our *ZeroNS+DBN* network outperforms their method for every single metric except AMLc. Our *Random+DBN* network outperforms their method for every single metric except AMLt and AMLc. This makes sense: 8-fold cross-validation is performed on the combined SMC, Ballroom, and Hainsworth datasets in [43]. The GTZAN dataset, on the other hand, is used solely for testing purposes. We surmise that the *TCN+DBN* performance would be much closer to ours had they computed their results on each dataset individually.

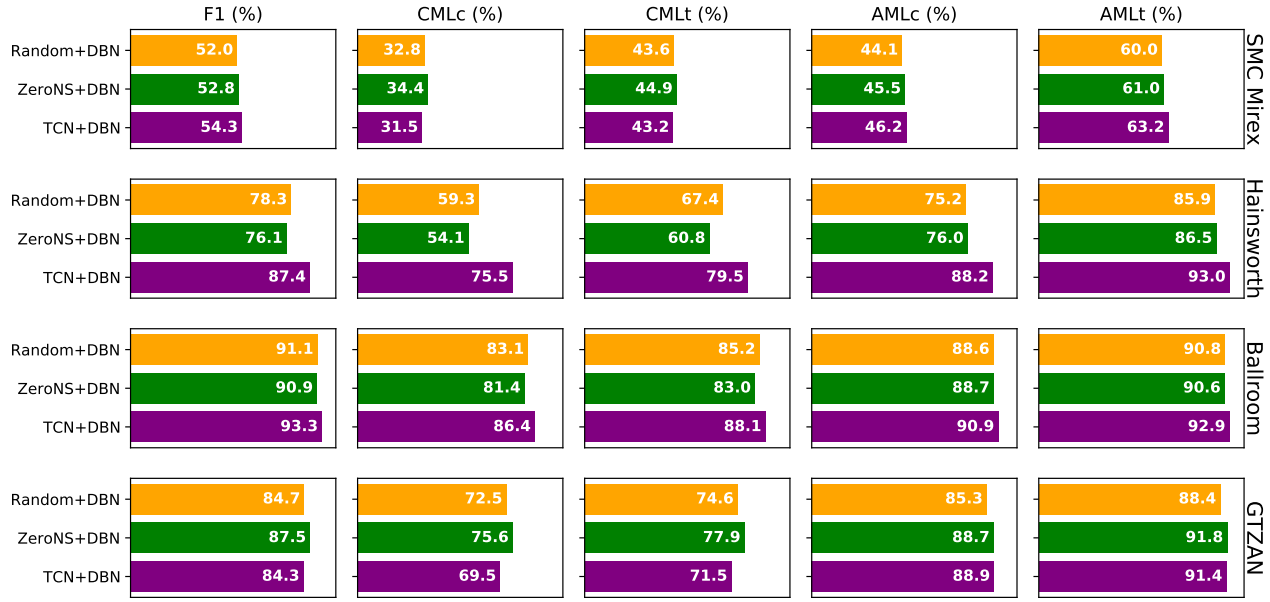



Fig. 4: Benchmark of three different supervised models on the beat tracking task: our randomly initialized fully-convolutional network followed by the DBN in [30]; ZeroNS followed by the DBN; and the SOTA temporal convolutional network of [43]. Rows, columns, and results follow the same structure as Figure 3.

### *E. Low-Data Beat Tracking*

We also test the effects of pre-training in a low-data learning setup. For each dataset, we extract two constant folds for testing and validation. These each contain an eighth of the total dataset. The remaining data is used to train a randomly initialized model and a ZeroNS model. The DBN from [30] is used in conjunction with all models. However, instead of using all the remaining data for training, we use subsets of size 1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64, and 96 tracks. For each subset, we train our networks on 10 randomly selected subsets of the training data, so as to maximize the diversity of our low-data learning reference examples. We also stop training once performance on the validation set has not increased for 20 epochs. The model that performs best on that set is selected for testing. Figure 5 displays our results. For each size of training set, we report the mean and standard deviation over 10 trials on all four datasets and all five evaluation metrics.

One can notice that for all metrics and datasets, the ZeroNS network significantly outperforms the randomly initialized network up to 24–32 tracks. Even better, the randomly initialized network struggles to learn anything until then; we observe mean F1-scores of less than 0.33 for a training set of less than 24 on the SMC dataset. More generally, the fully-supervised network only starts to match performance using 96 tracks when 32 tracks are used for training. In contrast, the ZeroNS network obtains an F1-score of 0.401 on the SMC dataset, or more than 77% of the fully-supervised network's performance on the 8-fold cross-validation setting of the previous experiment, with just one track for training. We observe similar behavior on the other datasets with a training set size of 1; the ZeroNS network has a mean F1-score of 0.604 on the Hainsworth dataset, 0.729 on the Ballroom dataset, and 0.820 on the GTZAN dataset versus 0.783, 0.911, and 0.847 for the randomly initialized network in the previous, fully-supervised experiment. As the number of training examples increases, the randomly initialized network's performance does catch up with the ZeroNS network's one.

The exact same behavior is observed using other evaluation metrics. On the Hainsworth dataset, the AMLc has a mean value of less than 0.04 when fewer than 12 examples are used to train a randomly initialized network. In comparison, the ZeroNS network reaches a value of 0.559 with just one training example. Both networks only start performing similarly when 24 examples are used (0.659 versus 0.736); the ZeroNS network still performs more than a standard deviation better in this case.

This leads us to conclude the following: our proposed pretext task significantly helps our network on the downstream task under a regime of limited labeled data. Indeed, ZeroNS embeddings already contain relevant information for beat tracking, and few annotated examples are needed for the system to perform well. A similar idea is broached in semi-supervised learning [78], where it is theorized that humans learn by being exposed to large amounts of unlabeled data, and a small amount of labeled examples that help guide their learning. Our network is learning in the same way; a large catalog of unlabeled data is used to understand the notion of musical beat, and only a few examples suffice to teach it how to track the beat.

### *F. Cross-Dataset Generalization*

Annotating a song's beats is a lengthy and time-consuming process. More importantly, most datasets are annotated using different techniques [77], which can lead to a form of bias. In our final evaluation setting, we explore the cross-dataset generalization capabilities of our pre-trained models. We train our randomly initialized and ZeroNS networks on each of the Ballroom, SMC, and Hainsworth datasets using 8-fold cross-validation. For each fold, the highest performing model is

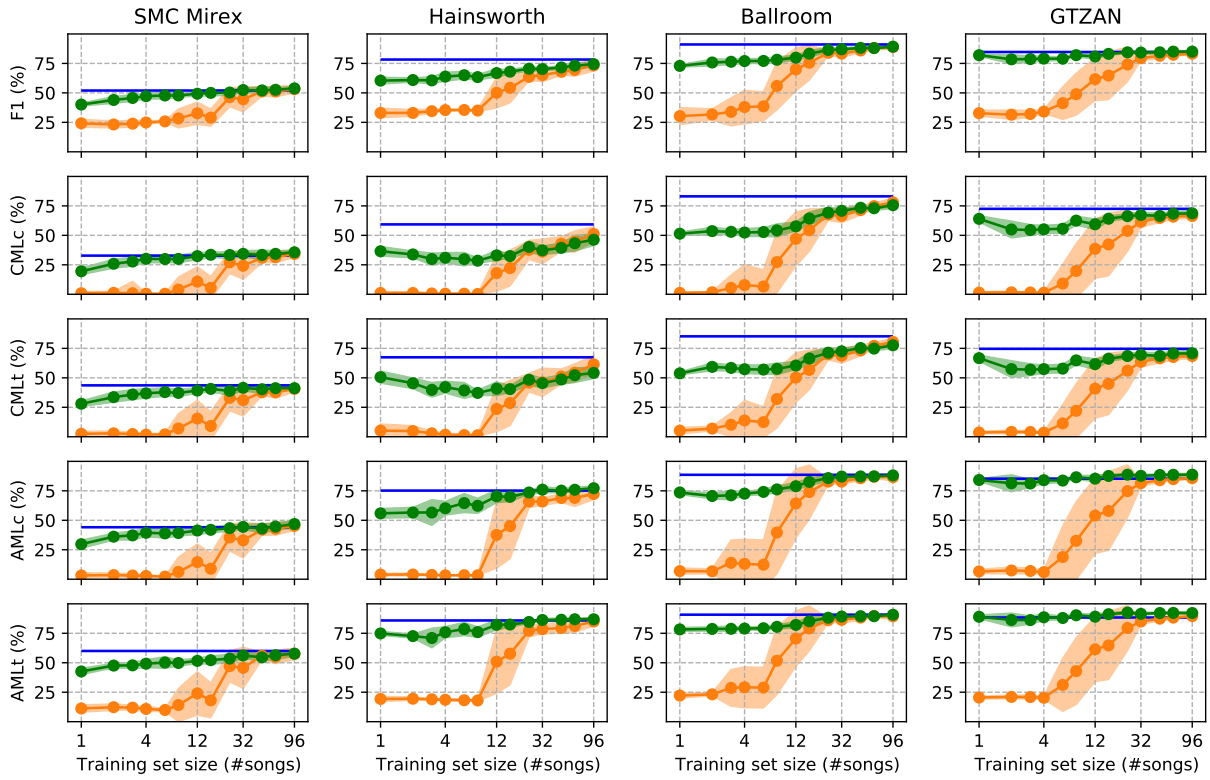



Fig. 5: *Random+DBN* and *ZeroNS+DBN* model performances on our low-data experiments. For all datasets, points represent scores using 1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64, and 96 tracks of the available training set. The green curve denotes fine-tuned ZeroNS and the orange curve, a supervised model with random initialization. Dots and shaded areas represent the mean and standard deviation of each metric across ten trials respectively. We display the *Random+DBN* results from Section IV-D as top-line results in blue.

tested on the GTZAN dataset, our largest and most diverse dataset. Figure 6 displays our results.

We observe that ZeroNS pre-training drastically improves our model's generalization capabilities. The gap in performance is quite stark: when training on the SMC dataset, the Random model's mean F1-score is 0.638 whereas the ZeroNS model's is 0.748; when training on the Hainsworth dataset, the Random model's mean AMLt is 0.834 whereas the ZeroNS model's is 0.894; when training on the Ballroom dataset, the Random model's mean CMLc is 0.584 whereas the ZeroNS model's is 0.656. These results demonstrate that ZeroNS pre-training helps mitigate the bias towards certain genres or annotation strategies. ZeroNS pre-training is however not on par with the training method from [43], where the SMC, Ballroom, and Hainsworth datasets are combined for training and the GTZAN dataset is used for evaluation. In the future, we hope to use our pre-training to improve beat tracking results on musical genres or styles with little to no annotated data, so as to expand the scope of MIR beyond Western music and the U.S. Billboard.

## V. Discussion and Conclusion

Unsupervised beat tracking, once a central theme in MIR, has now lost most of the attention it had two decades ago. Indeed, the surge of deep learning has shifted the focus towards supervised representation learning. Yet, the ability to learn a beat detection function with little or no supervision remains an urgent need for various reasons: because humans learn this way; because the overwhelming majority of recorded music is unlabeled; and because annotating audio is time-consuming.

In this article, we have proposed to revisit the old problem of unsupervised beat tracking with new tools: high-quality source separation, deep convolutional networks in the time–frequency domain, and most importantly, multitrack contrastive learning.

We have presented a new pretext task in self-supervised learning, which we named "synchrony prediction" (sync-pred), and which consists in distinguishing whether two musical parts (percussive and non-percussive) are in sync or out of sync. This is a simple model-agnostic task, involving a single audio file per batch and no artificial perturbations of the audio data. Thanks to an off-the-shelf source separation system (Spleeter), we have scaled up sync-pred to 98 hours of audio; that is, ten times more data than the largest annotated dataset for beat tracking (GTZAN, 9 hours). Besides, we have argued that sync-pred is interesting because it resembles two musicians grooving to a collective pulse when playing together.

We have implemented sync-pred as part of a deep learning model, named "Zero-Note Samba" or ZeroNS. In doing so, we do not claim that the architectural design of ZeroNS is optimal; only that it showcases some interesting properties of sync-pred. First, we observe that sync-pred essentially acts as

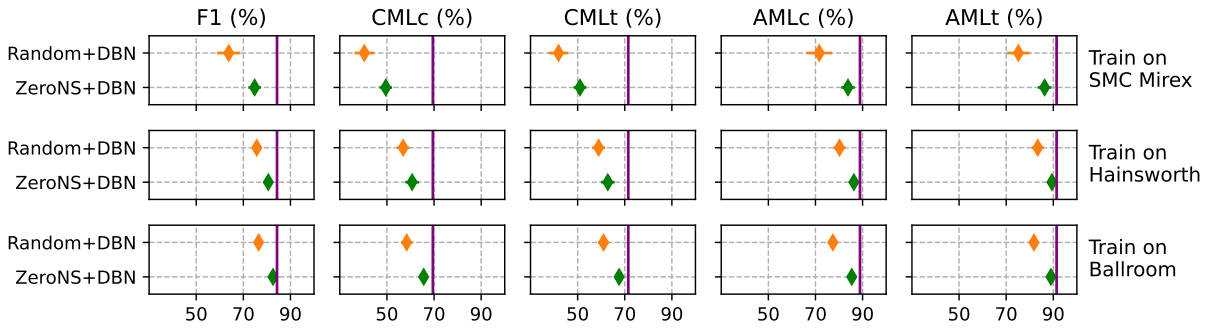



Fig. 6: Mean and standard deviation scores obtained on GTZAN when training on the SMC, Hainsworth, and Ballroom datasets. Green points represent the ZeroNS model's performance. Orange points represent the randomly initialized model's performance. Vertical purple lines represent the results of the SOTA temporal convolutional network of [43], which is trained on the SMC, Hainsworth, and Ballroom datasets and evaluated on GTZAN.

a "transientness prior" on the learned embeddings. Second, when used in conjunction with the dynamic programming algorithm from [29] or the dynamic Bayesian network from [30], the learned embeddings match or outperform previous SOTA unsupervised methods. Third, sync-pred offers a form of pre-training for supervised models that allows to drastically reduce the quantity of annotated data; typically from 100 songs down to ten songs or fewer. Fourth, this pre-training mitigates dataset bias and improves beat tracking metrics on a cross-dataset evaluation setting by two standard deviations. These findings lead us to think that sync-pred has potential in beat tracking, and perhaps other problems in which multiple sources, agents, or modalities operate in synchrony.

Our main negative finding is that, under a $k$-fold cross-validation scenario, pre-training ZeroNS with sync-pred does not improve beat tracking performance, or marginally so. It remains unclear how knowledge may be transferred from pretext to downstream task in a way that favors within-dataset generalization. In a recent report [79], we have proposed to do so with coincidence prediction instead of sync-pred, yet with limited success so far. Variations in the formulation of multitrack contrastive learning or in the architecture of ZeroNS might solve this shortcoming in the future.

Beyond the task of beat tracking, it would be valuable to inquire how sync-pred could be employed at larger musical time scales: namely, downbeat estimation and structure analysis. To this end, sync-pred may not only be formulated between two audio sources, but also between audio and video, audio and motion, or audio and neurophysiology. Lastly, we note that sync-pred is not limited to the time domain: it could also be formulated in the frequency domain, with downstream tasks such as tonality estimation and chord transcription.

## REFERENCES

[1] J. London, *Hearing in time: Psychological aspects of musical meter*. Oxford University Press, 2012.
[2] P. Janata, S. T. Tomic, and J. M. Haberman, "Sensorimotor coupling in music and the psychology of the groove." *J. Exp. Psychol. Gen.*, vol. 141, no. 1, p. 54, 2012.
[3] W. Fitch, "Rhythmic cognition in humans and animals: Distinguishing meter and pulse perception," *Front. Syst. Neurosci.*, vol. 7, p. 68, 2013.
[4] H. Honing *et al.*, "Structure and interpretation of rhythm and timing," *Tijdschrift voor Muziektheorie*, vol. 7, no. 3, pp. 227–232, 2002.
[5] H. Honing, "Without it no music: beat induction as a fundamental musical trait," *Ann. N. Y. Acad. Sci.*, vol. 1252, no. 1, pp. 85–91, 2012.
[6] M. F. McKinney and D. Moelants, "Ambiguity in tempo perception: What draws listeners to different metrical levels?" *Music Percept.*, vol. 24, no. 2, pp. 155–166, 2006.
[7] I. Winkler, G. P. Háden, O. Ladinig, I. Sziller, and H. Honing, "Newborn infants detect the beat in music," *Proc. NY Ac. Sci.*, vol. 106, no. 7, pp. 2468–2471, 2009.
[8] Y. Seki and K. Tomyta, "Effects of metronomic sounds on a self-paced tapping task in budgerigars and humans," *Curr. Zool.*, vol. 65, no. 1, pp. 121–128, 2019.
[9] P. Cook, A. Rouse, M. Wilson, and C. Reichmuth, "A california sea lion (zalophus californianus) can keep the beat: motor entrainment to rhythmic auditory stimuli in a non vocal mimic." *J. Comp. Psychol.*, vol. 127, no. 4, p. 412, 2013.
[10] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.
[11] S. Durand, J. P. Bello, B. David, and G. Richard, "Robust downbeat tracking using an ensemble of convolutional networks," *IEEE/ACM Trans Audio. Speech. Lang. Process.*, vol. 25, no. 1, pp. 76–89, 2016.
[12] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks." in *Proc. ISMIR*. New York City, 2016, pp. 255–261.
[13] R. Vogl, M. Dorfer, G. Widmer, and P. Knees, "Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks." in *Proc. ISMIR*, 2017, pp. 150–157.
[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[15] A. M. Zador, "A critique of pure learning and what artificial neural networks can learn from animal brains," *Nat. Commun.*, vol. 10, no. 1, pp. 1–7, 2019.
[16] M. Fuentes, B. McFee, H. Crayencour, S. Essid, and J. Bello, "Analysis of common design choices in deep learning systems for downbeat tracking," in *Proc. ISMIR*, 2018.
[17] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. ISMIR*, 2011.
[18] M. Fuentes, L. S. Maia, M. Rocamora, L. W. Biscainho, H. C. Crayencour, S. Essid, and J. P. Bello, "Tracking beats and microtiming in Afro-Latin American music using conditional random fields and deep learning," in *Proc. ISMIR*, 2019.
[19] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Trans. Audio Speech Lang. Process.*, vol. 14, no. 1, pp. 342–355, 2005.
[20] G. Peeters and J. Flocon-Cholet, "Perceptual tempo estimation using gmm-regression," in *Proc. Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies*. ACM, 2012, pp. 45–50.
[21] M. Goto and Y. Muraoka, "A real-time beat tracking system for audio signals," in *Proc. ICMC*, 1995.
[22] T. Lenc, H. Merchant, P. E. Keller, H. Honing, M. Varlet, and S. Nozaradan, "Mapping between sound, brain and behaviour: Four-level framework for understanding rhythm processing in humans and non-human primates," *Philos. Trans. R. Soc. B*, vol. 376, no. 1835, 2021.
[23] B. De Man, M. Mora-Mcginity, G. Fazekas, and J. D. Reiss, "The open multitrack testbed," in *Proc. AES*. Audio Engineering Society, 2014.

[24] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "MedleyDB: A multitrack dataset for annotation-intensive MIR research," in *Proc. ISMIR*, vol. 14, 2014, pp. 155–160.

[25] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *J. Open Source Softw.*, vol. 5, no. 50, p. 2154, 2020.

[26] N. Turpault, S. Wisdom, H. Erdogan, J. Hershey, R. Serizel, E. Fonseca, P. Seetharaman, and J. Salamon, "Improving sound event detection in domestic environments using sound separation," in *Proc. DCASE*, 2020.

[27] G. Meseguer-Brocal, R. Bittner, S. Durand, and B. Brost, "Data cleansing with contrastive learning for vocal note event annotations," in *Proc. ISMIR*, 2020.

[28] E. Fonseca, A. Jansen, D. P. Ellis, S. Wisdom, M. Tagliasacchi, J. R. Hershey, M. Plakal, S. Hershey, R. C. Moore, and X. Serra, "Self-supervised learning from automatically separated sound scenes," in *Proc. IEEE ICASSP*, 2021.

[29] D. P. Ellis, "Beat tracking by dynamic programming," *J. New Mus. Res.*, vol. 36, no. 1, pp. 51–60, 2007.

[30] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking." in *Proc. ISMIR*, 2015, pp. 72–78.

[31] H. C. Longuet-Higgins, "Perception of melodies," *Nature*, vol. 263, no. 5579, pp. 646–653, 1976.

[32] H. C. Longuet-Higgins and C. S. Lee, "The perception of musical rhythms," *Perception*, vol. 11, no. 2, pp. 115–128, 1982.

[33] R. B. D. B. Mont-Reynaud and R. Dannenberg, "Following an improvisation in real time'," in *Proc. ICMC*, 1987.

[34] E. W. Large and J. F. Kolen, "Resonance and the perception of musical meter," *Conn. Sci.*, vol. 6, no. 2-3, pp. 177–208, 1994.

[35] A. J. Lambert, T. Weyde, and N. Armstrong, "Adaptive frequency neural networks for dynamic pulse and metre perception." in *Proc. ISMIR*, 2016, pp. 60–66.

[36] M. Goto and Y. Muraoka, "Issues in evaluating beat tracking systems," in *Proc. IJCAI Workshop*, 1997, pp. 9–16.

[37] M. Alonso, B. David, and G. Richard, "Tempo and beat estimation of musical signals," in *Proc. ISMIR*, 2004.

[38] B. McFee, V. Lostanlen, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, D. Lee, K. Lee, O. Nieto, J. Mason, D. Ellis, E. Battenberg, S. Seyfarth, R. Yamamoto, K. Choi, viktorandreevichmorozov, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, and T. Kim, "librosa/librosa: 0.8.0," Jul. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3955228

[39] S. Böck and M. Schedl, "Enhanced beat tracking with context-aware neural networks," in *Proc. DAFx*, 2011, pp. 135–139.

[40] N. Whiteley, A. T. Cemgil, and S. J. Godsill, "Bayesian modelling of temporal structure in musical audio." in *Proc. ISMIR*. Citeseer, 2006, pp. 29–34.

[41] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio." in *Proc. ISMIR*. Citeseer, 2013, pp. 227–232.

[42] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proc. ACM MM*, Amsterdam, The Netherlands, 10 2016, pp. 1174–1178.

[43] M. Davies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," in *Proc. EUSIPCO*. IEEE, 2019.

[44] Y.-N. Hung, J.-C. Wang, X. Song, W.-T. Lu, and M. Won, "Modeling beats and downbeats with a time-frequency transformer," in *Proc. IEEE ICASSP*. IEEE, 2022, pp. 401–405.

[45] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised Learning: Generative or Contrastive," *IEEE Trans. Knowl. Data Eng.*, no. 01, pp. 1–1, 2021.

[46] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. ICML*. PMLR, 2020, pp. 1597–1607.

[47] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *Proc. IEEE ICASSP*. IEEE, 2021, pp. 3875–3879.

[48] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," in *Proc. ISMIR*, 2021.

[49] B. Gfeller, C. Frank, D. Roblek, M. Sharifi, M. Tagliasacchi, and M. Velimirović, "SPICE: Self-supervised pitch estimation," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 1118–1128, 2020.

[50] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *Proc. ICCV*, 2017, pp. 609–617.

[51] I. Kavalerov, S. Wisdom, H. Erdogan, B. Patton, K. Wilson, J. Le Roux, and J. R. Hershey, "Universal sound separation," in *Proc. IEEE WASPAA*. IEEE, 2019, pp. 175–179.

[52] K. Lee and J. Nam, "Learning a joint embedding space of monophonic and mixed music signals for singing voice," in *Proc. ISMIR*, 2019.

[53] K. Kim, J. Lee, S. Kum, and J. Nam, "Learning a cross-domain embedding space of vocal and mixed audio with a structure-preserving triplet loss," in *Proc. ISMIR*, 2021.

[54] C.-Y. Chiu, A. W.-Y. Su, and Y.-H. Yang, "Drum-aware ensemble architecture for improved joint musical beat and downbeat tracking," *IEEE Signal Processing Letters*, vol. 28, pp. 1100–1104, 2021.

[55] C.-Y. Chiu, J. Ching, W.-Y. Hsiao, Y.-H. Chen, A. W.-Y. Su, and Y.-H. Yang, "Source separation-based data augmentation for improved joint beat and downbeat tracking," in *Proc. EUSIPCO*. IEEE, 2021, pp. 391–395.

[56] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proc. ISMIR*, 2017.

[57] M. Defferrard, S. P. Mohanty, S. F. Carroll, and M. Salathé, "Learning to recognize musical genre from audio," in *Proc. Web Conf.*, 2018.

[58] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio Speech. Lang. Process.*, vol. 14, no. 4, pp. 1462–1469, 2006.

[59] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[60] J. Schlüter and S. Böck, "Musical onset detection with convolutional neural networks," in *Proc. MML*, 2013.

[61] J. Andén and S. Mallat, "Scattering representation of modulated sounds," in *Proc. DAFx*, 2012.

[62] C. Schörkhuber, A. Klapuri, N. Holighaus, and M. Dörfler, "A matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution," in *Proc. AES*. Audio Engineering Society, 2014.

[63] B. R. Glasberg and B. C. Moore, "Derivation of auditory filter shapes from notched-noise data," *Hear. Res.*, vol. 47, no. 1-2, pp. 103–138, 1990.

[64] S. Mallat, "Understanding deep convolutional networks," *Philos. Trans. R. Soc. A:*, vol. 374, no. 2065, p. 20150203, 2016.

[65] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," in *Proc. ISMIR*, 2017.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[67] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints." *Journal of machine learning research*, vol. 5, no. 9, 2004.

[68] N. Hurley and S. Rickard, "Comparing measures of sparsity," *IEEE Transactions on Information Theory*, vol. 55, no. 10, pp. 4723–4741, 2009.

[69] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 278, no. 6, pp. H2039–H2049, 2000.

[70] S. Dixon, F. Gouyon, G. Widmer *et al.*, "Towards characterisation of music via rhythmic patterns," in *Proc. ISMIR*, 2004.

[71] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Trans. Audio Speech Lang Process.*, vol. 14, no. 5, pp. 1832–1844, 2006.

[72] S. W. Hainsworth and M. D. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP J. Adv. Signal Process.*, vol. 2004, no. 15, pp. 1–11, 2004.

[73] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293–302, 2002.

[74] U. Marchand and G. Peeters, "Swing ratio estimation," in *Proc. DAFx*, 2015.

[75] A. Holzapfel, M. E. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon, "Selective sampling for beat tracking evaluation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 9, pp. 2539–2548, 2012.

[76] S. Böck, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles." in *Proc. ISMIR*. Citeseer, 2014, pp. 603–608.

[77] 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.

[78] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[79] D. Desblancs, "Self-supervised beat tracking in musical signals with polyphonic contrastive learning," Master's thesis, École normale supérieure Paris-Saclay, 2021, https://arxiv.org/abs/2201.01771.

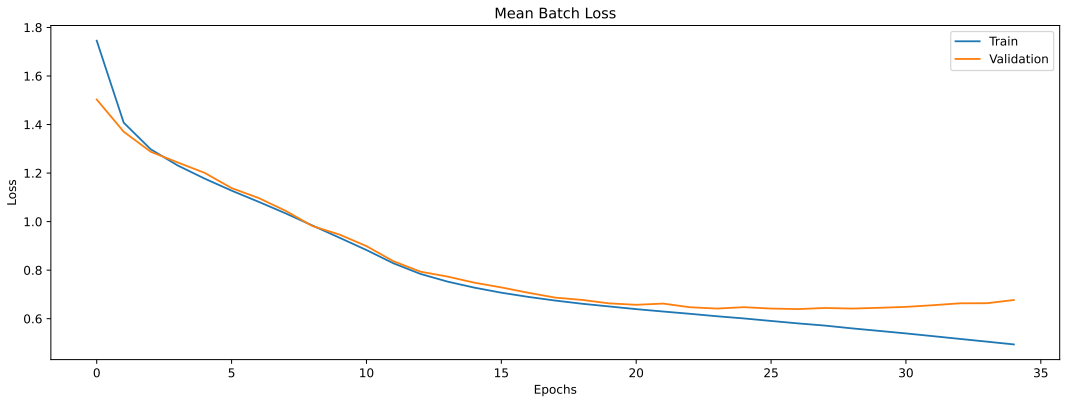



Fig. 1: Normalized temperature-scaled cross-entropy (NT-Xent) loss evolution over 35 epochs. The blue curve represents the training mean batch loss over 28,800 batches while the orange curve represents the validation mean batch loss over 6400 batches. As one can notice, the ZeroNS model starts to overfit after approximately 20 epochs. This motivated us to stop training.

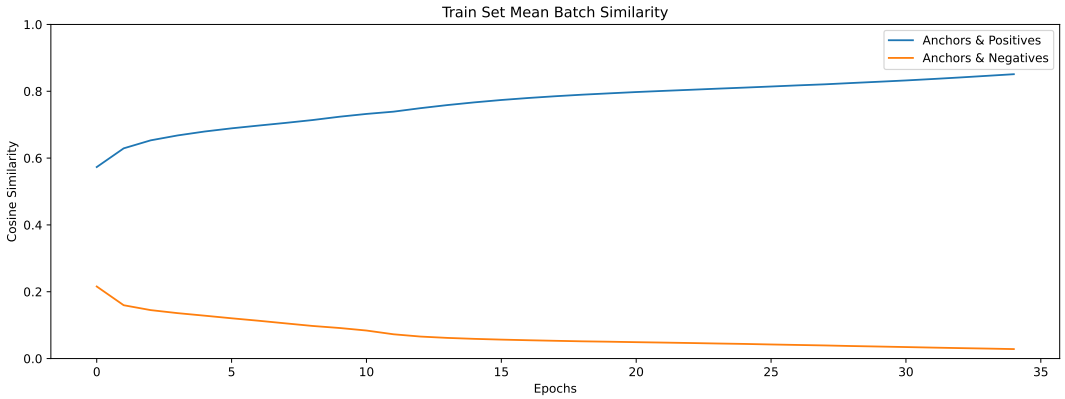



Fig. 2: Cosine similarity evolution on the training set over 35 epochs. The blue curve represents the mean cosine similarity between synchronized percussive parts (*positives*) and non-percussive parts (*anchors*). The orange curve represents the mean cosine similarity between non-synchronized percussive parts (*negatives*) and non-percussive parts (*anchors*).

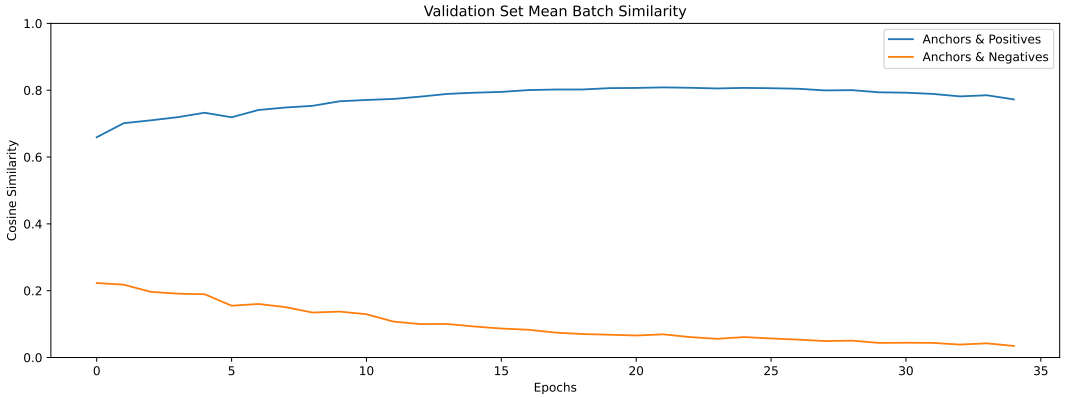
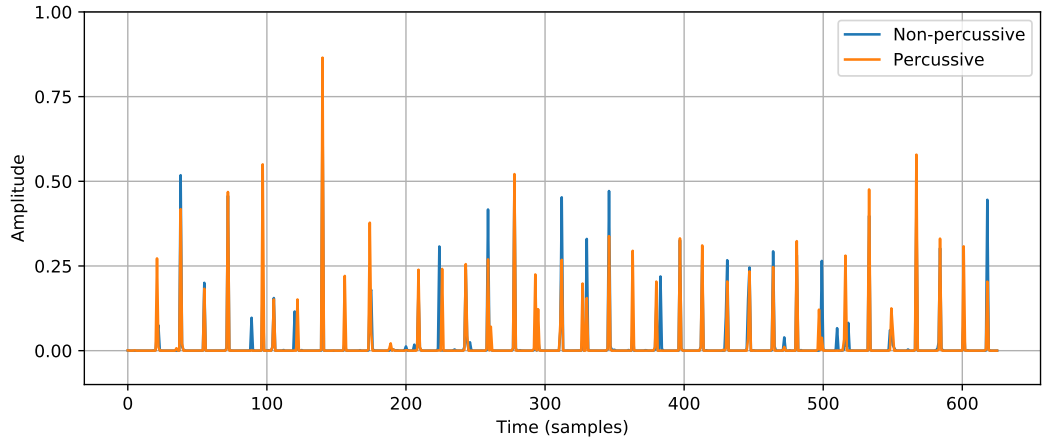
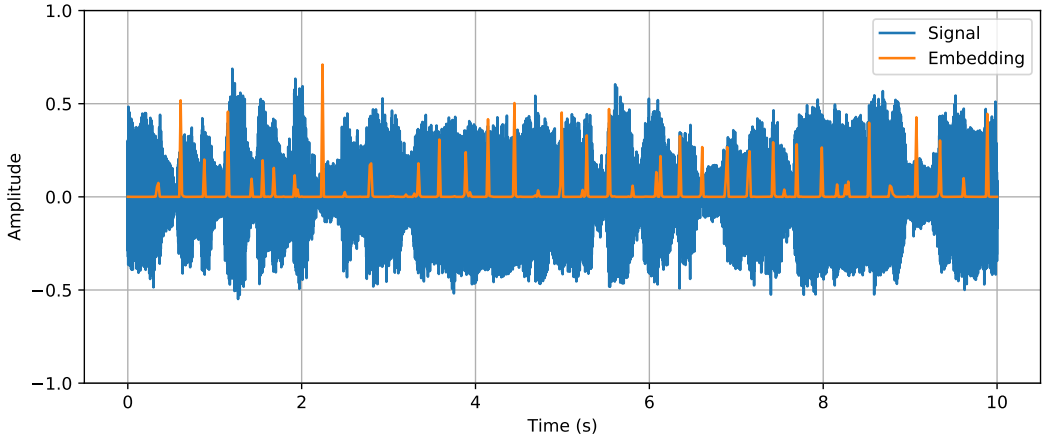


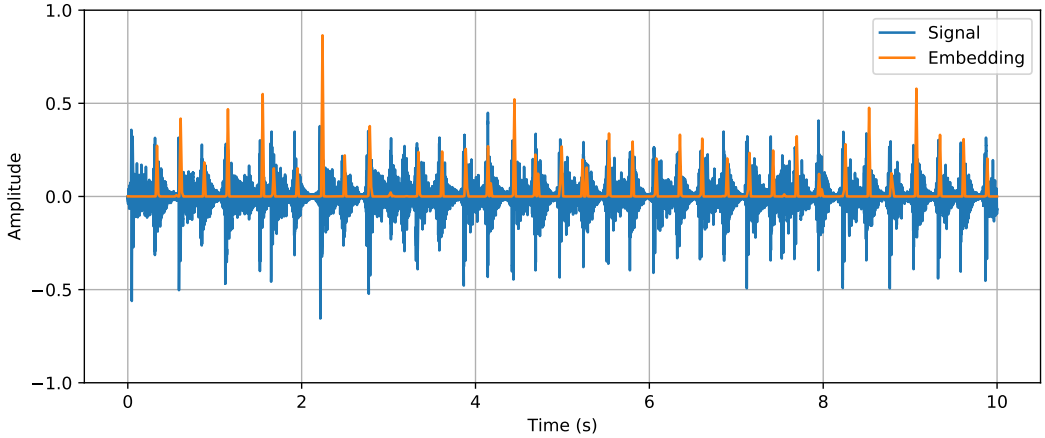

Fig. 3: Cosine similarity evolution on the validation set over 35 epochs. The blue curve and orange curves represent the same values as in the previous figure.




(a) Overlapped Embeddings

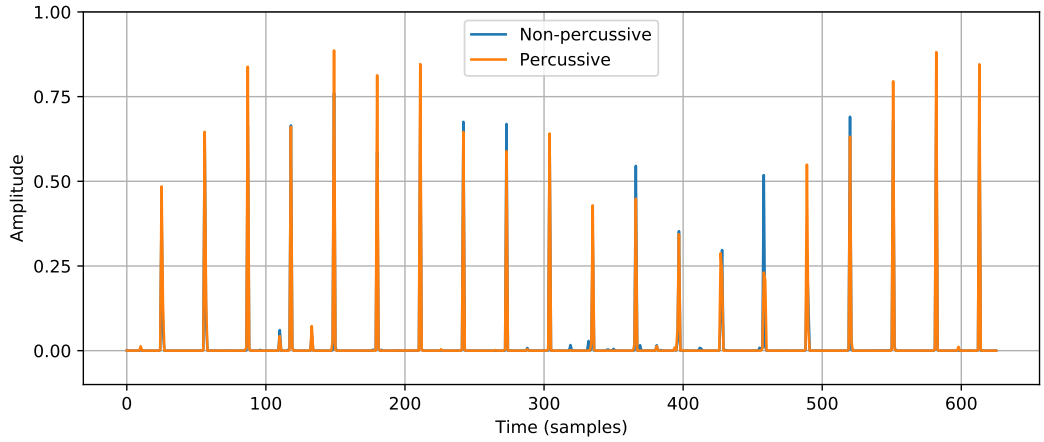



(b) Overlapped Non-percussive Waveform and Embedding

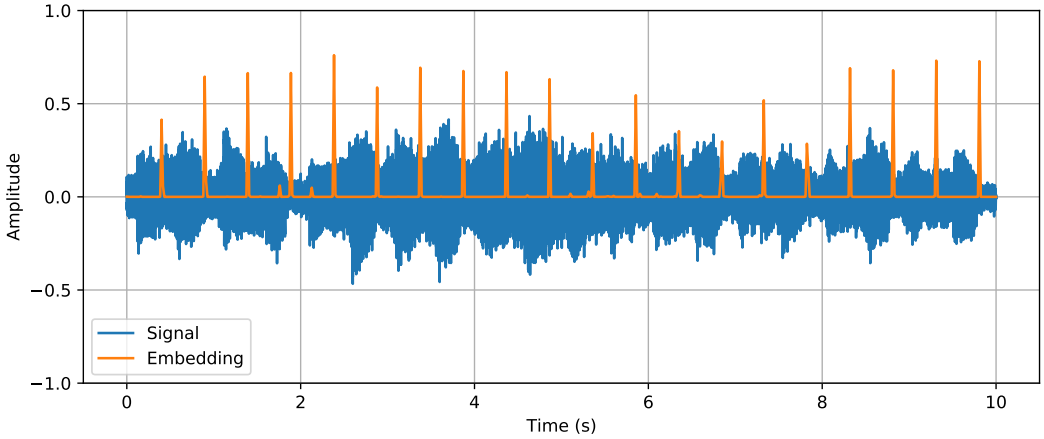



(c) Overlapped Percussive Waveform and Embedding

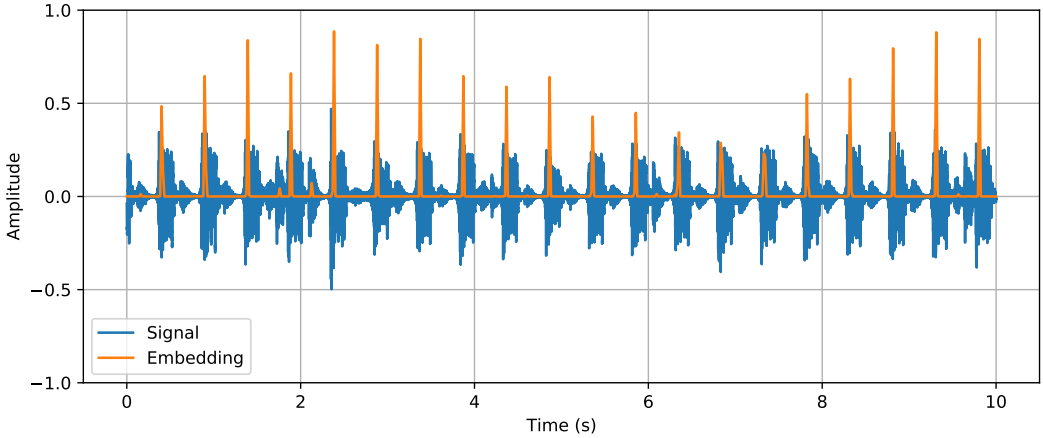
Fig. 4: First example signal. The cosine similarity between percussive and non-percussive embeddings is equal to 0.861 here.




(a) Overlapped Embeddings

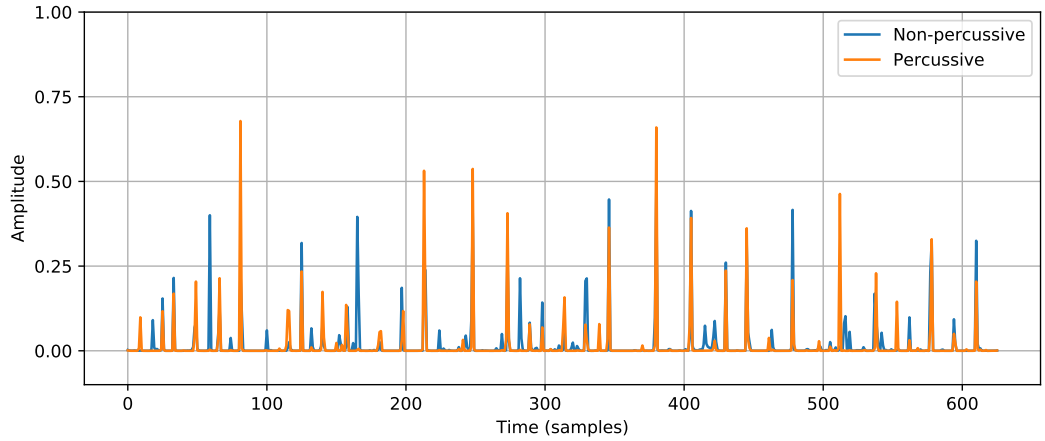



(b) Overlapped Non-percussive Waveform and Embedding

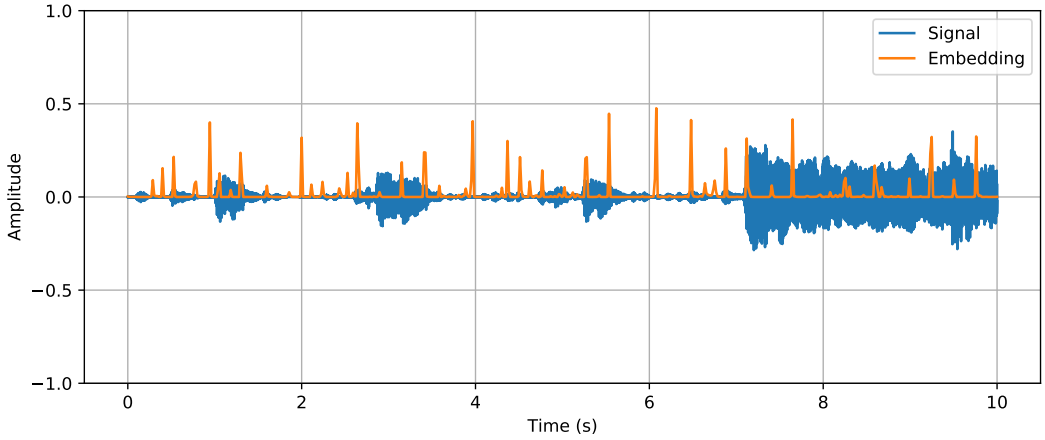



(c) Overlapped Percussive Waveform and Embedding

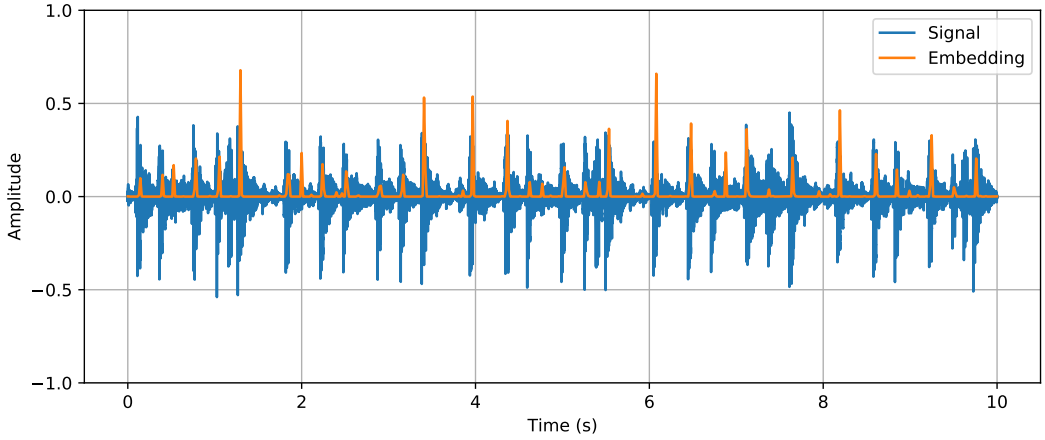
Fig. 5: Second example signal. The cosine similarity between percussive and non-percussive embeddings is equal to 0.971 here.




(a) Overlapped Embeddings




(b) Overlapped Non-percussive Waveform and Embedding




(c) Overlapped Percussive Waveform and Embedding

Fig. 6: Third example signal. The cosine similarity between percussive and non-percussive embeddings is equal to 0.747 here. Notice how the embedding during the "silent" parts of the non-percussive signal still contains peaks that could be seen as musical beats.