a compact set of input controls. It biases a DNN towards generating and processing audio by the insertion of signal processing elements, such as oscillators and filters, in the network structure. These are implemented using differentiable operators from a neural network training framework and therefore can back-propagate gradients during training.

DDSP methods have been successfully employed in resynthesis and tone transfer [16] tasks, where they are conditioned on a set of time-evolving inputs, including the fundamental frequency of the target signal, and generate audio on a frame-by-frame basis. Some approaches such as the Neural Source Filter (NSF) [14, 35] or the Neural Waveshaper [13] learn to control a non-linear filter that shapes a harmonic source towards a particular target sound. Other DDSP architectures [12, 17] directly drive a Harmonic plus Noise (HpN) synthesizer [36], effectively learning a mapping between the input controls and the synthesizer parameters that generate the output.

While the previous resynthesis architectures can generate realistic tone transfer, there is little users can do to manipulate the resulting audio other than controlling the pitch and loudness inputs. The DNN controls spectral modelling algorithms, which do not have musically meaningful parameters. In this work, we present a differentiable FM synthesizer module that features a compact set of well-known sound design controls driven by a DNN, enabling a potential user intervention into the synthesis process.

While the idea of differentiable FM is not new, we have not been able to find published literature with details and evaluations of systems employing it. We highlight two web repositories, one involving a 2-oscillator FM optimization strategy of audio excerpts [37] and another presenting an extension of the original DDSP project with a Differentiable FM synthesizer [38], where the experiments fail to reproduce musical instrument sounds. Our approach imposes a set of constraints on the FM synth that allows a DNN to generate instrument sounds.

2.6 Training objective for DDSP resynthesis

The training process for the DDSP algorithms usually involves a multi-scale spectral (MSS) loss function that includes the L_1 distance of the amplitude spectrograms of a synthesized and a target audio excerpt, in linear and logarithmic form [12]. This function is used as a reconstruction loss, with the DDSP model aiming to replicate the target spectra during training.

Despite its widespread use, the MSS loss and more generally, spectral-based distance metrics present pitch-based failure modes that can conspire against the generalization capabilities of NAS algorithms when trained with gradient descent, as demonstrated by Turian et. al. [39]. The authors show how such functions fail to propagate informative gradients for fine-tuning oscillator frequencies towards a frequency target due to fine grained ripple on the loss surface. Furthermore, they indicate that for the MSS distance, jointly optimizing amplitude and frequency generates misleading gradients for both tasks; this loss function can match spectral amplitudes only when the harmonics of target and prediction are aligned in frequency.

We argue that the MSS loss works well for DDSP because of two main reasons: Firstly, these models drive highly parameterized spectral modelling synthesizers with fine control on the output, either in the form of filter-distortion DNNs [14, 35], multiple parallel waveshapers [13], or the HpN synthesizer [12]. Secondly, and most importantly, all DDSP resynthesis architectures require as conditioning the fundamental frequency from the target signal, extracted with an estimator [40]. This ensures a harmonic alignment between the target and the prediction, and effectively avoids the problem of having to optimize pitch using gradient descent and the MSS loss during training.

A loss function that cannot propagate informative gradients cannot be employed to train a DNN. This is particularly disadvantageous for the case of FM generation, where the synthesis parameters include the modulation index I and the frequency ratios r that determine distance between the side-bands and the carrier. A small mismatch on ratio estimation can generate an unwanted vibrato-like effect, due to small frequency differences between oscillators. A big mismatch can hinder the joint optimization of the harmonics' positions and amplitudes. A DNN that does not learn how to precisely control the ratios could very easily incur either of those problems.

3. METHOD

In this section, we propose a set of constraints that ease training of a DNN with a differentiable FM synthesizer. Next, we present DDX7, a NAS architecture for FM synthesis controlled by a Temporal Convolutional Network (TCN) [41]. We train the DDX7 model for FM resynthesis of musical instrument sounds. This effectively results in a DX7 patch that is playable by an arbitrary audio input. A diagram of the architecture is shown in Figure 1.

3.1 Differentiable DX7

Our aim is to provide continuous control possibilities for a well-known FM synthesizer. We choose the Yamaha DX7 for its lasting influence on musical practice.

Considering the DX7 patch design, we observe that when fixing the frequency ratios and the routing of the oscillators, all the harmonics and overtones that can be generated take a fixed position in the spectrum. This *patch constraint* can allow the synthesis model to propagate informative gradients from the MSS reconstruction loss, provided the FM partials are pre-aligned with the ones of the target audio by means of pitch conditioning, as in the case of DDSP resynthesis.

We propose a control scheme for an FM synthesizer where a DNN controls the modulation indices and volume of the oscillators. The oscillator routing and frequency ratios remain fixed. One problem may arise when considering the maximum values that the modulation index I can take. For different ranges of I, the Bessel functions can create local minima during spectral optimization due to

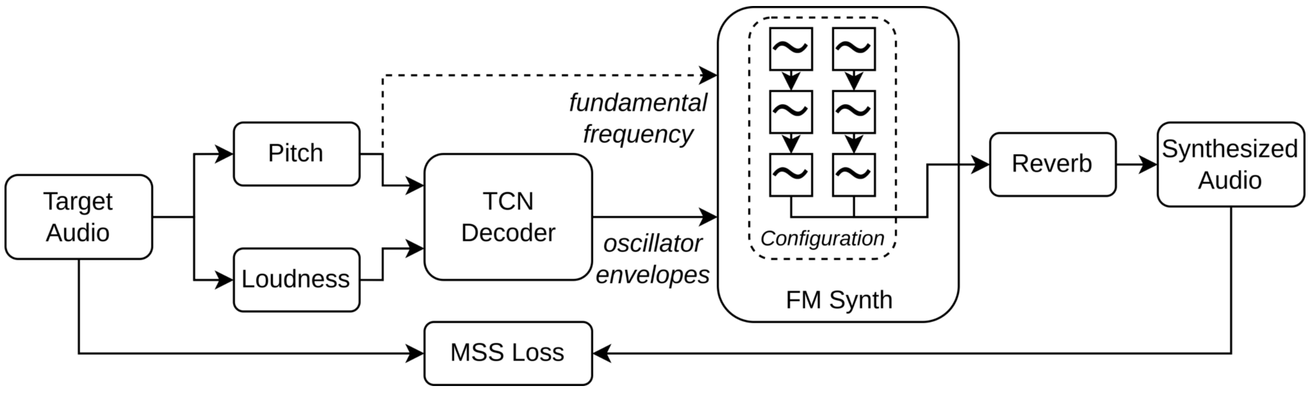

Figure 1. The DDX7 architecture employs a TCN decoder conditioned on a sequence of pitch and loudness frames to drive the envelopes of a few-oscillator differentiable FM synthesizer that features a fixed FM configuration with fixed frequency ratios, effectively mapping continuous controls of pitched musical instruments to a well-known synthesis architecture.

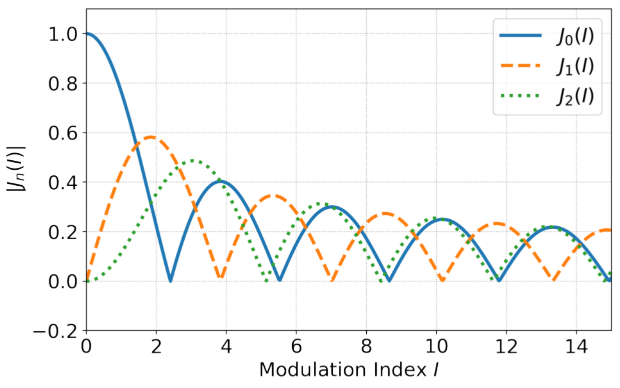

Figure 2. Absolute amplitudes of the first three harmonics generated by FM in function of I.

their oscillatory nature. In the DX7, the modulation index envelopes can take values of as much as 4π [42], but only for $I < 1.83$, are the Bessel functions strictly monotonic, with the carrier just exchanging energy with the sidebands, as shown in Figure 2. We analyze the effect of the maximum ranges through experiments in Section 5.

We implement the FM modulation algorithm in Pytorch, and adapt it manually for the different *FM configurations* that are tried in this work. Each configuration defines a fixed oscillator routing and a set of frequency ratios.

3.2 TCN Decoder

TCNs have been successfully employed a number of sequential modelling tasks, including for audio processing and generation [14, 27, 43]. These are fully-convolutional networks that employ 1-dimensional convolutions and exponentially growing dilations to efficiently model long sequences within their receptive field [41]. We choose TCNs for our DDX7 implementation for their fast training capabilities and good sequential modelling performance.

In our FM resynthesis problem, we aim to map a set of synchronous input sequences of pitch $f_{0_1}, ..., f_{0_T} \in \mathbb{R}$ and loudness $ld_1, ..., ld_T \in \mathbb{R}$ to the controls of our constrained synthesizer, i.e. the output levels $ol_1, ..., ol_T \in \mathbb{R}^6$ of the six oscillators. We define the parameterized mapping function f_θ as shown in Equation 3, where the conditioning sequence $c_1, ..., c_T \in \mathbb{R}^2$ is obtained by concatenating along a new dimension both pitch and loudness sequences, $\sigma(.)$ is the sigmoid activation function and A_{max} is the maximum output level value that the envelopes can take for that oscillator, as shown in Equation 4, where I_{max} is a hyperparameter describing the maximum modulation index range that the system can realize.

$$\hat{ol_1}, ...\hat{ol_T} = A_{max} \cdot \sigma(f_\theta(c_1, ...c_T)) \tag{3}$$

$$A_{max} = \begin{cases} 1 & \text{if carrier} \\ I_{max} & \text{otherwise} \end{cases} \tag{4}$$

We implement our mapping function f_θ with a simple, causal TCN architecture following [41], with 2 input and 6 output channels, processed by 5 TCN residual blocks with skip connections and 128 hidden channels each. Each residual block features two convolutional layers of kernel size 3, and the dilation increases by a factor of 2 in each block. Weight normalization, dropout with probability of 0.5 and ReLU activation functions are used throughout the network, with the exception of the output layer that features a sigmoid layer. This yields a relatively lightweight decoder, with about 400k parameters, and a receptive field T of 125 pitch and loudness frames.

3.3 Learnable Reverb

We employ a differentiable reverb module, similar to the one employed for the DDSP decoder [12], featuring learnable mix and decay parameters, and a trainable impulse response of 1 second length. This is used to estimate the room response of the dataset recordings, decoupling it from the FM sound generation block. It is applied directly to the FM synthesizer output and it is jointly optimized with the DNN during training.

4. TRAINING

4.1 Loss function

The DDX7 architecture can be trained with a corpus of audio from a musical instrument as supervision, employing stochastic gradient descent on minibatches with a spectral reconstruction objective. We employ the MSS reconstruction loss shown in Equation 5, where S_i and $\hat{S}_i$ are the magnitude spectrograms of the target and synthesized audio respectively, $||.||_1$ denotes the L_1 norm, and i is a particular Fourier transform analysis window on which the spectrogram is computed. We use $i \in \{64, 128, 256, 512, 1024, 2048\}$ with an overlap of 75% between windows.

$$L = \sum_i (||S_i - \hat{S}_i||_1 + ||logS_i - log\hat{S}_i||_1) \qquad (5)$$

4.2 Dataset

We train our DDX7 models on audio samples of instruments extracted from the University of Rochester Music Performance (URMP) dataset [44], an audio-visual dataset that contains classical pieces. We select the separated audio stems for violin, flute and trumpet as our training data, down-sample them to 16 kHz, remove silences and crop the audio files to instances of 4 seconds. We extract the A-weighted loudness [45] and fundamental frequency employing the CREPE [40] pitch estimator. We discard all instances that yield a mean pitch confidence smaller than 0.85, with the exception of the flute corpus, for which we relax the requirement down to 0.80 due to its short length. We further normalize pitch and loudness values within a range between 0 and 1.

We process each annotation with a hop size of 64 samples, yielding pitch and loudness sequences of 1000 frames for each 4-second instance. The selection of hop size and sample rate results in our TCN model featuring a receptive field of 0.5 s, and dictates the frame rate at which it drives the oscillators, 250 Hz. We linearly interpolate the envelope frames before feeding them into the oscillators. Finally, we separate the dataset into train, validation and test sets with 0.75 / 0.125 / 0.125 splits respectively.

4.3 FM configurations

For each target instrument, we select a different FM configuration extracted from the original patch set of the Yamaha DX7, which we retrieve from the web. [2] Then, we load the patches in Dexed [46], a DX7 emulator, and audit them searching for most similar to the target instruments. We select "STRINGS 1" for violin, "FLUTE 1" for flute and "BRASS 3" for the trumpet. We deploy the oscillator routing with the frequency ratios rounded up to one decimal point (to avoid vibrato-like effects) as differentiable FM modules, as shown in Figure 3. We do not deploy the oscillator feedback feature of the DX7 in our implementation, as it cannot be computed in parallel and we find it is very slow to render using a standard for loop in Python.

[2] `http://bobbyblues.recup.ch/yamaha_dx7/`

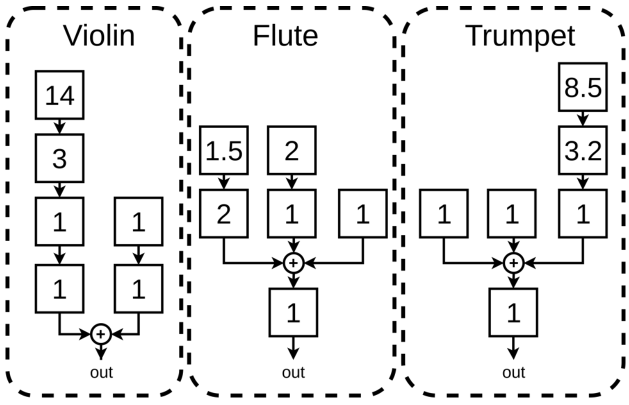

Figure 3. FM configurations used for training. Squares indicate sinusoidal oscillators and their frequency ratios.

4.4 Training process

Our models are trained for 120k steps, with the Adam optimizer, set with an initial learning rate of 3e-4, decreasing with a factor of 0.98 each 10k steps. We clip gradients to a maximum norm of 2, and employ a batch size of 16.

5. EVALUATION

We evaluate the performance of DDX7 on the resynthesis task, training the model on the flute, trumpet and violin corpus, and evaluate its performance on selected benchmarks with the Fréchet Audio Distance (FAD).

5.1 Fréchet audio distance

The Fréchet Audio Distance (FAD) presented by Kilgour et. al. [47] serves as a quality metric for audio enhancement, that correlates better with human listeners than SDR (signal-to-distortion ratio) or spectral differences such as the MSS loss. This is in line with other deep neural features used in Computer Vision that are found to outperform heuristic metrics by great margins [48]. It has been used to assess synthesis quality in previous works [13,29,31]. The FAD computes the Fréchet distance between multivariate Gaussian distributions inferred from the embeddings of a pre-trained VGGish model [49]. The compared distributions are generated from the embeddings of a corpus of audio for evaluation and a "background" corpus of high quality audio as reference.

5.2 Benchmarks

5.2.1 Maximum modulation index

We are interested into knowing which are the best modulation index limits for which our model can successfully render the instrument audio. Taking into account that optimizing for a particular spectra may be difficult for the original maximum modulation index range of the DX7 of 4π, we train our model for each instrument with three different maximum modulation index ranges for the oscillators: $I_{max} = \{4\pi, 2\pi, 2\}$, including the original DX7 range, a halved one and one limited at two, which includes the

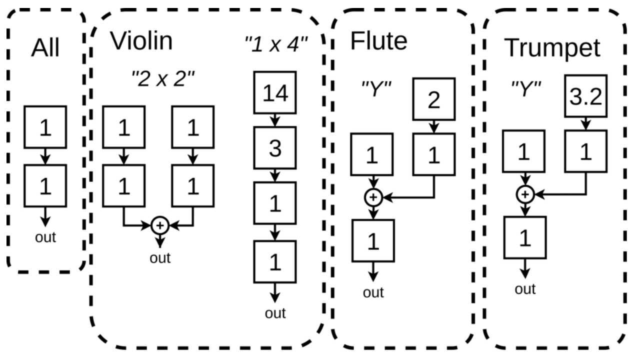

Figure 4. Ablated FM configurations.

global maximum of the first harmonic $J_1(I)$, but limits the other Bessel functions to a space where they are strictly monotonic. Finally, we load the corresponding configuration for each instrument selected, and we train three versions of DDX7, to account for each of the values of I_{max}.

Our FM model learns to control the envelopes of 6 oscillators frame-wise. As a baseline, we assess the performance of the common Harmonic plus Noise spectral modelling synthesizer employed for resynthesis tasks. We train a baseline pitch and loudness decoder that controls 121 frame-wise parameters of an HpN synthesizer, similar to the original solo instrument DDSP Decoder [12], but implemented in the same training framework as DDX7, to ensure that we use exactly the same dataset, preprocessing and frame rate. The model is roughly 11 times bigger than DDX7, with about 4.5 M parameters. We train it for 120k steps for parity with our model, with a batch size of 16, and an initial learning rate of 1e-4, decreasing to a factor of 0.98 for each 10k steps.

For evaluation with FAD, we generate background embedding distributions of the complete audio corpus of each instrument. Next, we generate embeddings distributions from the resynthesized excerpts of the test set for each model and instrument. We compute the FAD of these and the original test set against their corresponding background distribution. Results are shown in Table 1. We observe that the HpN baseline outperforms our model in violin and trumpet, but controlling 121 parameters instead of 6, and at a higher computational cost. Surprisingly, DDX7 outperforms the baseline on flute, suggesting that our *patch constraint* approach can bias a DNN towards a usable FM timbral space. We observe the best DDX7 performance is obtained with $I_{max} = 2$ for flute and violin and $I_{max} = 2\pi$ for trumpet. This may be correlated with the different spectra of the instruments, suggesting that the trumpet may require a bigger I_{max} for an improved reconstruction. Finally, for some configurations of I_{max} the models sound unnatural and fail at the estimation of the room response. We leave further analysis of this issue for future work.

5.2.2 *Oscillator ablation test*

We want to assess if our model is effectively optimized to leverage all of the modulators for the reconstruction task. We propose ablated versions of the previous patches with two and four oscillators, as shown in Figure 4. We train the DDX7 models on the ablated patches using the optimal I_{max} found with the previous benchmark and compare their resynthesis quality with the FAD following the same previous procedure. The results shown in Table 2 suggest that the violin and flute model benefit from the extra degrees of freedom present with more oscillators. On the other hand, the trumpet model works best with the smallest configuration, possibly due to an incorrect patch selection that hindered the optimization process. Finally, the 2-oscillator trumpet model outperforms the HpN baseline, suggesting that good results can be achieved with a small number of frequency-modulated oscillators.

	Fréchet Audio Distance		
Model	*Flute*	*Violin*	*Trumpet*
Test Data	2.074	0.577	1.069
HpN Baseline	4.326	**0.795**	**2.486**
DDX7 ($I_{max} = 2$)	<u>**2.731**</u>	<u>1.618</u>	4.941
DDX7 ($I_{max} = 2\pi$)	3.281	2.148	<u>3.326</u>
DDX7 ($I_{max} = 4\pi$)	2.938	1.637	3.853

Table 1. FAD of resynthesis results for all models computed against the background embedding distributions for each instrument complete corpus. Best results are in bold and best I_{max} configurations are underlined.

Instrument	**FM Configuration**				
	6	*4 "Y"*	*4x1*	*2x2*	*2*
Flute	**2.731**	3.246	-	-	3.364
Violin	**1.618**	-	1.877	5.620	8.270
Trumpet	3.326	2.943	-	-	**1.674**

Table 2. FAD for complete and ablated patches on DDX7.

6. CONCLUSION

We presented DDX7, an approach for FM resynthesis of musical instrument sounds that yields good reconstructions controlling few parameters, with relatively smaller models. We have shown that FM with a *patch constraint* can perform comparably well to a more complex baseline with just 6, and even less oscillators; we hope this motivates further research along this line, including for instance sound matching techniques to find suitable configurations.

Current resynthesis architectures feature synthesizers that are difficult to intervene in a musically meaningful way. In contrast, DDX7 learns to control an FM synthesizer that is common in the sound design practice. It replaces the ADSR generator of the original DX7 with a TCN that infers the envelopes from continuous control inputs. At runtime, it is possible to manipulate the timbre on-the-fly, either by re-shaping the spectrum with the ratios, altering dynamics on the envelopes, or by re-routing the oscillators. Finally, the small model size and causal temporal dependency make DDX7 an interesting candidate for real-time implementation. We leave an exploration of these affordances and possibilities for future work.

7. ACKNOWLEDGEMENTS

We would like to thank the ISMIR reviewers for their valuable feedback. Also, we would like to thank our colleagues Ben Hayes and Rodrigo Diaz for their advice and many compelling discussions about audio rendering and DNN optimization. This work was supported by UK Research and Innovation [grant number EP/S022694/1]. AM's contributions are supported by the Royal Academy of Engineering under the Research Chairs and Senior Research Fellowships scheme.

8. REFERENCES

[1] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Journal of the Audio Engineering Society*, vol. 21, no. 7, pp. 526–534, 1973.

[2] M. Lavengood, "What makes it sound '80s? The Yamaha DX7 electric piano sound," *Journal of Popular Music Studies*, vol. 31, no. 3, pp. 73–94, 2019.

[3] E. Miranda, *Computer Sound Design: Synthesis techniques and programming*. Routledge, 2012.

[4] B. Stevens, *Teaching Electronic Music: Cultural, Creative, and Analytical Perspectives*. Routledge, 2021.

[5] T. Pinch and F. Trocco, "The social construction of the early electronic music synthesizer," *ICON*, pp. 9–31, 1998.

[6] A. R. Jensenius and M. J. Lyons, *A NIME reader: Fifteen years of New Interfaces for Musical Expression*. Springer, 2017, vol. 3.

[7] T. West, B. Caramiaux, S. Huot, and M. M. Wanderley, "Making mappings: Design criteria for live performance," *New Interfaces for Musical Expression conference (NIME)*, 5 2021.

[8] J. Regimbal and M. M. Wanderley, "Interpolating audio and haptic control spaces," in *New Interfaces for Musical Expression conference (NIME)*. PubPub, 2021.

[9] C. Poepel and R. B. Dannenberg, "Audio Signal Driven Sound Synthesis," in *International Computer Music Conference*, 2005.

[10] V. Verfaille, U. Zolzer, and D. Arfib, "Adaptive digital audio effects (a-dafx): a new class of sound transformations," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1817–1831, 2006.

[11] V. Lazzarini, J. Timoney, and T. Lysaght, "Adaptive FM Synthesis," in *DAFX-07 the 10th Int. Conference on Digital Audio Effects*, September 2007.

[12] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[13] B. Hayes, C. Saitis, and G. Fazekas, "Neural waveshaping synthesis," *Proceedings of the 22th International Society for Music Information Retrieval Conference*, 2021.

[14] M. Michelashvili and L. Wolf, "Hierarchical timbre-painting and articulation generation," *Proceedings of the 21th International Society for Music Information Retrieval Conference*, 2020.

[15] O. Cifka, A. Ozerov, U. Şimşekli, and G. Richard, "Self-Supervised VQ-VAE for One-Shot Music Style Transfer," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 96–100.

[16] M. Carney, C. Li, E. Toh, P. Yu, and J. Engel, "Tone transfer: In-browser interactive neural audio synthesis," in *Joint Proceedings of the ACM IUI 2021 Workshops*, 2021.

[17] Y. Wu, E. Manilow, Y. Deng, R. Swavely, K. Kastner, T. Cooijmans, A. Courville, C.-Z. A. Huang, and J. Engel, "MIDI-DDSP: Detailed control of musical performance via hierarchical modeling," *International Conference on Learning Representations (ICLR) 2022*, 2022.

[18] M. Yee-King and L. McCallum, "Studio report: Sound synthesis with DDSP and network bending techniques," *Proceedings of the 2nd Conference on AI Music Creativity*, 2021.

[19] K. Nielsen, "Practical linear and exponential frequency modulation for digital music synthesis," *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20), Vienna, Austria, September 8–12, 2020-21*, 2020.

[20] N. Masuda and D. Saito, "Quality diversity for synthesizer sound matching," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx20in21)*, 2021.

[21] Sound On Sound Magazine. (2020) Korg Opsix. [Online]. Available: https://www.soundonsound.com/reviews/korg-opsix

[22] Max Kuehn, for Fidlar Music. (2022) Best FM Synth 2022. [Online]. Available: https://fidlarmusic.com/best-fm-synth/

[23] A. Horner, J. Beauchamp, and L. Haken, "Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis," *Computer Music Journal*, vol. 17, no. 4, pp. 17–29, 1993.

[24] M. J. Yee-King, L. Fedden, and M. d'Inverno, "Automatic Programming of VST Sound Synthesizers Using Deep Networks and Other Techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, Apr. 2018.

[25] G. Le Vaillant, T. Dutoit, and S. Dekeyser, "Improving synthesizer programming from variational autoencoders latent space," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx20in21)*, 2021.

[26] Z. Chen, Y. Jing, S. Yuan, Y. Xu, J. Wu, and H. Zhao, "Sound2Synth: Interpreting sound via FM synthesizer parameters estimation," *arXiv preprint arXiv:2205.03043*, 2022.

[27] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *The 9th ISCA Speech Synthesis Workshop*, 2016.

[28] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model," in *5th International Conference on Learning Representations*, Toulon, France, 2017.

[29] J. Nistal, S. Lattner, and G. Richard, "DrumGAN: Synthesis of Drum Sounds With Timbral Feature Conditioning Using Generative Adversarial Networks," in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, Montréal, Aug. 2020.

[30] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial Neural Audio Synthesis," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 17.

[31] A. Lavault, A. Roebel, and M. Voiry, "StyleWaveGAN: Style-based synthesis of drum sounds with extensive controls using generative adversarial networks," *arXiv preprint arXiv:2204.00907*, 2022.

[32] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows," *Applied Sciences*, vol. 10, no. 1, p. 302, 2020.

[33] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[34] S. Huang, Q. Li, C. Anil, S. Oore, and R. B. Grosse, "TimbreTron A WaveNet(CycleGAN(CQT(Audio))) Pipeline for Musical Timbre Transfer," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 17.

[35] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter waveform models for statistical parametric speech synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 402–415, 2019.

[36] Y. Stylianou, J. Laroche, and E. Moulines, "High-quality speech modification based on a harmonic+ noise model," in *Fourth European Conference on Speech Communication and Technology*, 1995.

[37] A. Jansson, "Implicit neural differentiable FM synthesizer," https://github.com/andreasjansson/fmsynth, 2022.

[38] J. Alonso, "DDSP-FM: differentiable FM synthesis," https://juanalonso.github.io/ddsp_fm/, 2021.

[39] J. Turian and M. Henry, "I'm Sorry for Your Loss: Spectrally-Based Audio Distances Are Bad at Pitch," *arXiv:2012.04572 [cs, eess]*, Dec. 2020, I Can't Believe It's Not Better! (ICBINB) NeurIPS 2020 Workshop.

[40] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A Convolutional Representation for Pitch Estimation," in *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2018 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., Sep. 2018, pp. 161–165.

[41] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv:1803.01271 [cs]*, Apr. 2018, arXiv: 1803.01271.

[42] D. Bristow and J. Chowning, "FM Theory and Applications: By Musicians for Musicians," *Yamaha Music Foundation*, 1986.

[43] C. J. Steinmetz and J. D. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *152nd AES Convention*, 2022.

[44] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a Multitrack Classical Music Performance Dataset for Multimodal Music Analysis: Challenges, Insights, and Applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, Feb. 2019.

[45] B. C. Moore, B. R. Glasberg, and T. Baer, "A model for the prediction of thresholds, loudness, and partial loudness," *Journal of the Audio Engineering Society*, vol. 45, no. 4, pp. 224–240, 1997.

[46] P. Gauthier, "Dexed - FM Plugin Synth," https://github.com/asb2m10/dexed, 2022.

[47] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms," in *Interspeech 2019*. ISCA, Sep. 2019, pp. 2350–2354.

[48] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, Jun. 2018, pp. 586–595.

[49] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, “CNN architectures for large-scale audio classification,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA: IEEE, Mar. 2017, pp. 131–135.

SINGING BEAT TRACKING WITH SELF-SUPERVISED FRONT-END AND LINEAR TRANSFORMERS

Mojtaba Heydari **Zhiyao Duan**
Department of Electrical and Computer Engineering
University of Rochester, 500 Wilson Blvd, Rochester, NY 14627, USA
mheydari@ur.rochester.edu zhiyao.duan@rochester.edu

ABSTRACT

Tracking beats of singing voices without the presence of musical accompaniment can find many applications in music production, automatic song arrangement, and social media interaction. Its main challenge is the lack of strong rhythmic and harmonic patterns that are important for music rhythmic analysis in general. Even for human listeners, this can be a challenging task. As a result, existing music beat tracking systems fail to deliver satisfactory performance on singing voices. In this paper, we propose singing beat tracking as a novel task, and propose the first approach to solving this task. Our approach leverages semantic information of singing voices by employing pre-trained self-supervised WavLM and DistilHuBERT speech representations as the front-end and uses a self-attention encoder layer to predict beats. To train and test the system, we obtain separated singing voices and their beat annotations using source separation and beat tracking on complete songs, followed by manual corrections. Experiments on the 741 separated vocal tracks of the GTZAN dataset show that the proposed system outperforms several state-of-the-art music beat tracking methods by a large margin in terms of beat tracking accuracy. Ablation studies also confirm the advantages of pre-trained self-supervised speech representations over generic spectral features.

1. INTRODUCTION

Music tempo and beat detection are two of the core and well-defined MIR topics, and scholars have proposed many approaches to addressing different aspects of them for various music genres. For instance, some works such as [1–4] proposed some unsupervised approaches to detect music beat and tempo by using some low-level features like onset strengths. More recently, several more recent approaches employ neural networks to address the mentioned tasks [5–8]. Extracting singing rhythmic parameters makes it possible to address several MIR problems such as automatic instrumental and singing tracks alignment, automatic music mix, and remix, interactive content creation on social media platforms, etc.

Many of the proposed music rhythmic analysis approaches [1, 5, 7] perform in an offline fashion while others [6, 9, 10] are capable of extracting music rhythmic features causally and even in real-time.

In contrast to all of the mentioned models, beat and tempo detection for singing voice is an untapped MIR task. There are significant inherent differences between the natures of complete music and singing voices. One of those differences is that complete music pieces usually contain rich percussive and harmonic profiles while singing tracks usually lack such beneficial parameters making their rhythmic analysis more demanding compared to the former group. Their other important difference is that music beat tracking models usually only use acoustical clues such as magnitude spectrogram as input features while singing tracks are more similar to speech signals where the models may require to deal with para-linguistics, semantic, and phonemic level inputs in addition to acoustical features.

Our contributions in this paper are as follows:

1- We introduce singing beat tracking as a novel MIR task. We propose two strategies to tackle the lack of annotated data for this task by leveraging pre-existing datasets and beat tracking and source separation techniques. We also propose a new evaluation scheme that can account for phase ambiguities of beat annotation for vocal music.

2- We propose two neural models for the singing beat tracking task. These models leverage pre-trained speech self-supervised models to extract feature embeddings, which are then fed into a linear transformer network to output beat predictions.

3- We evaluate the proposed models on hundreds of vocal tracks with diverse genres. Experiments show that the proposed models outperform three representative baselines designed or trained for general music beat tracking by a large margin. An ablation study is also performed to investigate the effect of speech self-supervised models over commonly used generic spectral features, and results again show an outperformance by a large margin.

The remainder of the paper is organized as follows: In Section 2, we describe the proposed vocal beat tracking task with dataset curation and a new evaluation scheme. In Section 3, we describe our proposed models with details. In Section 4, we present experimental comparisons with baselines and an ablation study on the feature extraction front end. Finally, Section 5 concludes the paper.

2. THE PROPOSED TASK

We propose singing beat tracking as a new MIR task, which aims to design algorithms that can track beats of singing voices in online or offline fashion. In this section, we describe the two cornerstones for this task: datasets and evaluation metrics.

2.1 Singing Data and Label Generation

Data for training and evaluating singing beat tracking algorithms require both singing voice recordings and their beat annotations. As there is no existing datasets that meet this requirement, we need to design a mechanism to collect such data. Instead of recording singing voices and annotating their beats manually, we propose to leverage existing MIR datasets and techniques to collect data for singing beat tracking, as illustrated in Figure 1.

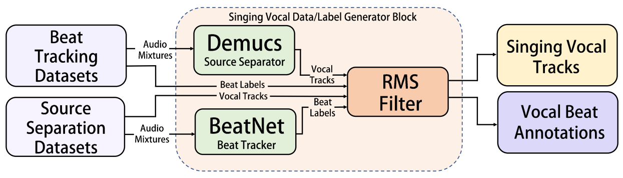

Figure 1. Singing data and label generation pipeline.

The first strategy we propose is to use pre-existing music beat tracking datasets in which the beat annotations are available. The idea is to obtain their singing tracks through music source separation. State-of-the-art (SOTA) singing voice separation methods have been shown to achieve outstanding results on popular genres of music, and the separated singing voice contains little interference from the background music. For this task, we employ Demucs Hybrid [11], one of the superior music source separation models. It uses a waveform-to-waveform convolutional autoencoder with a U-Net structure and bidirectional LSTM layers, and is designed to separate the music mixture into four sources including bass, drums, vocals, and others. We use the pre-trained model [1] which was trained on all 150 songs of MusDB [12] dataset. It is noted that source separation is used in some related works e.g. [13–15] to improve music beat tracking while in this work, we utilize it as one of the suggested systematic ways to generate singing vocal beat tracking datasets.

The second strategy we propose leverages pre-existing music source separation datasets, in which the isolated vocal tracks are available as a part of those datasets. To obtain beat annotations, we apply BeatNet [6] offline version on the complete songs (i.e., music mixtures). BeatNet is a SOTA online beat tracking method that uses a convolutional recurrent neural network (CRNN) and a hidden Markov model (HMM) decoder to extract music beats and downbeats. We modify the HMM decoder from particle filtering inference to Viterbi algorithm to improve its performance in the offline scenario. The reason that we run BeatNet on music mixtures instead of the separated vocal tracks is because it, the same as all other beat tracking methods, is trained on complete music pieces. While BeatNet has been shown to be fairly accurate on many songs with different genres, there are still annotation errors. To fix them, we also perform a manual revision by listening to the separated vocal track together with synthesized beeping sounds of the beat annotations and correcting errors.

While the beat annotations are obtained for the entire song, the ground-truth or separated vocal tracks show long chunks of silence due to the inactivity of singing. These long silent chunks are not useful and even are distracting for singing rhythmic analysis. Therefore, we normalize the energy of each vocal track using the root-mean-square (RMS), calculate a frame-wise RMS, and set a threshold to detect long silent chunks and split the vocal track into vocal segments. The RMS threshold is set to 0.01, and silent chunks shorter than 8 seconds are kept.

We apply these two strategies to a total of 8 existing beat tracking and music source separation datasets to obtain a total of 2248 vocal excerpts with beat annotations. The entire length of the vocal segments is 34h 35m. The datasets are summarized in Table 1.

2.2 Evaluation Metrics

For evaluation metrics, we adopt the commonly used F-measure in beat tracking. A beat is considered correctly detected if it is matched with a ground-truth beat with a time deviation smaller than 70 ms. In addition, we employ three additional metrics including P-score, Cemgil and Goto to provide a more detailed evaluation. Details about these three metrics can be found in [3].

For many music pieces, vocal tracks can align with the offbeat position (i.e., middle point between two adjacent ground-truth beats) rather that the beats. In fact, it also can be quite natural for humans to clap on the 180-degree phase shifted positions While this inherent ambiguity also exists in some instrumental music, it is much more common for singing voices. Apparently, the off-beat predictions would

Dataset	*# Number of vocal excerpts*	*Total Length*
Ballroom [16,17] *	452	2 h 38 m
GTZAN [18,19] *	741	5 h 44 m
Hainsworth [20]*	154	1 h 47 m
MUSDB18 [12]†	263	6 h 21 m
Rock Corpus [21]*	315	9 h 23 m
RWC pop [22,23]*	188	5 h 06 m
RWC Royalty free [22,23]*	29	19 m
URSing [24]†	106	3 h 17 m

Table 1. Datasets collected and adapted for singing beat tracking. $*$ denotes beat tracking datasets and † denotes music separation datasets.

[1] https://github.com/facebookresearch/demucs