# DIFFERENTIABLE WAVETABLE SYNTHESIS

*Siyuan Shan*[1,⋆]     *Lamtharn Hantrakul*[2]     *Jitong Chen*[2]     *Matt Avent*[2]     *David Trevelyan*[2]

[1] University of North Carolina at Chapel Hill
[2] Speech, Audio & Music Intelligence (SAMI) Team, ByteDance

## ABSTRACT

Differentiable Wavetable Synthesis (DWTS) is a technique for neural audio synthesis which learns a dictionary of one-period waveforms i.e. wavetables, through end-to-end training. We achieve high-fidelity audio synthesis with as little as 10 to 20 wavetables and demonstrate how a data-driven dictionary of waveforms opens up unprecedented one-shot learning paradigms on short audio clips. Notably, we show audio manipulations, such as high quality pitch-shifting, using only a few seconds of input audio. Lastly, we investigate performance gains from using learned wavetables for realtime and interactive audio synthesis.

***Index Terms***— Differentiable Digital Signal Processing, Wavetable Synthesis, Differentiable Dictionaries

## 1. INTRODUCTION

Although machine learning (ML) has revolutionized modern audio synthesis with unprecedented progress [1, 2, 3]; fast, robust and realtime neural audio synthesis remains a challenge. Purely deep audio models require significant parallelization [4] or custom kernels [3] for fast performance. Recent techniques fusing differentiable signal processing with deep neural networks (DNN) [5] have enabled efficient, interpretable and real-time use cases [6, 7] with minimal additional engineering.

Wavetable synthesis (WTS) is well-suited to realtime synthesis of periodic and quasi-periodic signals. Real-world objects that generate sound often exhibit physics that are well described by harmonic oscillations. These include vibrating strings, membranes, hollow pipes and human vocal chords [8]. By using lookup tables composed of single-period waveforms, WTS can be as general as additive synthesis whilst requiring less realtime computation [9]. We demonstrate three key contributions to neural audio synthesis using this approach: high fidelity synthesis, wavetables transferrable to other tasks and an order of magnitude reduction in computational complexity.

---

⋆ Work performed while a PhD intern at ByteDance. Corresponding Email: *siyuanshan@cs.unc.edu*

## 2. RELATED WORK

**Wavetable Synthesis (WTS):** Wavetable synthesis generates audio from a collection of static, single-period waveforms called "wavetables", that each capture a unique harmonic spectrum. Wavetables are typically 256 - 4096 samples in length; a collection can contain a few to several hundred wavetables depending on use case. Periodic waveforms are synthesized by indexing into the wavetables as a lookup table and interpolating between neighbouring samples. WTS was historically used on early hardware synthesizers where memory and computing power were limited. Today, WTS continues to underpin commercial sound design and synthesis tools due to its ability to generate a wide variety of timbres coupled with efficient performance. Wavetables are normally hand-crafted or extracted programatically from audio spectra [9, 10]. In this work, we learn data-driven wavetables.

**Differentiable dictionaries:** ML models can be equipped with differentiable memory or dictionaries that are learned end-to-end with model parameters. Models can write to external memory to record and lookup changes in state [11, 12]. Discriminative models can learn a differentiable dictionary for improved classification during inference [13]. Here, we formulate wavetables as a differentiable dictionary optimized jointly with model parameters.

**Differentiable Digital Signal Processing:** DDSP [5] describes a family of techniques utilizing strong inductive biases from DSP combined with modern ML. Successful applications include audio synthesis [5], pitch detection [14], artificial reverberations [15] and IIR filters [7]. Building on these works, we add WTS as a new technique for generative audio tasks.

## 3. METHODS

### 3.1. Differentiable Wavetable Synthesizer (DWTS)

**Wavetable as learnable dictionary:** We define a learnable dictionary $D = \{\boldsymbol{w}_i\}_i^N$ where $N$ is the number of wavetables and $\boldsymbol{w}_i \in \mathbb{R}^L$ denotes a one-cycle wavetable of length $L$. When a wavetable begins and ends on different values, this discontinuity causes synthesis artefacts. We append $\boldsymbol{w}_i[L + 1]$ to $\boldsymbol{w}_i$ and set $\boldsymbol{w}_i[L + 1] = \boldsymbol{w}_i[0]$. A wavetable $\boldsymbol{w}_i$ now
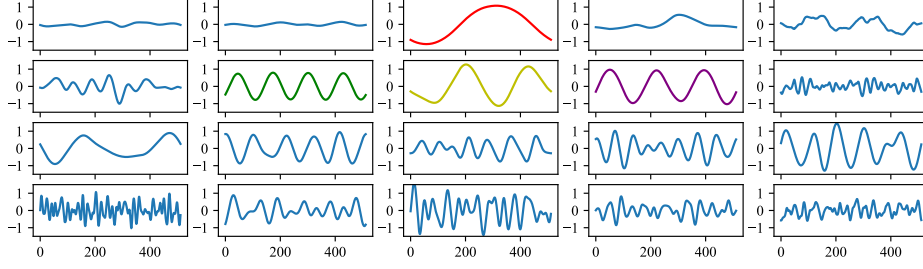
**Fig. 1**. Learned wavetables ordered with highest average attention weights appearing first (normal English reading order). Wavetables of key harmonics are highlighted: $f_0$ (red), $f_1$ (yellow), $f_2$ (purple) and $f_3$ (green). The remaining wavetables are data-driven combinations of higher harmonics. The first two wavetables appear to be silence.

contains $L + 1$ elements with $L$ learnable parameters. During training, we learn $D$ using gradient descent jointly with other parameters. During inference, $D$ is frozen and treated as a traditional, static collection of wavetables.

**Time-varying attention:** By sequentially morphing between wavetables, timbre can be changed over time [10]. Inspired by [16, 17] we generalize morphing as a time-varying linear attention over all wavetables i.e. $c_1^N, c_2^N...c_T^N$ where $N$ and $T$ are number of wavetables and timesteps respectively with constraints $\sum_{i=1}^{N} c_i(n) = 1$ and $c_i(n) \geq 0$.

**Phase:** Our model "draws" wavetables directly in the time domain. Phase relationships within and across wavetables can be controlled without needing to coherently manage independent magnitudes and phases in the complex frequency domain. This contrasts with [5], where the initial phase of all harmonic components are fixed at 0.

**Synthesis**: At the heart of WTS is a phase accumulator [9]. Given an input sequence of time-varying $f_0(n)$ over discrete time steps $n$, we can compute the instantaneous modulo phase $\tilde{\phi}$ by integrating $f_0(n)$:

$$\tilde{\phi}(n) = 2\pi \sum_{m=0}^{n} f_0(n) \mod 2\pi. \quad (1)$$

$\tilde{\phi}(n)$ is normalized into a fractional index $\tilde{j}(n) = \frac{L}{2\pi}\tilde{\phi}(n)$. We synthesize the signal $x(n)$ by linearly combining wavetables $\boldsymbol{w}_i$ in $D$ via:

$$x(n) = A(n) \sum_{i=1}^{N} c_i(n) \cdot \Phi(\boldsymbol{w}_i, \tilde{j}(n), \kappa), \quad (2)$$

where $A(n)$ is a time-varying amplitude controlling the signal's overall amplitude and $c_i$ denotes the time-varying attention on $\boldsymbol{w}_i$. $A(n)$ and $c_i(n)$ are constrained positive via a sigmoid. The function $\Phi(\boldsymbol{w}_i, \tilde{j}, \kappa)$ is a fractional indexing operator that returns the $(\tilde{j})$-th element of the vector $\boldsymbol{w}_i$ by using an interpolation kernel $\kappa$ to approximate $\boldsymbol{w}_i[\tilde{j}]$ when $\tilde{j}$ is non integer. Whilst more sophisticated interpolation kernels exist (cubic, spline etc.), we use linear interpolation for optimal



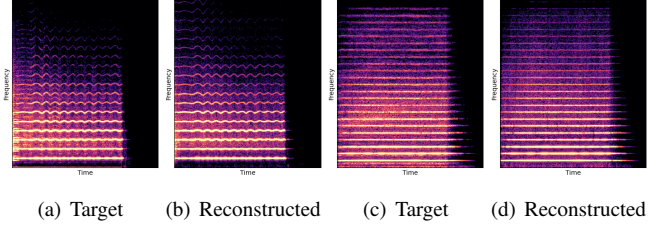| (a) Target | (b) Reconstructed | (c) Target | (d) Reconstructed |

**Fig. 2**. Spectrograms of two target samples and their corresponding reconstruction.

realtime performance. For readers unfamiliar with fractional indexing, please refer to the online supplement.

**Initialization:** $\boldsymbol{w}_i$ is randomly initialized with a zero-centered Gaussian distribution $\mathcal{N}(0, \sigma^2)$. We empirically find using a small $\sigma = 0.01$ improves training.

**Antialiasing:** At high $f_0$, the frequencies of upper harmonics in a wavetable can be above Nyquist and must be removed before lookup to prevent aliasing. This filter also prevents high frequency components present in the Gaussian noise initialization from causing aliasing at the start of training. Without this frequency-dependent anti-aliasing filter, we found aliasing artefacts alone prevented meaningful learning.

## 4. EXPERIMENTS

We benchmark against the original DDSP autoencoder [5], where a DNN controls an additive synth and filtered noise synth to produce harmonic and non-harmonic components of audio respectively. We replace the additive synth with our wavetable synth and use an identical filtered noise synth. Noise generation is a stochastic process that must be modelled separately. Like [5], we omit the optional reverb module when training on the NSynth dataset.

**Model:** We adopt an identical input tuple $(f_0(n), l(n), z(n))$ [5]. Fundamental frequency $f_0(n)$ in equation 1 is extracted by a pretrained CREPE model [18] with fixed weights. Loud-

**Table 1**. Reconstruction error comparison on the Nsynth dataset. Compared to the SOTA additive synthesis approach [5], our wavetable synthesis approach achieves comparable or lower errors.

| DDSP Additive Synthesis | Wavetable Synthesis (Ours) with $N$= | | | |
|---|---|---|---|---|
| | 5 | 10 | 20 | 100 |
| $0.5834 \pm 0.0035$ | $0.6448 \pm 0.0041$ | $0.5989 \pm 0.0042$ | $\mathbf{0.5712 \pm 0.0037}$ | $0.5756 \pm 0.0034$ |

ness $l(n)$ is an A-weighted log-magnitude extracted deterministically from audio. The residual embedding $z(n)$ is extracted from MFCC's via an encoder. For direct comparison, we use identical encoder and decoder architectures with approximately 7M parameters in total.

Unlike the autoencoder in [5], our setup contains an additional $N \times L$ learnable parameters in the wavetable dictionary $D$ during training. During inference however, the wavetables are frozen and the parameter counts near-identical.

**Loss:** We use a multi-scale spectral loss similar to [5]:

$$L_{\text{reconstruction}} = \sum_i ||S_i - \hat{S}_i||_1, \qquad (3)$$

where $S_i$ and $\hat{S}_i$ respectively denote magnitude spectrums of target and synthesized audio, and $i$ denotes different FFT sizes. We found the log term $|| \log S_i - \log \hat{S}_i||_1$ caused training instabilities and excluded it. This modification did not influence the quality of synthesized audio.

**Dataset** We use the same subset of the NSynth dataset in [5, 19, 20], containing 70,000 mono 16kHz samples each 4 seconds long. The examples comprise mostly of strings, brass, woodwinds and mallets. At 16kHz, a wavetable length $L = 512$ is enough to represent all harmonics at the lowest fundamental frequency of interest (20Hz).

## 5. RESULTS

### 5.1. Reconstruction Quality of DWTS

Table 1 reports the reconstruction error of the SOTA additive-based autoencoder from [5] and the proposed DWTS-based autoencoder. We vary the number of wavetables $N = 5, 10, 20, 100$. Our approach achieves the lowest reconstruction error of 0.5712 using only 20 wavetables. At the expense of a small reduction in quality compared to the baseline, $N$ can be low as 10. Fig 2 shows spectrograms of two samples and their matching reconstructions using wavetables. We encourage readers to listen to the online supplement[1].

Crucially, the wavetables in $D$ form an alternative, compact set of basis vectors spanning an $L$-dimensional space extracted directly from the data. When $N$ is very small at 5, reconstruction suffers due to an insufficient number of bases. 10-20 wavetables strike an optimal balance for the NSynth dataset. Wavetables reduce the number of control dimensions
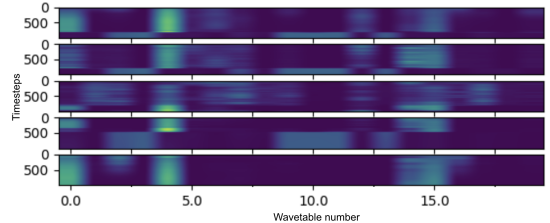
___

[1] https://lamtharnhantrakul.github.io/diffwts.github.io/



**Fig. 3**. Visualization of the time-varying attention weights of 5 samples using 20 wavetables.

by an order of magnitude compared to the 100 sinusoids in an additive synth [5]. More importantly, the extracted wavetables are an explicit dictionary that are portable to other tasks. We show this property in later sections.

### 5.2. Visualizing Wavetables

Fig 1 shows learned wavetables from the NSynth dataset when $N = 20$. Despite being initialized with noise, the learned wavetables are smooth and diverse in shape. They also match the physics of NSynth sounds. In acoustic instruments, energy is focused on lower frequencies, particularly the first few harmonics, compared to higher harmonics [21]. Wavetables in Fig 1 are ordered with highest average attention weights appearing first. The wavetable highlighted in red is a phase-shifted sinusoid of one period i.e. the fundamental frequency $f_0$. Other key partials $f_1$, $f_2$ and $f_3$ are highlighted in yellow, purple and green. The remaining wavetables are data-driven combinations of higher harmonics, compactly summarizing in a single wavetable entry what would have taken several sinusoidal components in an additive synth [5] to represent. Fig 3 shows the attention weights over time for five audio samples. The attention visibly shifts across wavetables to output the desired spectrum.

The asymmetric wavetables are the result of complex behavior in magnitude and phase. We found phase-locking wavetables to start and end at 0 deteriorated performance. It suggests the model takes advantage of phase relationships within and between wavetables. This precise control of wavetable phase will be particularly valuable in future work exploring synthesis of stereo and binaural audio [22].
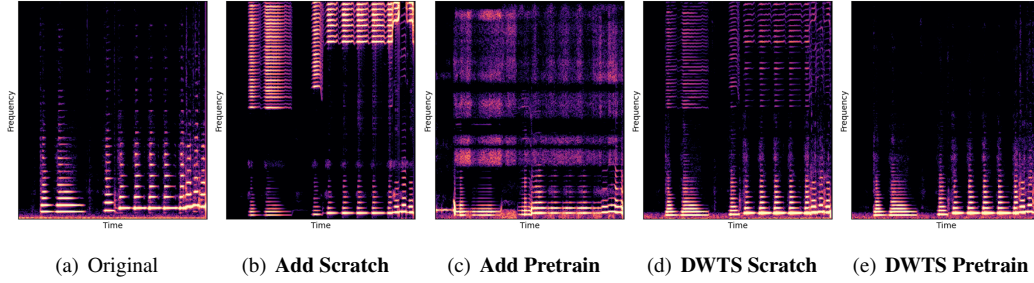
| (a) Original | (b) **Add Scratch** | (c) **Add Pretrain** | (d) **DWTS Scratch** | (e) **DWTS Pretrain** |

**Fig. 4**. Spectrograms of original audio (a) and synthesized samples from an input $f_0(n)$ pitch shifted down by an octave (b-e)

### 5.3. One shot learning and audio manipulations

In domains such as Neural Machine Translation (NMT), word embeddings [23] extracted from end-to-end language learning are useful for other language tasks. Analogously, we reason our data-driven wavetables should be useful in other scenarios like one-shot learning and data-efficient extrapolation. Unlike an implicit multi-dimensional vector, wavetables are an explicit and interpretable representation.

**One-shot setup:** Given only a single 4 second passage of saxophone from the URMP dataset [24], we train a new autoencoder model initialized with pretrained wavetables from Fig 1 (*DWTS Pretrain*). This model only outputs time-varying attention weights, since the wavetables are now a fixed dictionary lookup. We compare against three baselines: (1) additive-synth autoencoder trained from scratch (*Add Scratch*), (2) finetuning an additive-synth autoencoder pretrained on Nsynth (*Add Pretrain*) and (3) Wavetable-synth autoencoder trained from scratch (*DWTS Scratch*).

**Pitch extrapolation:** While all models achieve identical high quality one-shot reconstructions of the saxophone segment, only *DWTS Pretrain* is robust to overfitting during extrapolation. Fig 4 shows how all baselines exhibit high frequency artefacts when input $f_0(n)$ is shifted. *DWTS+Pretrain* remains artefact free even at extreme shifts, such as 3 octaves below the original sample.

We repeat with a 4 second piano passage. A piano is challenging to model due to the presence of both many harmonics and percussive hammer hits [21]. We also compare against the Librosa library pitch shift function based on traditional DSP [25]. When resynthesizing the segment 3 octaves down, *DWTS Pretrain* is the only method that preserves the hammer's percussive impact and independently shifts harmonic components. Librosa pitch shift loses the transient impact completely.

**Connection to PSOLA:** In this scenario, we hypothesize *DWTS Pretrain* approaches an optimal Pitch Synchronous Overlap and Add (PSOLA) algorithm [26]. PSOLA windows a single cycle of the original waveform in the time domain, re-patching and overlapping these windows at the new pitch.

Imperfections in this windowing and overlapping process can cause artefacts. *DWTS Pretrain* guarantees single-cycle waveforms in $D$, where re-pitching is trivially achieved using a slower phase accumulator $\tilde{\phi}(n)$ reading through a wavetable $\boldsymbol{w}_i$. This opens up applications like data-efficient neural sampling, pitch correction and efficient polyphony using multiple phase accumulators. We leave wavetables extracted from speech or singing voice for future work.

### 5.4. Computational Complexity

Realtime performance is determined by many factors including algorithmic complexity, low level optimizations, target hardware and the C++ ML framework used for matrix operations. Here, we consider only the key difference between the additive synth-based pipeline in [5], which already runs realtime [6], and our new approach.

A.) *Additive*: a bank of 100 sinusoids where harmonic coefficients are updated at 250 frames per second (FPS) [5]

B.) *DWTS*: 10 pre-learned wavetables where wavetable weights are also updated at 250 FPS.

The remaining elements of each approach are assumed identical in specification and performance. There are two clear areas of improvement with our new method. Firstly, *DWTS* requires only ten interpolated wavetable read operations per sample, compared to the 100 required for *Additive*. Secondly, both pipelines require frame-wise linear smoothing of control data to avoid unwanted artefacts in the synthesized signals. *Additive* requires all 100 sinusoidal amplitude coefficients to be smoothed per sample, whereas *DWTS* requires smoothing on only 10. We confirm this on a 2.6GHz 2019 Macbook Pro. Over 10k trials, *Additive* takes on average 251.32ms to generate all 250 frames in 1 second of audio, whereas *DWTS* takes only 20.19ms.

## 6. CONCLUSION

In this paper, we presented a differentiable wavetable synthesizer capable of high fidelity neural audio synthesis. We demonstrate how explicit and learnable wavetables offer

many advantages including robust one-shot learning and an order of magnitude reduction in computational complexity. The approach opens up applications such as data-efficient neural audio sampling and pitch-shifting.

## 7. REFERENCES

[1] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016.

[2] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, et al., "Tacotron: A fully end-to-end text-to-speech synthesis model," *CoRR*, vol. abs/1703.10135, 2017.

[3] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, Aäron van den Oord, Sander Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," *ICML*, 2018.

[4] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," 2021.

[5] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts, "Ddsp: Differentiable digital signal processing," *ICLR*, 2020.

[6] Francesco Ganis, Erik Frej Knudesn, Søren VK Lyster, Robin Otterbein, David Südholt, and Cumhur Erkut, "Real-time timbre transfer and sound synthesis using ddsp," *arXiv preprint arXiv:2103.07220*, 2021.

[7] Julian D. Parker Boris Kuznetsov and Fabián Esqueda, "Differentiable iir filters for machine learning applications," *DaFX*, 2020.

[8] Julius Smith, *Physical Audio Signal Processing: for Virtual Musical Instruments and Digital Audio Effects*, 01 2006.

[9] Robert Bristow-Johnson, "Wavetable synthesis 101, a fundamental perspective," in *Audio Engineering Society Convention 101*. Audio Engineering Society, 1996.

[10] Andrew B. Horner and James W. Beauchamp, *Instrument Modeling and Synthesis*, pp. 375–397, Springer New York, New York, NY, 2008.

[11] Alex Graves, Greg Wayne, and Ivo Danihelka, "Neural turing machines," *arXiv:1410.5401*, 2014.

[12] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, et al., "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[13] Siyuan Shan, Yang Li, and Junier B Oliva, "Meta-neighborhoods," *NeurIPS*, 2020.

[14] Jesse Engel, Lamtharn Hantrakul, Rigel Swavely, Adam Roberts, and Curtis Glenn-Macway Hawthorne, "Self-supervised pitch detection by inverse audio synthesis," in *ICML SAS Workshop*, 2020.

[15] Sungho Lee, Hyeong-Seok Choi, and Kyogu Lee, "Differentiable artificial reverberation," *CoRR*, vol. abs/2105.13940, 2021.

[16] Andrew Horner, "Wavetable matching synthesis of dynamic instruments with genetic algorithms," vol. 43, pp. 916–931, 11 1995.

[17] Robert Bristow-Johnson, "Comments on 'a simplified wavetable matching method using combinatorial basis spectra selection'," *journal of the audio engineering society*, vol. 51, no. 3, pp. 162–163, march 2003.

[18] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello, "Crepe: A convolutional representation for pitch estimation," in *ICASSP*, 2018.

[19] Lamtharn Hantrakul, Jesse H Engel, Adam Roberts, and Chenjie Gu, "Fast and flexible neural audio synthesis.," in *ISMIR*, 2019.

[20] Jesse H. Engel, Kumar Krishna Agrawal, Shuo Chen, I. Gulrajani, C. Donahue, and Adam Roberts, "Gansynth: Adversarial neural audio synthesis," *ICLR*, 2019.

[21] Thomas D. Rossing and Neville H. Fletcher, *Physics of Musical Instruments 2nd Edition*, Springer New York, New York, NY, 1998.

[22] Alexander Richard, Dejan Markovic, Israel D Gebru, Steven Krenn, Gladstone Alexander Butler, Fernando Torre, and Yaser Sheikh, "Neural synthesis of binaural speech from mono audio," in *ICLR*, 2020.

[23] Tomas Mikolov, I. Sutskever, K. Chen, Greg S Corrado, and J Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[24] Bochen Li, Xinzhao Liu, K. Dinesh, Z. Duan, and Gaurav 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, 2018.

[25] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the 14th Python in Science Conference*, 2015, pp. 18 – 24.

[26] F. Charpentier and M. Stella, "Diphone synthesis using an overlap-add technique for speech waveforms concatenation," in *ICASSP*, 1986, vol. 11, pp. 2015–2018.