Adapting Pitch-Based Self Supervised Learning Models for Tempo Estimation

Antonin Gagneré, Slim Essid, Geoffroy Peeters

ADAPTING PITCH-BASED SELF SUPERVISED LEARNING MODELS FOR TEMPO ESTIMATION

Antonin Gagneré, Slim Essid, Geoffroy Peeters

LTCI - Telécom Paris, Institut Polytechnique de Paris, France

ABSTRACT

Tempo estimation is the task of estimating the periodicity of the dominant rhythm pulse of a music audio signal. It has therefore a close relationship with dominant pitch estimation. Recently, both tasks have been addressed in a Self-Supervised Learning (SSL) fashion so as to leverage unlabelled data for training. In this work, we study the applicability of two successful pitch-based SSL models, SPICE and PESTO, for the purpose of tempo estimation. Both successfully exploit Siamese networks with a pitch-shifting view generation between the two branches. To apply these models for tempo estimation, we represent the audio signal by the Constant-Q transform (CQT) of its onset-strength-function and adapt their view generation using time-stretching (instead of pitch shifting), which is efficiently implemented by shifting the CQT. In a large experiment, we show that simply adapting PESTO in this way yields superior results than the previous SSL approach to tempo estimation for most datasets used in the reference benchmark. Further, since PESTO is light-weight, requiring only a few training data, we study a new learning scheme where the downstream datasets are processed directly in a SSL fashion (without access to labels) showing that this is an interesting alternative further improving the performance for some datasets.

Index Terms— tempo estimation, self-supervised-learning

1. INTRODUCTION

Given its wide range of applications (recommendation, playlist generation, synchronization, dj-ing, audio or audio/video editing, beat-synchronous analysis), tempo estimation remains a major tasks in Music Information Retrieval (MIR). At its core, tempo estimation seeks to estimate the periodicity of the dominant rhythm pulse of a music audio signal, often expressed in beat per minute (BPM). Formulated in such a manner it has a strong resemblance to the task of pitch estimation. Recently, there has been a notable shift in the task of pitch estimation towards the adoption of Self-Supervised Learning (SSL). This has shown superiority over the conventional supervised models [1, 2]. In this work we explore the adaptation of such pitch-based SSL systems to the task of tempo estimation.

Works related to SSL in MIR. In recent years, the field of machine learning has been significantly reshaped with the emergence of SSL. Unlike supervised learning, which relies on vast amounts of labeled data, SSL leverages unlabeled data to learn meaningful representations. By designing tasks where the data itself provides the supervision, SSL methods have shown potential in reducing the need for expensive annotations while still capturing the underlying structures and patterns inherent in the data. A traditional technique employs a Siamese network [3, 4, 5, 6, 7] in which two different views of an input are fed to a neural network which is trained by applying a criterion (*invariance* or *equivariance*) between the two corresponding output embeddings.

Recently, the equivariance-based approach has been used for the development of MIR systems for pitch and tempo. Among those SPICE (Self-supervised PItCh Estimation) [1] defines the two views by pitch-shifting a given signal of one view to obtain the other, observing that within the Constant-Q transform (CQT) domain this simply maps to a translation. They trained the network such that the difference between the two embeddings is proportional to the known pitch-shifting factor applied. They add an extra decoder to ensure regularization. PESTO (Pitch Estimation with Self-Supervised Transposition-Equivariant Objective) [2] tackles pitch estimation with a much lighter network (without the need for a decoder) and formulate the equivariance following a classification approach. PESTO has been shown to have superior performances than SPICE. As for tempo estimation, to our knowledge only one attempt has been made: Quinton [8] defines the two views by time-stretching a given music signal. Those are fed to a TCN encoder which is trained solely based on an equivariant loss, that does not admit trivial solutions, hence avoids collapse.

Works related to tempo estimation. The task of tempo estimation traditionally relied on hand-crafted signal processing and statistical models. With the rise of datasets annotated into tempo, data-driven approaches became prevalent in the field. [9] pioneered the use of deep learning techniques for tempo estimation. Their pipeline employed a Bi-LSTM to predict beat position in a signal which is then processed by a bank of resonating comb filters to output a tempo estimation. Later on, [10] proposed the first deep learning only system. They used a CNN that directly predicts the local tempo (defined as tempo classes) from a Mel-spectrogram, alleviating the need to create mid level features. The current state-of-the-art in tempo estimation is obtained using a smartly designed TCN system [11, 12] that is trained to jointly estimate beat, downbeat and tempo [13]. Hybrid systems that use both domain-knowledge and deep-learning have also been proposed. [14] uses explicitly the harmonic series representation of periodic signals (proposed by [15, 16] to represent tempo and rhythm) through a Harmonic-Constant-Q-Modulation as input to a CNN. We use a similar (but simplified) representation here.

Close to our work, is the work of [17] which explores the adaptation of SPICE for the task of tempo estimation. The authors studied the relationship between the learned representation and the data representation by using synthetic data. However they did not apply it to the task of tempo estimation which is our main objective here.

Paper objective and organization. The objective of this work is to show that pitch-based SSL models can be effectively adapted to the task of tempo estimation. For this we study the adaptation of two state-of-the-art SSL pitch-estimation models, SPICE and PESTO, for the task of tempo estimation. We first detail their process (section

This work was performed using HPC resources from GENCI-IDRIS (Grant 2022-AD011013924). The material contained in this document is based upon work funded by the ANR-IA and Hi! PARIS

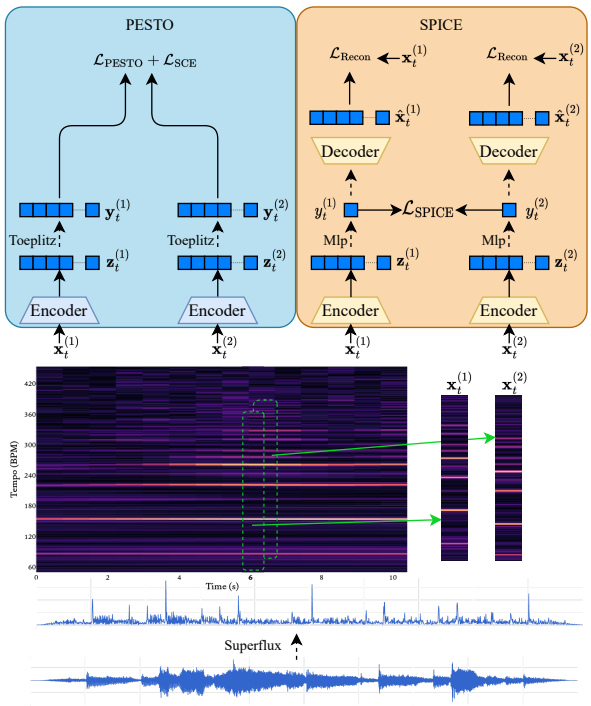

Fig. 1: Training routines for PESTO (top left) and SPICE (top right), with the proposed audio front-end for their adaptation to tempo estimation (bottom).

2) and explain how we adapt them (section 3) to our needs. We then perform a large-scale evaluation (section 4) comparing SPICE and PESTO, studying the influence of the dataset used for the SSL training, comparing our results to Quinton's SSL model using the same protocol [8] and to the results of fully-supervised systems.

With consideration to reproducibility we release the training code and annotations used for evaluation.[1]

2. SSL APPROACHES TO PITCH ESTIMATION

In both SPICE and PESTO, the encoders are fed with two different views $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ which represent pitch-shifted versions of the same signal frame around time t. Those are CQT slices with different vertical offsets. From these inputs the models are forced to learn pitch-equivariant representation $y_t^{(1)}$ and $y_t^{(2)}$. This equivariance is formulated differently for SPICE and PESTO. We illustrate this in Figure 1. We denote by $\mathbf{z}_t^{(1,2)}$ the outputs of the encoders.

In SPICE, $\mathbf{z}_t^{(1)}$ and $\mathbf{z}_t^{(2)}$ are projected using the same "pitch-head" (a 2 hidden-layer MLP) to two scalars $y_t^{(1)}$ and $y_t^{(2)}$. They represent relative pitches. SPICE is trained to predict the known amount $k_t^{(1)} - k_t^{(2)}$ of pitch-shifting between $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$. To do so, SPICE minimizes the following loss:

$$\mathcal{L}_{\text{SPICE}} = h\left(\left|\left(y_t^{(1)} - y_t^{(2)}\right) - \sigma\left(k_t^{(1)} - k_t^{(2)}\right)\right|\right) \quad (1)$$

where h is a proposed Huber loss function and σ a scaling factor to force the model to capture a given pitch range.

In PESTO, $\mathbf{y}_t^{(1,2)}$ are vectors which present the probability distribution over the pitch classes. The equivariance loss is derived from the following: if $\mathbf{x}_t^{(2)}$ is a pitch-shifted version of $\mathbf{x}_t^{(1)}$, then their pitch probability distributions, $\mathbf{y}_t^{(1)}$ and $\mathbf{y}_t^{(2)}$, should be shifted accordingly. To capture such information the authors define the following linear form : $\phi : \mathbb{R}^d \to \mathbb{R}$, $\mathbf{y} \mapsto (\alpha, ..., \alpha^d)^\top \mathbf{y}$ and introduce the concept of k-transposition. Observing that if $\mathbf{y}_t^{(2)}$ if a k-transposition of $\mathbf{y}_t^{(1)}$ then $\phi(\mathbf{y}_t^{(2)}) = \alpha^k \phi(\mathbf{y}_t^{(1)})$. The equivariance loss is then defined as follows:

$$\mathcal{L}_{\text{PESTO}} = h\left(\frac{\phi(\mathbf{y}_t^{(1)})}{\phi(\mathbf{y}_t^{(2)})} - \alpha^k\right) \quad (2)$$

SPICE and PESTO follow a similar approach, involving an equivariant loss and a regularization loss. For the regularization SPICE relies on a reconstruction task ($\mathcal{L}_{Recon}$) by adding a decoder directly connected to the scalars $y_t^{(*)}$; while PESTO applies a proposed Shifted Cross Entropy ($\mathcal{L}_{SCE}$) between the two shifted distributions $\mathbf{y}_t^{(*)}$. In the case of pitch estimation, it has been shown in [2] that while PESTO's performance is only slightly affected when the regularization component is removed, SPICE strongly relies on its decoder, not only increasing the number of parameters to be learnt, but also hurting its adaptation to other tasks. Indeed, while a single scalar (such as y_t) can accurately encode the simple harmonic structure of a pitched sound, it cannot encode the one of a complex rhythm pattern.

3. PROPOSAL

To be able to apply SPICE or PESTO for tempo estimation, we replace the CQT of the audio signal by the low-frequency (frequencies are chosen to represent the periodicity of the tempo) of its onset strength function. Figure 1 illustrates our overall pipeline.

3.1. Audio front-end

We use audio sampled at 22050 Hz and estimate its Onset Strength Function (OSF) using the Superflux algorithm [18]. Following standard recommendations, we use a window size of 512 and a hop size of 128. The initial spectrogram is filtered using 24 filters per octave and enhanced with a frequency-axis maximum filtering of size 3. The resulting OSF, o_τ, has a sampling rate of 172Hz.

We then represent the periodicity of o_τ using a CQT, denoted by $\mathbf{x}_t$ We use the nnAudio [19] implementation with a hop size of 128 (which corresponds approximately to 0.7 s). For SPICE we used $Q = 64$ bins per octave and a minimal frequency of 1Hz. For PESTO we used $Q = 48$ bins per octave and a minimum frequency of 0.8Hz, resulting in 320 bins up to the Nyquist-frequency.

3.2. Architecture overview

The SPICE encoder is composed of six 1D convolutional layers, each of them being followed by batch normalization, ReLU activation and a max pooling operation. The output is then flattened in a 1024 embedding $\mathbf{z}_t$ which is projected into a scalar y_t by the tempo head (2 linear layers and a sigmoid output activation). The output of the tempo head y_t is fed to the decoder, whose architecture mirrors the encoder, in order to reconstruct the input slice $\hat{\mathbf{x}}_t \approx \mathbf{x}_t$.

PESTO only has an encoder. Its architecture is designed to preserve transposition. Such property is ensured by wisely setting the kernel size and padding of each layer in order to preserve the frequency resolution along the convolutional layers. The input $\mathbf{x}_t$ is first layer-normalized and preprocessed by two 1D convolutions with skip connections followed by four 1D convolutions after each a non

[1] https://github.com/antoningagnere/sstempo

linear leaky ReLU activation and dropout are applied. The output $\mathbf{z}_t$ is then flattened and fed to a Toeplitz fully connected layer [2] and normalized by a softmax layer to give the vector $\mathbf{y}_t$.

3.3. Training Details

To train SPICE and PESTO we used a batch size of 256 and the ADAM optimizer with an initial learning rate of 10^{-4}. Like in [1] the CQT frames are shuffled so that within a batch they likely come from different tracks. For PESTO, we schedule the learning rate using a cosine. Also, as PESTO employs a gradient-based loss weighting method, adapting its training to the task of tempo estimation gets simplified. In contrast, finding suitable loss weighting parameters for SPICE was challenging, we performed an extensive hyper-parameter search to find the best setting. We set the weights to 0.01 and 1000 for the tempo and reconstruction losses, respectively. For SSL training, we performed 50 epochs.

SPICE and PESTO use two different processes to create the views $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ (the transposed CQT). SPICE samples two offsets from a uniform distribution $k_i \sim \mathcal{U}(k_{min}, k_{max})$ and extracts, from the CQT, slices spanning the range : $[k_i, k_i + F]$. In the case of tempo estimation our hyperparameter search leads us to set $k_{min} = 0$, $k_{max} = 32$ and $F = 256$. PESTO creates two views by only sampling one integer $k \sim \mathcal{U}(-k_{max}, k_{max})$ and cropping the original frame from the range $[k_{max}, K - k_{max} - 1]$ and $[k_{max} - k, K - k_{max} - k - 1]$. We set $k_{max} = 16$.

Upon the completion of the SSL training the model can infer relative tempo estimation. To map its output to absolute tempo values we consider two strategies: (a) calibration using synthetic data (section 3.4), or (b) supervised fine-tuning using real data (section 3.5).

3.4. Calibration using synthetic data

In the first strategy, we perform a calibration step using synthetic data as proposed in [1]. For this, we created a dataset of synthetic signals following the process described in [17]: we generated 300 tracks, each consisting of a 30-second long signal with a periodic click according to tempi drawn randomly in $[60, 240]$ BPM. To prevent any boundary effect, we only consider the central CQT frame of each track. In SPICE [1], they observe a linear relationship between the ground-truth pitch v and SPICE output y, we did not observe such a linearity in the case of tempo. We therefore learn the mapping between y and the tempo v using a simple 1-Nearest-Neighbor classifier: for each synthetic track i of tempo v_i we associate its output y_i. For a given y we then simply look at the closest (according to the Euclidean distance) y_i and return the corresponding v_i. The same process is used for PESTO, but $\mathbf{y}$ is then a vector.

3.5. Supervised fine-tuning using real data

In the second strategy, we fine-tuned the model using real annotated data; hence in a supervised way. Since SPICE's output y_t is a scalar, we strip away the final layer and consider $\mathbf{z}_t$. We then add on top of $\mathbf{z}_t$ a linear classifier with 300 output units (corresponding to tempo classes $\{0, ..., 299\}$ BPM), followed by a softmax. For PESTO, we directly add on top of $\mathbf{y}_t$ a linear classifier with 300 output units and a softmax. To enhance the robustness of our model, we incorporate data augmentation techniques. At the beginning of each epoch, audio excerpts are randomly time-stretched using a factor drawn uniformly from the interval $[0.8, 1.2]$. We only train the linear classifier (the encoder remains frozen) minimizing cross entropy. We used the ADAM optimizer with a learning rate of 10^{-3} during 100 epochs.

4. EXPERIMENTS

4.1. Evaluation metrics

To evaluate the performance of ours models we report the standard Accuracy 1 and Accuracy 2 metrics, applying a $\pm 4\%$ tolerance in line with [20]. While Accuracy 1 assesses the model's capability to pinpoint the exact tempo against the ground truth, Accuracy 2 expands this scope by accounting for common "octave errors", incorporating tempo variations of $\left\{\frac{1}{3}, \frac{1}{2}, 2, 3\right\}$.

Tempo annotations are provided at the track level while our two models output a prediction at each time frame t (one prediction every $\sim$ 0.7s). To derive a single (global) tempo estimation, we choose the tempo prediction that occurs the most frequently over the track duration.

4.2. Datasets

In the following we will use the following datasets either for SSL training, supervised fine-tuning or testing: - FMA-medium [21] (the medium-size subset of the Free Music Archive (FMA) dataset gathering 25.000 excerpts of 30-s duration, without any annotation.) - Hainsworth [22], - GTZAN-rhythm [23], - Giantsteps [24] and - ACM Mirum [25]. The use of these datasets differ according to the three experimental protocols described in the following.

Experiment ①. The goal here is to compare SPICE and PESTO for tempo estimation. Both are trained in a SSL way using FMA-medium and then calibrated using synthetic data. We test them on the individual datasets Hainsworth, GTZAN-rhythm, Giantsteps and ACM -Mirum.

Experiment ②. Here we check if PESTO can be trained with a very small dataset. We train it in a SSL way using in turn one of the individual annotated datasets and test on the same datasets. In all cases, the calibration is performed on synthetic data so that there is no over-fitting (but only the knowledge of the domain of the test data). In any case, the test ground-truth labels are obviously never seen during training.

Experiment ③. The goal of this experiment is to replicate the evaluation protocol proposed by Quinton in [8]: systems are trained in a SSL way on a large dataset (here FMA-medium) and then a supervised fine-tuning is done using a Leave-One-Dataset-Out (LODO) paradigm, *i.e.* all datasets except the one used for the test are used for supervised fine-tuning (*e.g.*, if the test is on Hainsworth, fine-tuning is done using GTZAN-rhythm, Giantsteps and ACM -Mirum). The finetuning protocol is detailed in 3.5.

4.3. Results and discussions

Results of all experiments are given in Table 1. The lowest part presents previously published results using fully-supervised approaches.

Experiment ①. PESTO consistently outperforms SPICE across all test datasets in both Accuracy 1 and 2. Particularly, the performance on ACM Mirum (0.713/ 0.961) demonstrates its capability to adapt and generalize well, even on unseen or diverse musical data. For all datasets, the difference between the two accuracy values is smaller for PESTO than for SPICE. This suggests that PESTO is less prone to octave errors than SPICE. The decoder might play a crucial role being responsible of a high number of octave errors. Our insight is the following: while struggling with the reconstruction task, the decoder might encourage the encoder to fall in the trap of "octave"

Table 1: Performance metrics of our models in different experimental settings in comparison to both SSL and supervised baselines. Higher overall performance for each metric and dataset is indicated with the † symbol. Highest performance for an SSL-based approach is shown in bold. Finally highest performance within an experiment is underlined.

Exp	Method	SSL trained on	Hainsworth		GTZAN		Giantsteps		ACM Mirum	
			Acc 1	Acc 2	Acc 1	Acc 2	Acc 1	Acc 2	Acc 1	Acc 2
SSL training with calibration step										
①	PESTO	FMA	0.658	0.842	0.668	0.903	0.714	0.875	0.713	0.961
	SPICE	FMA	0.509	0.811	0.494	0.865	0.465	0.699	0.436	0.898
②	PESTO	Hainsworth	0.626	0.779	0.593	0.839	0.684	0.797	0.540	0.816
	PESTO	GTZAN	0.595	0.806	0.661	0.873	0.384	0.604	0.622	0.859
	PESTO	Giantsteps	0.568	0.847	0.564	0.904	**0.797**	0.907	0.407	0.891
	PESTO	ACM Mirum	0.540	0.720	0.551	0.819	0.602	0.718	0.668	0.810
SSL training with supervised finetuning using LODO										
③	PESTO	FMA	0.743	**0.896**†	0.738	**0.942**	0.706	**0.958**	0.668	**0.971**
	SPICE	FMA	**0.784**	0.883	**0.745**	0.887	0.730	0.898	0.632	0.933
	Quinton	MTT	0.518	0.856	0.741	0.919	0.470	0.886	**0.747**	0.965
Supervised baselines										
	Schreiber [10]	-	0.770	0.842	0.694	0.926	0.730	0.893	0.795	0.974
	Foroughmand [14]	-	0.734	0.829	0.697	0.891	0.836	0.979†	0.733	0.965
	Böck 1 [9]	-	0.806†	0.892	0.697	0.950	0.589	0.864	0.741	0.976
	Böck 2 [13]	-			0.830†	0.950†	0.870†	0.965	0.841†	0.990†

errors to facilitate the reconstruction task. In such situation it can focus on reconstructing only a sub range of the seen tempo.

Experiment ②. We only indicate here the results for the PESTO model since the SPICE model did not perform well under this condition. This may be due to the large number of parameters of SPICE (2.3M) which prevents its training using only small datasets (Hainsworth has only 220 tracks). Providing few but diverse tracks seems to greatly harm SPICE's ability to learn meaningful representations. We did not explore if adding data augmentation could help but we do not expect it to radically change the outcome. On the contrary, the small number of parameters of PESTO (70k) allows training it with such small datasets. Interestingly, PESTO leads to a higher Accuracy 1 (0.797) when trained in a SSL way using Giantsteps (664 tracks) than when using FMA (0.714) or the LODO fine-tuning (0.706). Training using only Giantsteps also produces the highest Accuracy 2 for all the test-set (*i.e.* to recognize tempi of Hainsworth it is better to perform the SSL training on Giantsetps than on Hainsworth). However, for all datasets, it leads to an Accuracy 1 lower than in any other scenarios considered in this experiment. Giantsteps is composed of electronic dance music, which often has more pronounced beat patterns, as opposed to the other datasets which contain more diversity of genres. As PESTO performs very well in such setting, this suggests to explore a special use case, in which a user can pre-train PESTO solely relying on their specific music library without necessarily trying to generalize to a very large music collection.

Experiment ③. In terms of Accuracy 2, PESTO outperforms SPICE and Quinton's SSL approach for all datasets. For Accuracy 2, the supervised fine-tuning on real data also outperforms the calibration using synthetic data. This is not the case when considering Accuracy 1. Compared to calibration, the supervised fine-tuning only increases Accuracy 1 for Hainsworth and GTZAN (but decreases it for Giansteps and ACM Mirum). In contrast, supervised fine-tuning is beneficial to SPICE for all datasets. Both PESTO and SPICE models outperform Quinton's SSL approach [8] in all metrics and datasets, except for Accuracy 1 on ACM Mirum.

We also explored adding the Extended Ballroom [26] dataset for fine-tuning, i.e. using LODO + Extended Ballroom. For PESTO, this lead to an increase of Accuracy 1 for Giansteps (from 0.706 to 0.831) and for ACM Mirum (from 0.668 to 0.725). To remain fair with Quinton [8] protocol, we did not report these results in Table 1.

Benchmarking vs. supervised baselines. Overall, the method that performs the best is the one proposed by [13], which obtained the highest score for all datasets except Hainsworth, on which it was not tested, and Accuracy 2 on Giantsteps. On Hainsworth, PESTO surpassed all the other supervised systems in Accuracy 2 (0.896). Except for Accuracy 1 on ACM Mirum, our models surpassed at least one of the supervised baselines. This observation remains true concerning PESTO solely relying on a calibration step, highlighting the potential of SSL applied to tempo estimation.

5. CONCLUSION

In this work, we studied the adaptation of two successfull pitch-based SSL models, SPICE and PESTO, for the task of tempo estimation. In a large-scale evaluation, we demonstrate that these adapted models outperform previously proposed tempo-based SSL models, and reach Accuracy 1 and 2 close to fully supervised approaches. Among the two models, PESTO leads to the best Accuracy 2 for all datasets while SPICE does for Accuracy 1. We also demonstrate that supervised fine-tuning of the models leads to superior performance than calibration on synthetic data. Finally, we propose a new paradigm in which the model is trained in a SSL way directly on the test-data which constitutes a sort of zero-shot learning in a domain adaptated fashion. For the Giantsteps dataset this paradigm leads to the best Accuracy 1 overall. Future works will therefore concentrate on studying more in depth and extending this paradigm.

6. REFERENCES

[1] Beat Gfeller, Christian Havnø Frank, Dominik Roblek, Matthew Sharifi, Marco Tagliasacchi, and Mihajlo Velimirovic, "SPICE: Self-supervised pitch estimation," *Audio, Speech and Language Processing, IEEE Transactions on*, vol. 28, pp. 1118–1128, 2020.

[2] Alain Riou, Stefan Lattner, Gaëtan Hadjeres, and Geoffroy Peeters, "PESTO: Pitch estimation with self-supervised transposition-equivariant objective," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Milan, Italy, 2023.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. of ICML (International Conference on Machine Learning)*, 2020.

[4] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko, "Bootstrap your own latent - A new approach to self-supervised learning," in *Proc. of NeurIPS (Conference on Neural Information Processing Systems)*, 2020.

[5] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *Proc. of ICML (International Conference on Machine Learning)*, 2021.

[6] Xinlei Chen and Kaiming He, "Exploring simple siamese representation learning," in *Proc. of IEEE CVPR (Conference on Computer Vision and Pattern Recognition)*, 2021.

[7] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. of IEEE CVPR (Conference on Computer Vision and Pattern Recognition)*, 2020.

[8] Elio Quinton, "Equivariant self-supervision for musical tempo estimation," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Bengaluru, India, Dec. 2022.

[9] Sebastian Böck, Florian Krebs, and Gerhard Widmer, "Accurate tempo estimation based on recurrent neural networks and resonating comb filters," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Malaga, Spain, 2015.

[10] Hendrik Schreiber and Meinard Müller, "A single-step approach to musical tempo estimation using a convolutional neural network," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Paris, France, 2018.

[11] Matthew E. P. Davies and Sebastian Böck, "Temporal convolutional networks for musical audio beat tracking," in *2019 27th European Signal Processing Conference (EUSIPCO)*, Coruña, Spain, 2019.

[12] Sebastian Böck, Matthew E. P. Davies, and Peter Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Delft, Netherlands, 2019.

[13] Sebastian Böck and Matthew E. P. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Virtual Conference, 2020.

[14] Hadrien Foroughmand and Geoffroy Peeters, "Deep-Rhythm for Tempo Estimation and Rhythm Pattern Recognition," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Delft, Netherlands, 2019.

[15] Geoffroy Peeters, "Template-based estimation of tempo: using unsupervised or supervised learning to create better spectral templates," in *Proc. of DAFX*, Graz, Austria, 2010.

[16] Geoffroy Peeters, "Template-based estimation of time-varying tempo," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–14, 2007.

[17] Giovana Morais, Matthew E. P. Davies, Marcelo Queiroz, and Magdalena Fuentes, "Tempo vs. pitch: Understanding self-supervised tempo estimation," in *Proc. of IEEE ICASSP (International Conference on Acoustics, Speech, and Signal Processing)*, Rhodes Island, Greece, 2023.

[18] Sebastian Böck and Widmer Gerhard, "Maximum filter vibrato suppression for onset detection," in *Proc. of DAFx (International Conference on Digital Audio Effects)*, Maynooth, Ireland, 2013.

[19] K. W. Cheuk, H. Anderson, K. Agres, and D. Herremans, "nnaudio: An on-the-fly gpu audio to spectrogram conversion toolbox using 1d convolutional neural networks," *IEEE Access*, vol. 8, pp. 161981–162003, 2020.

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

[21] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson, "FMA: A dataset for music analysis," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Suzhou, China, 2017.

[22] Stephen W. Hainsworth and Malcolm D. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP J. Adv. Signal Process.*, vol. 2004, no. 15, pp. 2385–2395, 2004.

[23] Ugo Marchand and Geoffroy Peeters, "GTZAN-rhythm: Extending the GTZAN test-set with beat, downbeat and swing annotations," in *Late-Breaking/Demo Session of ISMIR (International Society for Music Information Retrieval)*, Malaga, Spain, 2015.

[24] Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections," in *Proc. of ISMIR (International Society for Music Information Retrieval)*, Malaga, Spain, 2015.

[25] Geoffroy Peeters and Joachim Flocon-Cholet, "Perceptual tempo estimation using gmm-regression," in *ACM Multimedia, MIRUM workshop*, Nara, Japan, 2012.

[26] Ugo Marchand and Geoffroy Peeters, "The extended ballroom dataset," in *Late-Breaking/Demo Session of ISMIR (International Society for Music Information Retrieval)*, New York, USA, 2016.