VIRTUAL ANALOG MODELING OF DISTORTION CIRCUITS USING NEURAL ORDINARY DIFFERENTIAL EQUATIONS

Jan Wilczek *

WolfSound
Katowice, Poland
jan.wilczek@thewolfsound.com

Alec Wright and Vesa Välimäki †

Acoustics Lab
Dept. Signal Processing and Acoustics
Aalto University, Espoo, Finland
alec.wright@aalto.fi

Emanuël A. P. Habets

International Audio Laboratories Erlangen ‡
Erlangen, Germany
emanuel.habets
@audiolabs-erlangen.de

ABSTRACT

Recent research in deep learning has shown that neural networks can learn differential equations governing dynamical systems. In this paper, we adapt this concept to Virtual Analog (VA) modeling to learn the ordinary differential equations (ODEs) governing the first-order and the second-order diode clipper. The proposed models achieve performance comparable to state-of-the-art recurrent neural networks (RNNs) albeit using fewer parameters. We show that this approach does not require oversampling and allows to increase the sampling rate after the training has completed, which results in increased accuracy. Using a sophisticated numerical solver allows to increase the accuracy at the cost of slower processing. ODEs learned this way do not require closed forms but are still physically interpretable.

1. INTRODUCTION

Virtual Analog (VA) models are digital emulations of audio systems that were originally built using analog electronic or electromechanical components [1]. They arose out of the demand for the reproduction of characteristic tonal distortions of analog devices but with the digital stability and ease of integration with existing software. Devices modeled range from filters [2] through time-varying effects [3–5], amplifiers [6–8], and mechanical reverb units [9] to tape or vinyl distortions [10].

One can distinguish three general approaches to VA modeling [4, 5]. In *black-box* modeling, only the input-output relation of a system is examined and a signal model is constructed to mimic that behavior. Neural networks have successfully been applied to this kind of modeling for guitar amplifiers [6–8]. An architecture provided in [7] was chosen as a baseline for this work.

In *white-box* modeling, the internal structure of the system under study is examined and used to construct an algorithm reproducing the behavior of the device. Sometimes this approach is referred to as *physical modeling*. Typical methods in this category are numerical solutions of ordinary differential equations (ODEs) derived from electronic circuit analysis [3, 11, 12] or wave-digital filters [13].

In white-box modeling, it is assumed that analog electronic circuits are dynamical systems that can be described by differential equations. However, the derivation of these equations is a difficult task that requires expert knowledge. Additionally, solving these equations poses a challenge because of the inherent aliasing [11]. Typical approaches to this problem, such as oversampling or local iterations of implicit solvers, increase the processing times, often preventing real-time applications.

Grey-box modeling falls in between the two already mentioned approaches. In this methodology, we use some knowledge about the inner workings of the device under study to design a model and, subsequently, take advantage of the available signal data to adjust the model's parameters. This approach has been successfully applied to distortion circuits modeling [2], adjusting component values of white-box circuit models [14], and time-varying effects modeling [4, 5] also in conjunction with neural networks.

In related work, Parker et al. used a multilayer perceptron (MLP) to learn the residual of a state-space system [2] in a grey-box fashion. This residual network (ResNet) [15] approach was called State Trajectory Network (STN). The authors successfully applied STN to model analog first-order and second-order clipper circuits and an analog filter. STN can be used at sampling rates different than the sampling rate of the training set by properly scaling the residual but the authors did not provide any results concerning this feature.

Recent research relating deep learning and ODEs has indicated that ODEs governing analog electronic circuits could be learned from data. The concept of teaching a neural network the derivative of an unknown function and then supplying the learned derivative to a numerical solver, termed ODENet, was introduced in [16]. Karlsson and Svanström applied it to dynamical systems modeling [17]. Since this approach combines data-driven neural network training (an approach typically classified as black-box) and the assumption that the modeled system is governed by an ODE (usually associated with white-box models), it could be classified as grey-box.

The potential benefits of coupling a neural network with a numerical solver in VA modeling are manifold. First, we could obtain an empirical ODE that would replace complicated analytical expressions based on simplifying assumptions and often unknown, imprecise, or condition-dependent physical quantities [14]. Taking this even further, we could model audio effects previously not described by ODEs using solely signal data. Second, we could possibly alleviate various problems inherent to numerical solutions of closed-form ODEs such as aliasing. Third, since an ODE is physically interpretable, we could alter the sampling rate of the model

* This work was supported by a fellowship within the IFI programme of the German Academic Exchange Service (DAAD).

† This research is part of the activities of the Nordic Sound and Music Computing Network—NordicSMC (NordForsk project no. 86892).

‡ A joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits (IIS).

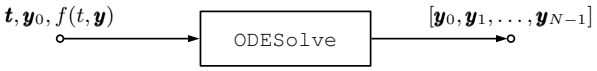


Figure 1: A general pipeline for solving ODEs. In ODENet, derivative function f is replaced by a neural network.

after training, which is typically not possible with purely black-box models. Finally, by obtaining simplified, alias-free ODEs, we could add real-time performance capabilities to white-box models, thus, letting musicians and producers use previously unavailable software emulations of analog hardware.

To validate the above possible benefits, we adapted ODENet to VA modeling. In this paper, we applied it to the first- and the second-order diode clipper. Through these examples, *we confirmed all of the above potential benefits* leaving possible real-time implementation for future work. This paper discusses the ODENet implementation in the context of VA, presents the results of experiments, and analyzes strengths and weaknesses of this approach.

This paper is organized as follows. In Section 2, the general concept of learning a derivative is described. In Section 3, the proposed extension to this approach, which facilitates VA modeling, is discussed. In Section 4, we apply the derivative-learning approach to the first-order and the second-order diode clipper circuits, compare it to the baseline, and discuss the results. Finally, Section 5 recapitulates the most important conclusions and forms the perspective for future work. Audio examples of the presented models can be found under `https://www.thewolfsound.com/publications/dafx2022/`. The full code without the datasets can be found under `https://github.com/JanWilczek/va-phaser-with-neural-odes`.

2. NEURAL ORDINARY DIFFERENTIAL EQUATIONS

An ordinary differential equation (ODE) is an equation of the form

$$\frac{d\boldsymbol{y}}{dt} = f(t, \boldsymbol{y}), \tag{1}$$

where $\boldsymbol{y} = \boldsymbol{y}(t)$ is the vector-valued *unknown function* of an *independent variable* t, here representing time, and $f(t, \boldsymbol{y})$ is a known function of t and $\boldsymbol{y}$, which equals to the first derivative of $\boldsymbol{y}$ over time [18]. An entry of the unknown function vector may, for example, represent the voltage across a capacitor inside a circuit. We refer to a vector-valued unknown function for generality and because every system of ODEs can be formulated as a system of first-order ODEs [17].

In Figure 1, one can see a generic approach to solving a system of ODEs with a numerical solver. A vector of time points $\boldsymbol{t}$, is supplied with the derivative function $f(t, \boldsymbol{y})$, and the *initial value* $\boldsymbol{y}_0$ (value of $\boldsymbol{y}$ at $t = t[0]$) to a numerical solver (marked as `ODESolve`). The result is a series of estimates of the value of $\boldsymbol{y}$ at the time points specified in $\boldsymbol{t}$. The initial value plays a crucial role in solving most ODEs because it determines the initial state and may influence the overall dynamics of the system.

Chen et al. [16] showed that a neural network used in conjunction with a numerical solver can learn $f(t, \boldsymbol{y})$ from data. In that framework, termed ODENet, the derivative function f from Eq. (1) and Figure 1 is replaced with a *derivative network* f which is trained based on the loss calculated using the solver's output.

Karlsson and Svanström [17] demonstrated that a neural network can learn $f(t, \boldsymbol{y})$ describing a simple dynamical system such as an oscillator. That indicated that a neural network can learn the derivative function of ODEs describing analog electronic circuits for VA purposes, which could provide the benefits mentioned in Section 1.

3. EXTENSION FOR VIRTUAL ANALOG MODELING

To use ODENet for VA modeling, a few issues must be considered: what initial value to supply, how to incorporate the input signal (e.g., a guitar signal), how to parametrize the neural network, and which solver to use.

3.1. Overview

In Figure 2, our implementation of the ODENet framework for VA modeling is presented. For clarity, we omit the idea of mini-batches and assume that a single example is processed at a time. One dataset example is a sequence of samples of the input signal to the modeled audio effect, $\boldsymbol{x} = [x_0, \ldots, x_{N-1}]^\mathrm{T}$, and the corresponding target sequence (desired effect output), $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{N-1}$. The output of ODENet is an estimated sequence, $\hat{\boldsymbol{y}}_0, \ldots, \hat{\boldsymbol{y}}_{N-1}$, which is used to compute the loss.

3.2. Initial Value

Supplying an initial value is marked as "set_initial_value($\boldsymbol{y}_0$)" in Figure 2. A proper initial value to the numerical solver plays a crucial role in obtaining a correct result of the ODE [18]. An inaccurate initial condition can lead the solver to a completely incorrect part of the solution space. That is why we decided to supply the ground truth initial value for each subsequence during training, an approach known in deep learning as teacher forcing [19, 20]. This is especially important because we split our training set into sequences of 22 050 samples. A group of sequences constitutes a minibatch. Each minibatch is processed in subsequences of 2048 samples (after each subsequence, there is a gradient step). Thus, each subsequence is most likely to start in the middle of a waveform, so a zero value would be incorrect most of the time.

However, at test time, the framework must not be able to access ground truth information. Additionally, in contrast to the training phase, the test sequence is processed as one long sequence without minibatches. Therefore, we decided to use an all-zero vector as the initial value of the test sequence, i.e., $\boldsymbol{y}_0 = \boldsymbol{0}$.

3.3. Excitation

In ODEs modeled with ODENet prior to this work, only a simple, fixed-cosine excitation had been considered [17]. In the context of VA modeling, the excitation is the input signal to the modeled audio effect. We provide the input signal directly to the neural network. The derivative network uses linear interpolation to obtain the input signal value at time points specified by the solver. In other words, when the solver calls the network f as $f(t, \boldsymbol{y}_t)$ for some t and $\boldsymbol{y}_t$, the network linearly interpolates the discrete input signal $\boldsymbol{x}$, to obtain an estimate of $x(t)$, x_t, and then uses x_t and $\boldsymbol{y}_t$ as its input to output the estimate of the derivative $\hat{\boldsymbol{y}}'_t$. This can be seen in the "Solver Loop" in Figure 2.

3.4. Neural Network Parametrization

Although the ODENet framework as a whole may be perceived as a form of a recurrent neural network (RNN) (because it may use

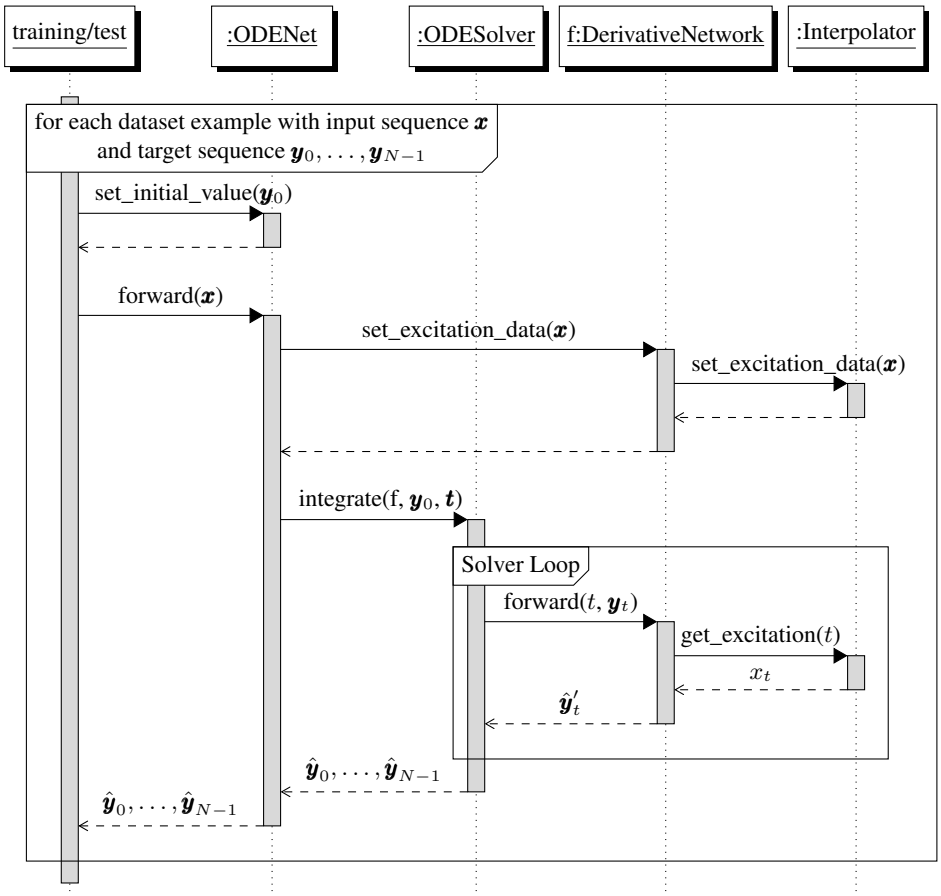


Figure 2: Sequence diagram containing the details of ODENet processing. The label "forward" denotes instance calls with specified arguments.

previous outputs to compute subsequent outputs), the derivative itself only maps the time and the state vector to the derivative value as shown in Eq. (1). This property narrows down the space of possible parametrizations of the derivative network to feedforward networks, by our choice, MLPs.

In Figure 3, we show how ODENet works under the forward Euler integration scheme [11]. The current input sample and current output sample are provided to the derivative network that estimates the current derivative value. This derivative value is used to estimate the next output sample, which is retained for the subsequent iteration.

The difference between the processing in Figure 3 and STN (ResNet) processing is that the latter would use a "current input, previous output" neural network input, not the "previous input, previous output" (see Figure 2 in [2]). Generally speaking, while STN has a fixed numerical scheme, ODENet allows for a flexible choice of a numerical solver and a use of different solvers at training and at test time. Additionally, ODENet learns the continuous derivative, whereas the STN learns its discrete approximation. Finally, ODENet and STN have significantly different learning dynamics. From our observations, the training of STN requires severe regularization in the form of learning rate schedules [21].

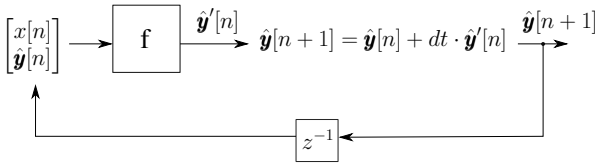


Figure 3: ODENet processing using the forward Euler scheme.

3.5. Numerical Solvers

Applying numerical solvers to audio processing is challenging. In time-stepping methods, one needs to multiply the derivative by the time step size or its fraction at each step [18]. In audio, the step size is very small, resulting in large derivative variations, numerical errors, and slow neural network learning. To combat these issues, we scaled the underlying ODE by the sampling rate so that the time step size is 1. We adequately scaled down the state vector passed to the derivative network to not alter the derivative learned.

Another challenge related to numerical solvers is their real-time performance. Explicit solvers can become unstable when applied to certain kinds of ODEs with an insufficient step size (i.e., sample rate). For example, Yeh et al. needed to use 38x oversam-

pling with the forward Euler scheme when solving the analytical ODE of the first-order diode clipper [11]. Implicit solvers, which iterate locally to reduce the error and, thus, provide very accurate solutions, have unpredictable running times unless one bounds the number of iterations per step. They also need oversampling, albeit smaller [11]. Explicit versus implicit choice is an example of a broader design issue of choosing a numerical solver: typically, with increased accuracy comes increased processing time. In this work, we show that oversampling is not needed when using a derivative learned from data, but the solver choice is still relevant for the model's accuracy.

4. MODELING DISTORTION CIRCUITS

This section presents the analog systems that were modeled, the modeling procedure, and the obtained results. First, the training procedure is outlined. Second, the first-order diode clipper is presented and the results of modeling it discussed with distinction between model quality and accuracy at different sampling rates. Finally, the second-order diode clipper is treated in a similar way.

4.1. Training

After [5–8], this work used the following combined loss function for neural network training of the first-order diode clipper

$$\mathcal{E}(y, \hat{y}) = \mathcal{E}_{\text{ESR}}(y_\text{p}, \hat{y}_\text{p}) + \mathcal{E}_{\text{DC}}(y, \hat{y}), \tag{2}$$

where y is the target signal, $\hat{y}$ is the estimated signal, "p" in the subscript marks signals that were pre-emphasized with a first-order high-pass filter of the form [5, 6]

$$H(z) = 1 - 0.85z^{-1}, \tag{3}$$

error-to-signal ratio (ESR), $\mathcal{E}_{\text{ESR}}$, is given by

$$\mathcal{E}_{\text{ESR}}(y_\text{p}, \hat{y}_\text{p}) = \frac{\sum_{n=0}^{N-1}(y_\text{p}[n] - \hat{y}_\text{p}[n])^2}{\sum_{n=0}^{N-1}(y_\text{p}[n])^2}, \tag{4}$$

and the direct current (DC) loss term, $\mathcal{E}_{\text{DC}}$, is given by

$$\mathcal{E}_{\text{DC}}(y, \hat{y}) = \frac{\left(\frac{1}{N}\sum_{n=0}^{N-1}(y[n] - \hat{y}[n])\right)^2}{\frac{1}{N}\sum_{n=0}^{N-1}(y[n])^2}. \tag{5}$$

For the second-order diode clipper, we used plain $\mathcal{E}_{\text{ESR}}$ given by Eq. (4) not to suppress the DC component in the second state of the circuit discovered in the synthesized data.

The dataset used for clipper circuits modeling consisted of 7 minutes and 59 seconds of guitar and bass recordings from [22] and [23], respectively. The amount of guitar recordings was roughly the same as the amount of bass recordings and their ordering was arbitrary. All recordings were single-channel and used 44.1 kHz sampling rate. The target distortion signals were synthesized from SPICE models of the circuits with the schematics from Figures 4 and 7 using LTspice XVII by Analog Devices. Approximately 20% of the dataset were used as the test set. The remaining data was split into the bass-only validation set and the training set according to the 80:20 rule.

The loss functions and dataset handling were implemented using the CoreAudioML library[1]. The remainder of the pipeline was implemented using the PyTorch library. We used exponential learning rates and the Adam optimizer [24].

[1] https://www.github.com/Alec-Wright/CoreAudioML

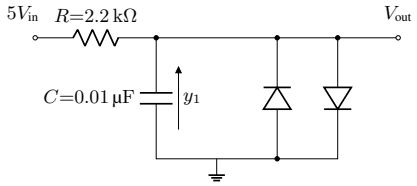


Figure 4: First-order diode clipper circuit with marked state y_1. Input voltage V_{in} was scaled by 5 in the target signal synthesis process.

Table 1: Compared network architectures for diode clipper modeling.

Model	Number of parameters	Epochs in training
LSTM8	361	200
STN 3 × 4 tanh	48	340
ODENet9-FE ReLU	127	300
ODENet9-IA SELU	127	600

4.2. First-Order Diode Clipper

The first-order diode clipper is a circuit used to achieve signal distortion, e.g., in guitar effects pedals [11]. Its schematic is shown in Figure 4. It can be regarded as consisting of two parts: an RC low-pass filter and a diode limiter.

The first-order diode clipper is a system particularly interesting in the context of ODENet, because it is governed by a known, nonlinear ODE derived by Yeh [11, 12]. The circuit has one state (voltage across a capacitor), y_1, which is taken as the output voltage V_{out}. The first-order diode clipper had already been modeled with STN [2].

4.2.1. Compared Models

The smallest derivative network that has reached a validation loss smaller than the assumed arbitrary threshold of 0.01 was a $2\times9\times9\times9\times1$ MLP (ODENet9). This network was subsequently trained in two setups: i) with the forward Euler (FE) scheme and the Rectified Linear Unit (ReLU) nonlinearity and ii) with the implicit Adams-Bashforth-Moulton (IA) scheme [17] and the scaled exponential linear unit (SELU) nonlinearity [25].

For benchmark, we chose the STN from [2] of size $2 \times 4 \times 4 \times 4 \times 1$ with the tanh nonlinearity and bias enabled only in the second layer (STN 3 × 4 tanh), and a Long Short-Term Memory (LSTM)-based architecture from [7] with 8 hidden units and an 8×1 output MLP mapping the hidden states to an output sample (LSTM8). Each model is listed with its hyperparameters in Table 1. The number of epochs in training was determined by the early stopping criterion [19].

All of the architectures were trained on audio data at 44.1 kHz sampling rate but tested on four different sampling rates to analyze the presence of aliasing in the output and inspect the interpretability of the learned ODEs. During the test, the STN and ODENet models were informed about the new value of the time step, whereas for LSTM8 it was not possible; LSTM implicitly learns a fixed time step from the training data.

Table 2: Signal-to-distortion ratio across all test samples in dB for the first-order diode clipper models.

Test sampling rate [kHz]	44.1	22.05	48	192
LSTM8	**30.9**	**18.5**	**30.7**	19.6
STN 3×4	20.4	-6.6	21.3	21.5
ODENet9-FE	21.3	2.6	21.7	23.6
ODENet9-IA	26.4	-0.8	27.3	**27.7**

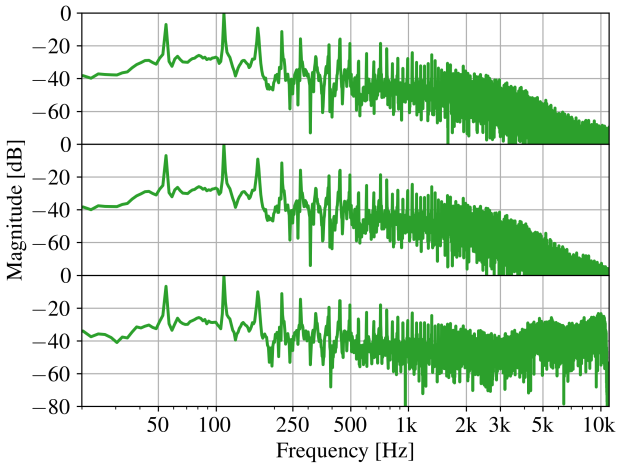


Figure 5: Magnitude spectrum of the A2 note played on the bass guitar as processed by the first-order diode clipper: (top) Target spectrum, (middle) ODENet9-FE output at 192 kHz sampling rate, and (bottom) ODENet9-FE output at 22.05 kHz sampling rate, showing aliasing at high frequencies. All spectra were normalized to the highest absolute value before the conversion to the decibel scale.

All models were tested on one long sequence, but the implicit Adams scheme consistently diverged in this test setting. Therefore, tests with ODENet9-IA were conducted using segments of 22 050 samples which were concatenated afterwards.

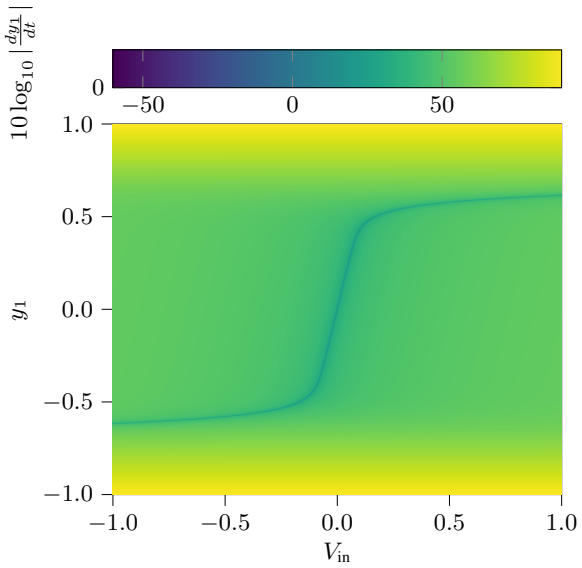
4.2.2. Results

In Table 2, test results of the compared models in terms of the signal-to-distortion ratio (SDR) rounded to one decimal place are shown. The best results (highest SDR) are given in bold. The models were separately evaluated in terms of the learned model quality (the test sampling rate equal to the training sampling rate) and the performance at sampling rates different from the training sampling rate.

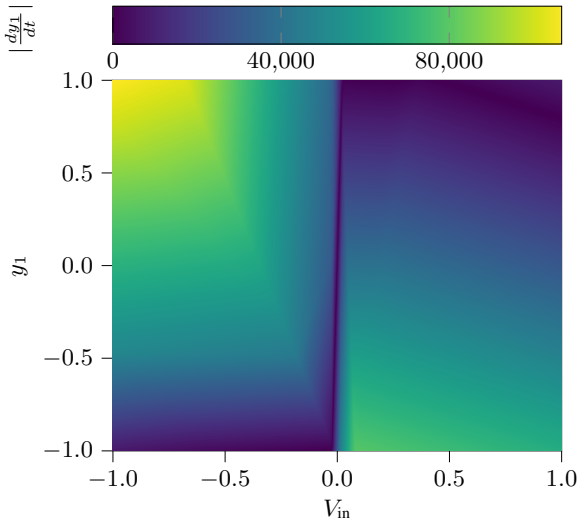
At 44.1 kHz, the SDR was very high for all compared architectures (more than 20 dB). In terms of the SDR, the LSTM outperformed all other models for 22.05 kHz, 44.1 kHz, and 48 kHz.

For 22.05 kHz test sampling rate, STN and ODENet models produced aliased output, which can be seen in Figure 5. This is to be expected because these models were trained with a higher sampling rate.

ODENet has some advantage for 192 kHz test sampling rate, with both models having the two highest SDR values. As could be expected, architectures that take the sampling rate into account during processing (STN, ODENet) obtained a higher SDR than the non-informed one. There was an advantage of the implicit solver over the explicit one in terms of the SDR but at the cost of doubled time needed to process the same amount of data.



(a) Closed-form. Every magnitude m is transformed to $10\log_{10}(m)$.



(b) ODENet9-FE

Figure 6: Magnitude of the derivative of the first-order diode clipper: analytical (top) and learned (bottom).

In Figure 6, one can see the analytical derivative function from [11, 12] and the one learned by ODENet9-FE. The learned derivative function, although similar in the S-shape, is much smoother than the analytical form. This difference probably comes from the limited frequency bandwidth of the dataset due to sampling. The learned derivative network visualization is on a par with the one shown in [2].

All in all, results comparable to the established LSTM and

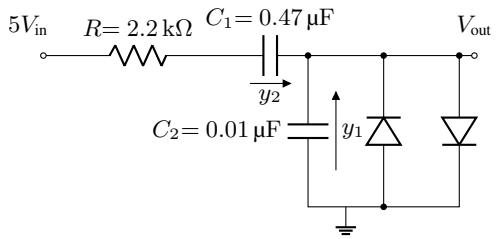


Figure 7: Second-order diode clipper circuit with marked states y_1 and y_2.

Table 3: Compared network architectures for the second-order diode clipper modeling

Model	Number of parameters	Epochs in training
LSTM16	1250	2000
STN 2×30 tanh	1112	630
ODENet20 Softsign	542	1200
ODENet30 Softsign	1112	1200

STN architectures prove that ODENet can be used for VA modeling and at sampling rates higher than the training sampling rate even for the simplest numerical scheme, i.e., the forward Euler.

4.3. Second-Order Diode Clipper

The second-order diode clipper circuit and the values of the components were taken from [2]. The circuit is similar to the first-order diode clipper with the only difference being an additional capacitor inserted between the output node and the resistor as shown in Figure 7. This additional capacitor introduces high-pass filtering and makes the clipping asymmetrical [2]. The model has two states, y_1 and y_2. The former is taken as the circuit output voltage V_{out}. This circuit was modeled with STN using real-world measurements in [2]. However, we again used a SPICE simulation that processed the same dataset as in the case of the first-order diode clipper.

4.3.1. Compared Models

In Table 3, the hyperparameters of the trained networks are presented. As a benchmark, we chose the LSTM-based architecture with 16 memory cells (LSTM16) and a $3 \times 30 \times 30 \times 2$ STN (STN 2×30). The derivative network of the ODENet was parametrized by a $3 \times 30 \times 30 \times 2$ MLP (ODENet30) and a $3 \times 20 \times 20 \times 2$ MLP (ODENet20), both with the Softsign nonlinearity. The former was chosen to have roughly the same number of trainable parameters as the benchmark, the latter was meant to be significantly smaller. The derivative network was used with the forward Euler (FE), trapezoidal rule (TR), and explicit Runge-Kutta of order 4 (RK4) numerical schemes [18]. The training proceeded as in the case of the first-order diode clipper.

Table 4: Signal-to-distortion ratio across all test samples in dB for the second-order diode clipper models.

Test sampling rate [kHz]	44.1	22.05	48	192
LSTM16	**21.2**	13.5	**20.6**	7.9
STN 2×30	16.9	**15.7**	16.8	**16.3**
ODENet20-RK4	14.3	9.0	14.2	14.1
ODENet30-FE	11.6	6.4	11.5	11.0
ODENet30-TR	15.5	8.2	15.4	13.9
ODENet30-RK4	15.7	14.7	15.6	15.7

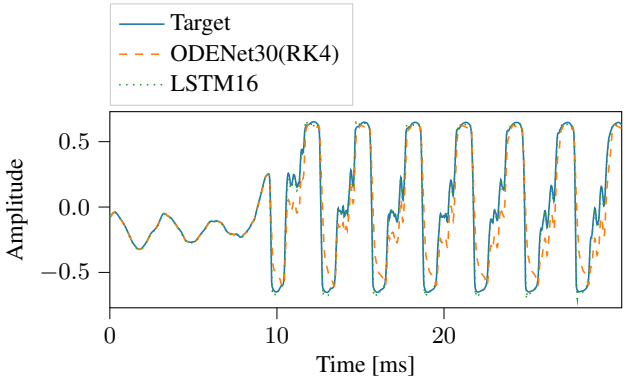


Figure 8: Time-domain comparison of the results of the second-order diode clipper modeling. Presented signal is a guitar note.

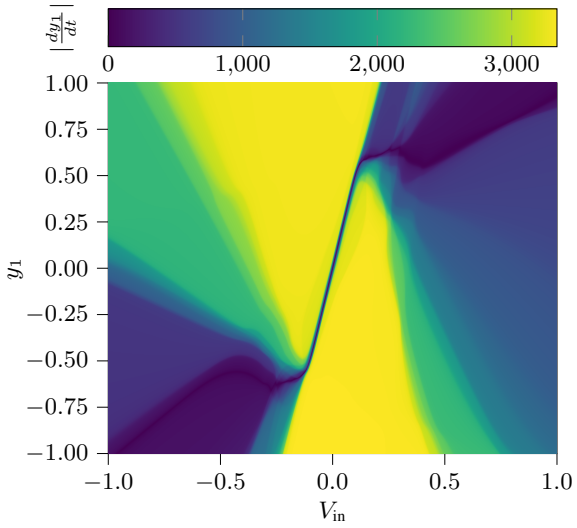
4.3.2. Results

Results in terms of the SDR are summarized in Table 4. As in the case of the first-order diode clipper, the LSTM-based architecture obtained the highest SDR for 44.1 kHz and 48 kHz. Somewhat surprisingly, STN obtained the highest SDR for the remaining sampling rates, even for 22.05 kHz, which introduced aliasing in this model's output for the first-order diode clipper. Again, the LSTM-based architecture was inferior to other (sampling rate-informed) models at 192 kHz.

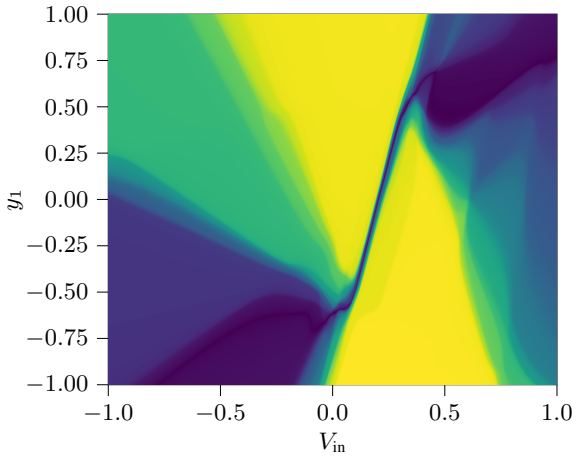
ODENet models clearly performed worse, with the most sophisticated ODENet30-RK4 being close behind STN. Results reveal, however, the expected outcome: more complicated solvers perform better than simple ones even though they are using the same derivative network architecture. Additionally, a smaller network using an advanced scheme ODENet20-RK4 outperformed a larger network using a simpler scheme ODENet30-FE, but this result should be treated more as a hint rather than a general rule.

Figure 8 shows the accuracy of LSTM16 and ODENet30-RK4 at 44.1 kHz in the time domain. ODENet seems to follow the signal well for small oscillations but fails to match the peaks of the waveform.

One can obtain more insight into the ODENet by inspecting the learned derivative of the state. The derivative is a vector-valued function of three variables: the input voltage, V_{in}, and the two states, y_1 and y_2. In Figure 9 the magnitude of the derivative of the first state learned by ODENet30-RK4 is shown for two values of y_2. One can imagine these figures as snapshots taken at different positions of the state space. The magnitude of the derivative of the second state for $y_2 = 0$ can be seen in Figure 10. If the magnitude



(a) $y_2 = 0$.



(b) $y_2 = 1$.

Figure 9: Magnitude of the learned derivative of the first state of the second-order diode clipper for two different values of the second state.

of the learned derivative is asymmetrical with respect to the input voltage, the chosen value for y_2 should make it clear.

The derivative of the first state resembles the S-shape of the first-order diode clipper derivative. The clipping behavior manifests itself in regions with large derivative magnitudes, which "push" the output towards the S-shape. Furthermore, for $y_2 = 1$, the behavior of the clipper becomes asymmetrical, which corresponds to the previous analysis of the circuit in [2]. One can see it even better in Figure 10, which shows that the derivative of the second state is inherently asymmetrical with respect to the input or y_1 state. As one can see from Figures 9 and 10, the learned derivative would be challenging to derive analytically in a white-box fashion, which confirms the usefulness of the ODENet in VA modeling.

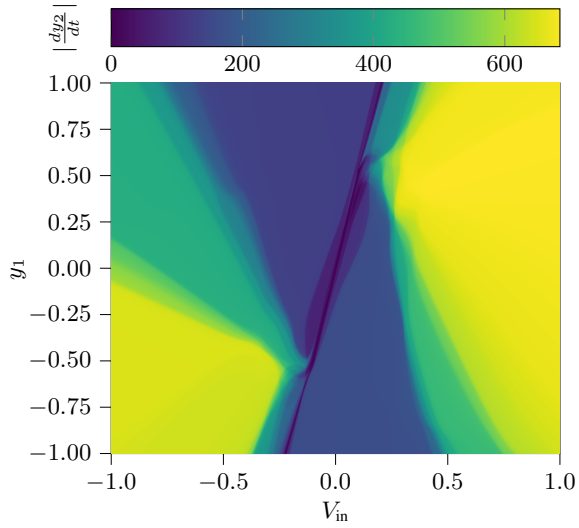


Figure 10: Magnitude of the learned derivative of the second state of the second-order diode clipper for $y_2 = 0$.

4.4. Time-Varying Models

We also applied ODENet to phaser modeling following the "toy-problem" approach from [5]. However, even with ground truth-informed dataset, ODENet was severely underfitting contrary to the baseline from [5]. There seems to be an inherent difficulty for ODENet in learning this time-varying system. This observation was confirmed in [20], where STN, which is similar in nature to ODENet, failed to fit a phaser dataset. Possible explanations and solutions of this difficulty could be investigated in future work.

4.5. Challenges

Applying ODENet to VA is not without challenges. One needs to obtain the state data to successfully train the derivative network. To know which state data to capture, some insight into the modeled device is needed as in white-box modeling. Additionally, the implementation we used was quite slow to train because ODENet allows parallelization only through the usage of minibatches.

5. CONCLUSIONS

In this paper, the concept of learning ODEs from data and solving them numerically for VA modeling was presented. We adapted this framework, termed ODENet, to handle an input signal and the initial conditions of ODEs describing analog electronic circuits. We successfully applied it to VA modeling of two distortion circuits: the first- and the second-order diode clipper. Our approach obtained comparable performance to the baseline at the sampling rate of the training set while using fewer trainable parameters and showed acceptable performance at increased sampling rates. In some cases, the accuracy increased at higher sampling rates. The learned derivative is physically interpretable but does not have to be derived analytically and does not need oversampling, which are big advantages over purely white-box models. The learned derivative may be used in conjunction with any numerical solver, which allows an accuracy-performance trade-off. Future work could in-

clude modeling systems with more states or time-varying systems, or a deeper analysis of the learned ODE-numerical solver coupling.

6. ACKNOWLEDGMENTS

The main part of the work was conducted in July–November 2021, when the first author was visiting the Aalto Acoustics Lab in Espoo, Finland. He is deeply thankful to the members of the lab for inspiring conversations and the feedback related to this work. The authors also thank anonymous reviewers for their insightful comments.

7. REFERENCES

[1] U. Zölzer, *DAFX: Digital Audio Effects*, John Wiley & Sons, Ltd, 2nd edition, 2011.

[2] J. D. Parker, F. Esqueda, and A. Bergner, "Modelling of Nonlinear State-Space Systems Using a Deep Neural Network," in *Proc. of the 22nd Int. Conf. on Digital Audio Effects (DAFx), Birmingham, UK*, 2019.

[3] F. Eichas, M. Fink, M. Holters, and U. Zölzer, "Physical Modeling of the MXR Phase 90 Guitar Effect Pedal," in *Proc. of the 17th Int. Conf. on Digital Audio Effects (DAFx), Erlangen, Germany, September 1-5*, 2014.

[4] R. Kiiski, F. Esqueda, and V. Välimäki, "Time-Variant Gray-Box Modeling of a Phaser Pedal," in *Proc. of the 19th Int. Conf. on Digital Audio Effects (DAFx), Brno, Czech Republic, September 5–9*, 2016, pp. 121–128.

[5] A. Wright and V. Välimäki, "Neural Modelling of Periodically Modulated Time-Varying Effects," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects (DAFx), Vienna, Austria, September 2020-21*, 2020.

[6] A. Wright and V. Välimäki, "Perceptual Loss Function for Neural Modelling of Audio Systems," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[7] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-Time Black-Box Modelling with Recurrent Neural Networks," in *Proc. of the 22nd Int. Conf. on Digital Audio Effects (DAFx-19), Birmingham, UK*, 2019.

[8] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-Time Guitar Amplifier Emulation with Deep Learning," *Applied Sciences*, vol. 10, no. 3, 2020.

[9] J. D. Parker, *Dispersive Systems in Musical Audio Signal Processing*, Ph.D. thesis, Aalto University, Espoo, Finland, Oct. 2013.

[10] J. Pakarinen, V. Välimäki, F. Fontana, V. Lazzarini, and J. Abel, "Recent Advances in Real-Time Musical Effects, Synthesis, and Virtual Analog Models," *EURASIP J. Adv. Sig. Proc.*, vol. 2011, January 2011.

[11] D. Yeh, J. Abel, and J. Smith, "Simulation of the Diode Limiter in Guitar Distortion Circuits by Numerical Solution of Ordinary Differential Equations," in *Proc. of the 10th Int. Conf. on Digital Audio Effects (DAFx), Bordeaux, France, September 10-15*, 2007.

[12] D. T. Yeh, J. S. Abel, A. Vladimirescu, and J. O. Smith, "Numerical Methods for Simulation of Guitar Distortion Circuits," *Computer Music Journal*, vol. 32, no. 2, pp. 23–42, 2008.

[13] J. O. Smith, *Physical Audio Signal Processing*, http://ccrma.stanford.edu/~jos/pasp/, 2010, Online book, 2010 edition. Retrieved June 18, 2021.

[14] F. Esqueda, B. Kuznetsov, and J. D. Parker, "Differentiable White-Box Virtual Analog Modeling," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects (DAFx), Vienna, Austria, September 2020-21*, 2021.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[16] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," in *Proc. of the 31st Conf. on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada*, 2018.

[17] D. Karlsson and O. Svanström, "Modelling Dynamical Systems Using Neural Ordinary Differential Equations," M.S. thesis, Chalmers University of Technology, 2019.

[18] M. Gockenbach, *Partial Differential Equations: Analytical and Numerical Methods*, Society for Industrial and Applied Mathematics, Philadelphia, 2011.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org.

[20] A. Peussa, E.-P. Damskägg, S. Mimilakis, T. Sherson, L. Juvela, A. Gotsopoulos, and V. Välimäki, "Exposure Bias and State Matching in Recurrent Neural Network Virtual Analog Models," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects (DAFx), Vienna, Austria, September 2020-21*, 2021.

[21] L. N. Smith, "A Disciplined Approach to Neural Network Hyper-parameters: Part 1 – Learning Rate, Batch Size, Momentum, and Weight Decay," *US Naval Research Laboratory Technical Report 5510-026*, 2018.

[22] J. Abeßer, P. Kramer, Ch. Dittmar, and G. Schuller, "Parametric Audio Coding of Bass Guitar Recordings using a Tuned Physical Modeling Algorithm," in *Proc. of the 16th Int. Conf. on Digital Audio Effects (DAFx), Maynooth, Ireland, September 2-5*, 2013.

[23] Ch. Kehling, J. Abeßer, Ch. Dittmar, and G. Schuller, "Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score- and Instrument-related Parameters," in *Proc. of the 17th Int. Conf. on Digital Audio Effects (DAFx), Erlangen, Germany, September 1-5*, 2014.

[24] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. of the 3rd Int. Conf. for Learning Representations, San Diego*, 2015.

[25] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," in *Proc. of the 30th Conf. on Neural Information Processing Systems (NIPS), Long Beach, CA, USA*, 2017, pp. 972–981.