POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

NEURAL DIODE CLIPPER FOR WAVE DIGITAL FILTERS

ASSIGNMENT REPORT IN
SELECTED TOPICS IN MUSIC AND ACOUSTIC ENGINEERING

Shijie Yang,

Advisor:
Prof.YEE-KING
MATTHEW JOHN

Academic year:
2022-2023

Abstract: Wave Digital Filters (WDFs), a popular explicit physical modeling method, are extensively used to emulate analog musical devices. However, most of the nonlinear elements do not have closed-form WDF solutions. In this report, we present the use of neural networks for simulating diode pairs within a WDF. We modeled various diode clipper circuits, evaluating neural network performance and developed two real-time audio plugins using different methods. Our results suggest that even a small neural network can achieve high modeling accuracy compared to SPICE simulation. These experiments demonstrate that neural networks present a promising approach to model nonlinear behavior within WDFs.

Key-words: Neural Audio Effects, Wave Digital Filter, Diode Pair, Neural Network, Diode Clipper

1. Introduction

The technique of Virtual analog (VA) modeling enables to create digital algorithms that emulate the behavior of these analog audio equipment. In physical modeling, Wave Digital Filters (WDFs) [6, 18] have been used extensively. WDFs are particularly effective for simulating the linear components of circuits, providing a high degree of modularity and desirable properties [9].
However, most nonlinear circuits pose a significant challenge due to the lack of closed-form WDF solutions. These necessitate the resolution of delay-free loops using iterative methods (Newton's method or its variants), K-method lookup tables, or functional approximation [1, 10, 11, 15, 17].
To address this challenge, many researchers propose the innovative application of machine learning to capture the behavior of the nonlinear ports in the wave domain. Rather than *the black-box* method [4, 5, 20] that emulates the circuits by analyzing and replicating only input-output behavior, the *white-box* [8] method that emulates the circuits by analyzing internal behavior and replicating their physical operation, machine learning applied to *grey-box* [2] modeling in the WDFs, which utilizes knowledge of the circuit to train neural networks within a *white-box* structure.
In this report, we use an efficient training method for diode pair that extracts its wave domain behavior. The dataset describing a nonlinearity in the Kirchhoff domain is used to train a neural network and replace the diode pair in a WDF model of a diode clipper circuit. Then, the combined neural network and WDF model will be built into an audio effect plugin that can run in real time.
The structure of this report is organized as follows. Section 2 presents some background of WDFs we need in this project. Section 3 presents the idea of merging machine learning into WDFs and a process for training neural networks to emulate the behavior of diode pair in the wave domain. Results appear in Section 4. Section 5 illustrates the real-time audio plugins implementation. Section 6 summarizes and concludes the report.

2. Wave Digital Filters Theory

WDF is a kind of physical modeling method originally devised to model a digital filter after an analog one. The concept of waves, or more specifically, traveling waves, is used to describe the interaction between different parts of a circuit. These waves can either be incident (traveling towards an element) or reflected (traveling away from an element). The reflection of waves is determined by the port resistances of the elements involved. When there's no adaptation, an instantaneous reflection can cause numerical issues, making the computation unfeasible. To avoid this, the process of "adapting" is used. When a component is adapted, the reflected wave is not instantly dependent on the incident wave, which helps in avoiding computational issues.

2.1. Parametric Wave Definition

In the Kirchhoff Domain, circuit components are described using the Kirchhoff pair: voltage v and current i. In the Wave Domain, components are characterized by the wave pair: incident wave a and reflected wave b. The wave variables and Kirchhoff pair can be interconverted based on the parametric wave definition given in [18]:

$$a = R^{\rho-1}v + R^{\rho}i, \quad b = R^{\rho-1}v - R^{\rho}i \tag{1}$$

$$v = \frac{1}{2}R^{1-\rho}a + \frac{1}{2}R^{1-\rho}b, \quad i = \frac{1}{2}R^{-\rho}a - \frac{1}{2}R^{-\rho}b \tag{2}$$

where a and b are defined as the incident and reflected wave for a given circuit port; the R is the port impedance. ρ is a wave definition parameter: "voltage waves" are defined by $\rho = 1$, "current waves" are given when $\rho = 0$, "power waves" are give when $\rho = 1/2$.

2.2. Linear One-Ports Elements in Wave Domain

Most simple circuit elements, such as resistors and capacitors, are commonly implemented as one-port elements and can be adapted, which means the current reflected wave is not related to the current incident wave. The adaption condition and wave mapping of the linear one-port elements used in this report are shown in Table 1.

Component Symbols	Constitutive Equation	Adaptation Condition	Wave Mapping
R	$v(t) = Ri(t)$	$R_0 = R$	$b[n] = 0$
V	$v(t) - v_g(t) = Ri(t)$	$R_0 = R$	$b[n] = R^{\rho-1}v_g[n]$
C	$C\frac{dv(t)}{dt} = i(t)$	$R_0 = \frac{T}{2C}$	$b[n] = a[n-1]$

Table 1: Adaptation conditions and wave mappings of common linear one-port components. From top to bottom, the components are resistors, resistive voltage sources, and capacitors, where $v_g(t)$ is an ideal voltage source, and T is the sampling time step.

2.3. Adaptors

In WDFs, the topological interconnections of the circuit are characterized by adaptors. Simple WDF adaptors such as series and parallel adaptors can be implemented generally as N-port elements. Like the linear one-port elements, adaptors can be adapted at a certain port so that the adapted port is unrelated to the current incident wave of that port. In this project, we will use 3-port parallel adaptor and 2-port series adaptor. A 3-port parallel adaptor is defined by the voltage wave relationship:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 & \frac{R_2}{R_1+R_2} & \frac{R_1}{R_1+R_2} \\ 1 & -\frac{R_1}{R_1+R_2} & \frac{R_1}{R_1+R_2} \\ 1 & \frac{R_2}{R_1+R_2} & -\frac{R_2}{R_1+R_2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \tag{3}$$

where a_n, b_n and R_n are the incident waves, reflected waves, and port impedance at a given port. Port 0 in equation (3) has been adapted by setting $R_0 = \frac{1}{1/R_1+1/R_2}$. This adaptation condition can be more intuitively written in terms of port conductances: $G_0 = G_1 + G_2$.

A 2-port series adaptor is similarly defined by:

$$\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}, \tag{4}$$

where port 0 in equation (4) has been adapted by setting $R_0 = R_1$.

3. Machine Learning for Diode Pairs in Wave Domain

Numerous pieces of analog audio equipment, especially iconic guitar pedals like the Ibanez Tube Screamer and MXR Distortion +, are fundamentally designed with a diode pair at their core. As shown in Figure 1, the op-amp gain stage of the Boss DS-1 features two 1N4148 diodes arranged in reverse parallel. These diodes are commonly employed to produce a clipping effect, resulting in sound distortion.

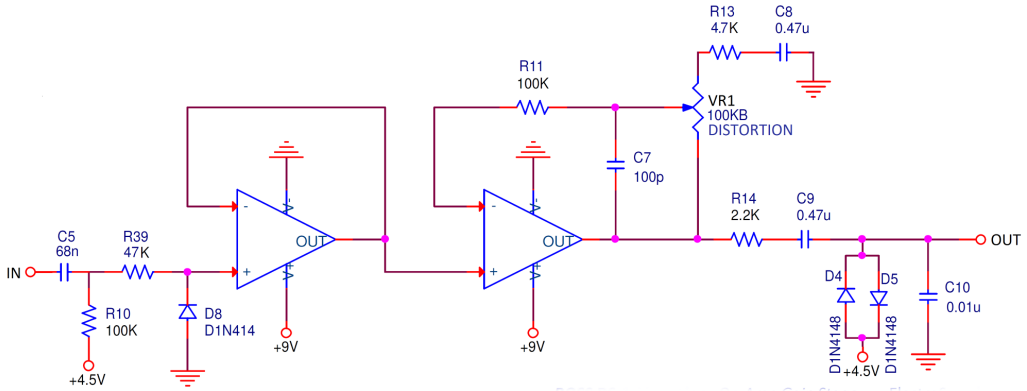


Figure 1: Op-Amp Gain Stage of Boss DS-1

3.1. Diode Pair in Kirchhoff Domain and Wave Domain

The behavior of a single diode in the Kirchhoff domain is captured by the Shockley ideal diode model:

$$i = I_s \left(e^{v/V_T} - 1 \right), \tag{5}$$

where I_s is the reverse-bias saturation current and V_T is the thermal voltage, as introduced by Shockley [16]. Building on this foundation, Werner *et al.* [19] provided an explicit model for diode pairs in the wave domain. This model, which leverages the Lambert $\mathcal{W}$ function, is given by:

$$b = a - 2\lambda V_T \left[\mu_0 \mathcal{W}\left(\frac{RI_s}{\mu_0 V_T} e^{\frac{\lambda a}{\mu_0 V_T}} \right) + \mu_1 \mathcal{W}\left(-\frac{RI_s}{\mu_1 V_T} e^{-\frac{\lambda a}{\mu_1 V_T}} \right) \right], \tag{6}$$

where $\lambda = sgn(a)$, and the factors μ_0 and μ_1 respectively represent the number of forward diodes M and reverse diodes N, and are defined as:

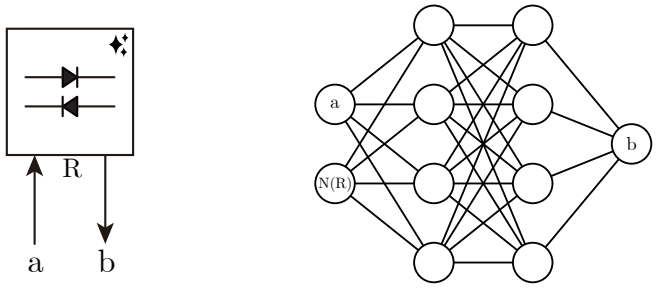
$$\mu_0 = \begin{cases} M, & a \geq 0 \\ N, & a < 0 \end{cases} \quad \text{and} \quad \mu_1 = \begin{cases} N, & a \geq 0 \\ M, & a < 0 \end{cases}. \tag{7}$$

3.2. Neural Diode Pair in Wave Domain

While we possess an explicit model for diode pairs, solving it necessitates dealing with both the exponential function and the Lambert $\mathcal{W}$ function. Notably, deriving a solution for the Lambert $\mathcal{W}$ function typically mandates iterative methods, introducing a significant performance overhead.
However, inspecting equation (6), we observe that once the diode type and the configuration are decided, the reflected wave b is solely dependent on the current incident wave a and the input impedance R. Thus, equation (6) can be reformulated using an abstract function f:

$$b = f(a, R) \tag{8}$$

Considering the lacking memory of the diode pair in the wave domain as shown in Equation (8), feedforward neural networks (FNNs) are postulated as suitable for modeling the abstract function f.
In this report, we exploit deep neural networks (DNNs), a type of FNNs, to fit the abstract function f. As an example, Figure 2a shows a diode pair in wave domain with a star symbol located at the upper right corner which denotes this model implemented via a neural network whose architecture is illustrated in Figure 2b. This is a 2x4 DNN; it takes the incident wave a and the normalized input impedance $N(R)$ as input features, where N is the normalization function applied [**?**].



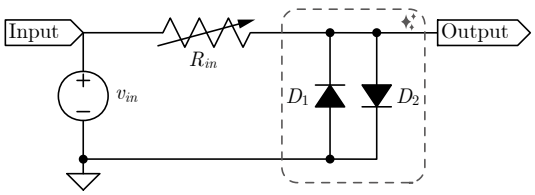
(a) (b)

Figure 2: A 2x4 DNN emulates a diode pair's behavior in the Wave Domain.

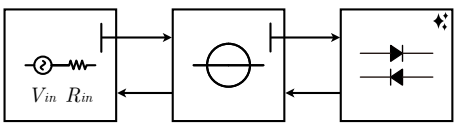
Despite the potential of DNNs in emulating the behavior of diode pairs, the training of such networks in the wave domain faces the difficulty of acquiring training data from the physical circuit because the data we can measure is in the Kirchhoff domain, not the wave domain.
To address this problem, Chowdhury *et al.* [2] suggested using the data generated by the explicit diode pair model (as described by Equation 6) to pre-train the DNN. The pre-trained DNN is then inserted into the final connection tree and fine-tuned using data collected from the Kirchhoff domain.
Chowdhury's method solves the untrainable problem since we can input and read the Kirchhoff data using the W2K Equation (1) at the target nodes in the connection tree. However, this training method is time-consuming because each update to the DNN parameters requires traversing the whole connection tree. Furthermore, fine-tuning is indispensable for achieving optimal performance as real-world diode models often deviate from the explicit model due to factors such as manufacturing processes and physical conditions.

3.3. Circuit Design for Collecting Training Data

In this report, we will use an efficient training method that caters specifically to the characteristics of diode pairs in the wave domain, allowing the DNNs to be trained directly using data collected from the Kirchhoff domain. This method takes advantage of batch processing, allowing for parallel feedforward operations which speed up the training process and get more reliable results.
With this approach, fine-tuning becomes optional as the DNNs are trained using Kirchhoff data. Our results show that our DNNs can achieve superior performance even without fine-tuning. Further details about the performance will be discussed in Section 4.
To facilitate the data collection for training, we designed a specific circuit shown in Figure 3a. This circuit is straightforward: it includes a resistive voltage source connected in series with a diode pair (the resistor of the voltage source is adjustable). The goal of the neural network is to emulate the 1-up/1-down (1U1D) diode pair configuration marked by the dashed box in Figure 3a.



(a) Circuit designed for collecting training data of diode pair. When there is a need to modify the diode type or its configuration, alterations can be made solely within the dashed box.



(b) Connection tree of training circuit in Wave domain.

Figure 3: Training circuit and its connection tree in wave domain.

Figure 3b shows the connection tree of the training circuit. Here, the resistive voltage source and diode pair are interconnected through a 2-port series adaptor. By inserting the resistive voltage source's adaption condition and the 2-port series adaptor's adaptation into the 2-port series adaptor's reflected wave mapping, allows us to derive the incident wave a_{dp} and the input impedance R_{dp} of the diode pair:

$$a_{\mathrm{dp}}[n] = v_{\mathrm{in}}[n], \tag{9}$$

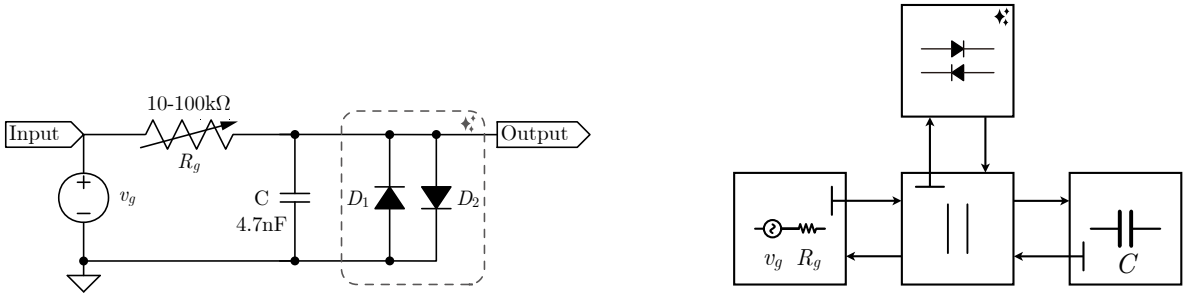
$$R_{\mathrm{dp}}[n] = R_{\mathrm{in}}[n]. \tag{10}$$

Equations (9) and (10) indicate that the reflected wave v_{in} and impedance R_{in} from the resistive voltage source can propagate directly to the diode pair. Furthermore, substituting equation (9) into W2K Equation (1) solves for the reflected wave b_{dp}, we can then derive the explicit relationship between the expected reflected wave and measured nodes voltage:

$$b_{\mathrm{dp}}[n] = 2v_{\mathrm{out}}[n] - v_{\mathrm{in}}[n]. \tag{11}$$

3.4. Dataset Generation

The data generation begins with constructing the training circuit shown in Figure 3a. We implemented the training circuits in LTspice at transient mode. The simulation time is set to 4.0s. The collected data is re-sampled to 192.0kHz and transformed into the wave domain using Equations (9) and (11).
Before activating the training circuit, v_{in} and R_{in} must be meticulously determined since they will influence the distributions and ranges of the a_{dp} and R_{dp}. The efficient selections of v_{in} and R_{in} depend on the final connection tree that will be ultimately implemented.
Now, consider the diode clipper circuit we aim to implement, as depicted in Figure 4a. Figure 4b presents its associated connection tree. Examining the signal block diagram in Figure 5, it becomes apparent that this circuit operates in two stages. The initial stage incorporates an RC low-pass filter, while the subsequent stage functions as a wave shaper, realized via the use of a diode pair.



(a) Diode clipper circuit we will emulate with 1U1D diode configuration.

(b) Connection tree of the diode clipper circuit we will finally emulate in Wave domain.

Figure 4: Circuit diagram of a diode clipper we will emulate and its connection tree in Wave domain.

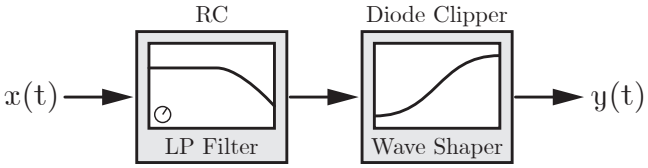


Figure 5: Signal block diagram of diode clipper circuit shown in Figure 4a.

In the wave domain, we can adapt the resistive voltage source, capacitor, and three-port parallel adaptor according to the adaption conditions specified in Section 2. By doing so, we can derive the explicit expressions for a_{dp} and R_{dp} for the diode clipper circuit illustrated in Figure 4a:

$$a_{\mathrm{dp}}[n] = \frac{2CF_{\mathrm{s}}}{1 + 2R_{\mathrm{g}}CF_{\mathrm{s}}} \left(\frac{1}{2CF_{\mathrm{s}}} v_{\mathrm{g}}[n] + R_{\mathrm{g}} a_{\mathrm{c}}[n-1] \right), \tag{12}$$

$$R_{\mathrm{dp}} = \frac{R_{\mathrm{g}}}{1 + 2R_{\mathrm{g}}CF_{\mathrm{s}}}, \tag{13}$$

where F_{s} is the sampling frequency, and $a_{\mathrm{c}}[n-1]$ is the incident wave of capacitor, C, at the previous time step.

3.4.1 Voltage Source Selection for Training Circuit

Since Equation (12) is recursive, it is difficult to explicitly determine the range of a_{dp}. Given that the total energy of the entire circuit system only comes from the resistive voltage source, let us assume that $a_{\mathrm{c}}[n-1] = v_{\mathrm{g}}[n]$. Inserting this assumption into Equation (12), solving a_{dp} and subsequently substituting it into Equation (9) to solve for v_{in}, we can get:

$$v_{\mathrm{in}}[n] = v_{\mathrm{g}}[n]. \tag{14}$$

Equation (14) indicates that the range of input signal v_{g} in the final connection tree is the same as the excitation signal v_{in} of connection tree for training circuit. After testing different ranges and signal types, we choose the $[-16, 16]$V as the range of v_{in}; models trained at data in this range can provide a 24dB gain margin (using 1V as the reference voltage).

As for the signal type of v_{in}, we employ white noise following the combination of both uniform and Gaussian distribution. Unlike authentic audio recordings, although authentic audio recordings roughly follow a Gaussian distribution, they are accompanied by too many samples falling to near zero. In our synthetic noise, the uniform noise ensures that the network is adequately trained across all regions of v_{in}, providing consistent performance throughout the entire input signal range, and the Gaussian noise further enhances the model's performance around the 0 vicinity.

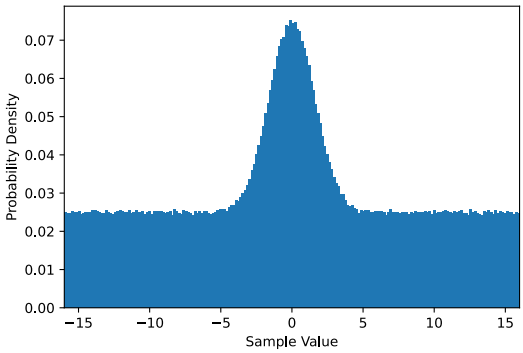
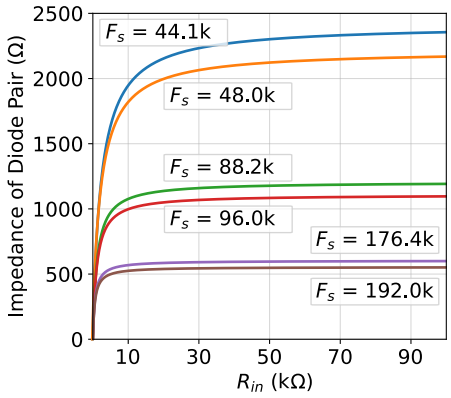


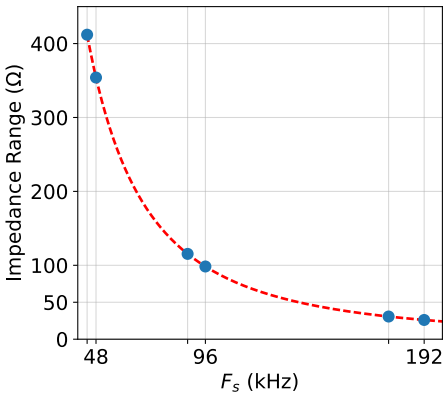
Figure 6: Distribution of input signal v_{in}

3.4.2 Variable Resistor Selection for Training Circuit

Equation (13) provides an explicit method for determining R_{dp} in the final connection tree. As shown in Figure 7a, R_{dp} exhibits an increasing trend with R_{g} and a decreasing relationship with F_{s} for a given R_{g}. Furthermore, Figure 7b illustrates that a rise in F_{s} results in a reduction in the range of R_{dp} (specifically, the difference between its maximum and minimum values). These observations suggest that, for a neural diode pair with a specific scale of network, a higher sampling frequency yields improved performance. This is due to the diode pair's input impedance exhibiting a narrower range.



(a) Characteristic curves between R_{in} (resistance of variable resistor) and R_{dp} (input impedance of diode pair in wave domain) @ standard sampling frequencies.



(b) Characteristic curve between range size of R_{dp} and sampling frequency.

Figure 7: Circuit diagram of a diode clipper we will emulate and its connection tree in wave domain.

Table 2 lists some specific values of R_{dp} at different standard F_{s} when R_{g} is set at both ends. We noticed that R_{dp} at two closing F_{s} can compose a continuous impedance range of R_{dp}. For this reason, we divide the F_{s} into three groups and train models separately. Furthermore, as denoted No.4 in 2, we also trained a model that can run at all standard sampling frequencies.

Model	F_s (kHz)	R_{dp} (Ω)		Selected R_{in} (Ω) for the model
		$R_{\text{g}} = 10k\Omega$	$R_{\text{g}} = 100k\Omega$	
1	44.1 48.0	1943.5 1814.2	2355.5 2168.3	[1700, 2500]
2	88.2 96.0	1076.3 997.6	1191.8 1096.0	[900, 1300]
3	176.4 192.0	568.8 525.0	599.5 551.0	[450, 650]
4	44.1-192.0	-	-	[450, 2500]

Table 2: R_{dp} at both ends of R_{g} @ standard F_s. F_s are divided into three groups for each model; the ranges of R_{in} preserve about 100Ω margin and are chosen for training circuits for each model.

3.4.3 Diode Models

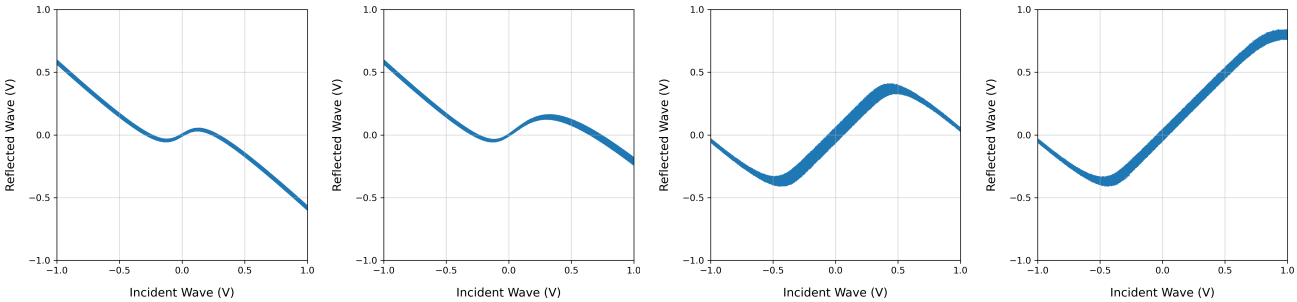
In this report, we will emulate the 1N34A and 1N4148. The 1N34A, shown in Figure 8a, is a germanium diode characterized by a low forward voltage, typically ranging between 0.2V and 0.3V. This diode is renowned for imparting a soft, thick distortion effect when incorporated into guitar distortion pedals. Its signature sound is frequently sought after to recreate the saturated tones reminiscent of 1960s amplifiers.
Conversely, the 1N4148, shown in Figure 8b, is a silicon diode with a notably higher forward voltage, generally between 0.6V and 0.7V. This elevated forward voltage enables the 1N4148 to produce a sharp, bright distortion effect in guitar pedals, delivering a more aggressive tonal edge.

(a) 1N34A (b) 1N4148

Figure 8: Diode 1N34A and 1N4148.

3.4.4 Data Check

Upon gathering the training data in LTspice and converting it into the wave domain, we can promptly examine the characteristic curves representing the relationship between incident and reflected waves. Figure 9 depicts these curves for diode pairs assembled using 1N34A and 1N4148 in both 1U1D and 1U2D configurations. These curves are similar to those given in the paper by Werner *et al.* [19].



(a) 1N34A with 1U1D (b) 1N34A with 1U2D (c) 1N4148 with 1U1D (d) 1N4148 with 1U2D

Figure 9: Incident-reflected waves characteristic scatter diagram of 1N34A and 1N4148 diode pair.

3.5. Training and Hyperparameter

For training our models, we utilize a combination of the mean square error (MSE), error-to-signal ratio (ESR), and the difference in DC offset between the target and the network's output signals. During validation, the same metrics are employed. The concepts of ESR and DC offset are inspired by the suggestions of Wright and Välimäki [21] and are defined as:

$$\mathcal{E}_{\text{ESR}} = \frac{\sum_{n=1}^{N} \left(y_t[n] - y_p[n]\right)^2}{\sum_{n=1}^{N} y_t[n]^2}, \tag{15}$$

$$\mathcal{E}_{\text{DC}} = \frac{\left|\frac{1}{N}\sum_{n=0}^{N-1}\left(y_t[n] - y_p[n]\right)\right|^2}{\frac{1}{N}\sum_{n=0}^{N-1}|y_t[n]|^2}, \tag{16}$$

where N is the number of signal samples, y_t and y_p are the target and predicted signal respectively.
For the training process, we employed the Adam optimizer, initiating with a learning rate of 1×10^{-3}. We incorporated a scheduler to decrease the learning rate if the validation loss did not show improvement over a span of 7 epochs. Our chosen batch size was 16 samples. The entirety of the training and validation for our neural networks was facilitated using PyTorch [7, 13]. We subjected the models to 200 epochs of training, leveraging the dataset outlined in Section 3.4. As for the activation function, we opted for the Exponential Linear Unit (ELU) [3].

4. Results

Table 3 presents the performance metrics of models trained to emulate the diode clipper circuit depicted in Figure 4a, specifically for the 1N4148 in a 1U1D configuration. It's evident that, after pre-training using our approach, all models, including the most compact ones, deliver commendable performance. The most impressive Mean Squared Error (MSE) listed is 6.16×10^{-6}, a benchmark set by the largest model in our lineup.
An interesting observation is the slight uptick in MSE when integrating the pre-trained model into the final connection tree. However, even the smallest network's performance remains superior compared to the explicit method leveraging the Lambert $\mathcal{W}$ function, which stands at 6.88×10^{-4} for this configuration.

Model Size	After Pre-train MSE	Before Fine-tune MSE	After Fine-tune MSE
2x8	8.85e-6	1.40e-4	5.69e-5
2x16	9.62e-6	1.41e-4	5.62e-5
3x16	6.32e-6	9.91e-5	5.30e-5
4x16	6.26e-6	9.74e-5	5.31e-5
2x32	6.43e-6	1.10e-4	5.33e-5
3x32	6.16e-6	9.82e-5	5.30e-5

Table 3: Performance metrics of models used to emulate the diode clipper circuit, configured with 1N4148 in a 1U1D configuration, working @ 44.1kHz and 48.0kHz.

Intriguingly, post fine-tuning in the final connection tree, all models converge to almost identical performance metrics. Pinpointing the exact cause of this phenomenon is earmarked for future exploration. Current speculation attributes this behavior to memory components, such as capacitors present in the final connection tree. These components might introduce high-frequency distortions due to the bilinear transform [12].
Specifically, in time domain, Figure 10 presents a comparative analysis of the outputs from three distinct models: the LTspice model (which was responsible for generating the training data), the 2x16 fine-tuned model emulating 1N4148 diode pair with 1U1D configuration (selected for real-time application and operated @ 48kHz), and the model based on the Lambert $\mathcal{W}$ function.

5. Real-Time Neural Diode Clipper Audio Plugin

For the real-time neural audio plugin implementation, we implemented two plugins using two different methods. The first approach leverages TorchScript provided by PyTorch. The second method employs the Neutone SDK [14]. Source code is available on GitHub [22]. This is a test helping us measure how much text needed to write to fill the bottom of this page.

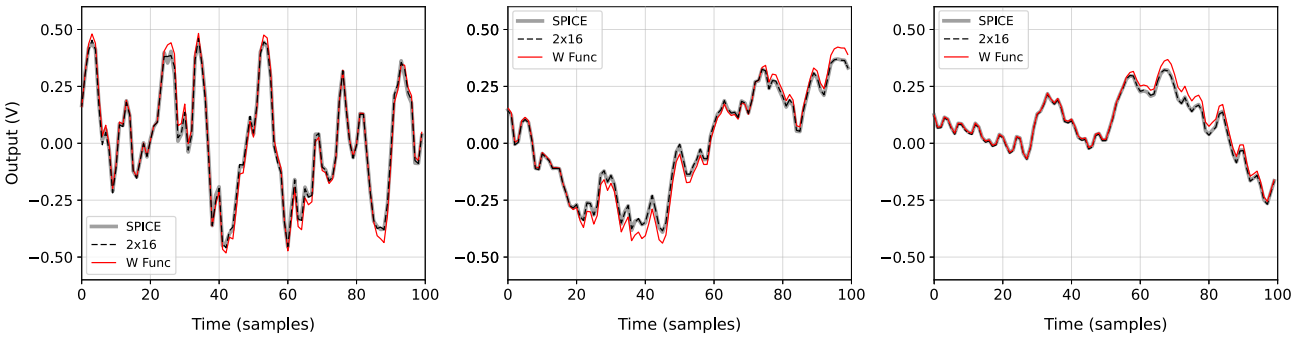


Figure 10: Comparing output of the simulation of SPICE, the 2x16 model operating @ 48.0kHz, and the model based on Lambert w function. From left to right: $R_{\text{g}} = 10.3k\Omega$, $R_{\text{g}} = 55.9k\Omega$, $R_{\text{g}} = 92.5k\Omega$.

5.1. TorchScript with PyTorch

TorchScript tool enables us to transition a trained PyTorch model from a Python environment to a static model suitable for non-Python environments, such as C++. With this conversion, the static model can be integrated into JUCE, facilitating the construction of a neural audio plugin. Given that our neural audio plugin essentially operates as a *grey-box*, this method provides an efficient division between *white-box* and *black-box* processing. Specifically, JUCE handles the white-box processing related to the WDF connection tree, while LibTorch manages the black-box processing of the diode pair in the wave domain.
Figure 11 displays the GUI of the neural diode clipper audio plugin, developed using the JUCE framework in conjunction with LibTorch. This plugin integrates four distinct models and offers six adjustable parameters:

- DIODE: Allows users to toggle between diode models, either the 1N34A or the 1N4148.
- CONFIG: Enables switching between diode pair configurations, offering choices of 1U1D or 1U2D.
- DRIVE: Controls the gain of the first stage, adjustable within the range of [-25dB, 25dB].
- TONE: Adjusts the cutoff frequency, with values ranging from [0.00, 1.00]. This parameter corresponds to the resistance of R_g which varies between [10k, 100k]Ω, as illustrated in Figure 4a.
- MIX: Ratio between the distorted and original sound.
- LEVEL: Output Gain.

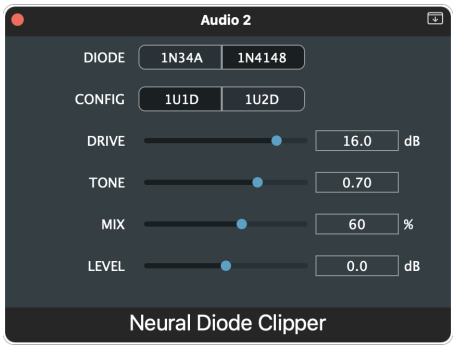


Figure 11: GUI of neural diode clipper audio plugin implemented by JUCE framework and LibTorch.

5.2. Neutone SDK

Neutone SDK offers tools to transform existing PyTorch models into versions compatible with their proprietary audio plugin. While TorchScript remains the foundational technology of the Neutone SDK, the conversion process of a trained PyTorch model to its C++ counterpart is significantly streamlined. This method is particularly advantageous for rapid model testing, eliminating the need to engage with JUCE directly. However, this convenience comes with limitations. Entrusting the entirety of the process to the Neutone SDK can pose challenges when dealing with intricate computational logic or when integrating additional techniques.
Figure 12 depicts the Neutone plugin operating within Logic Pro. This plugin is designed to load models converted using tools provided by the Neutone SDK. It offers four adjustable parameters, which can be mapped to the parameters of a neural network. These include latent features, a common parameter in timbre transformers, used to modify the style of the transformation. In our specific implementation, we solely utilize the "A" parameter, which corresponds to the "TONE" parameter as presented in Figure 11.

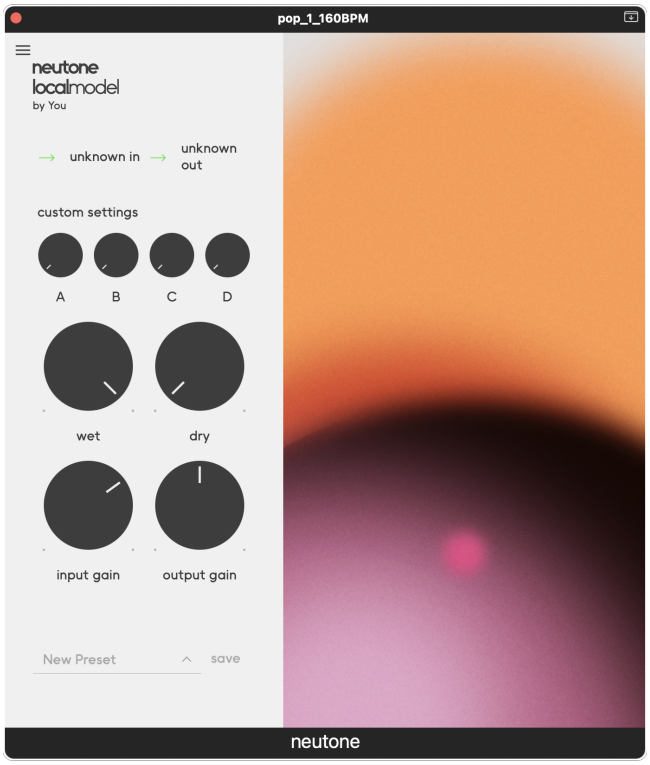


Figure 12: Neutone plugin GUI

6. Conclusions

In this report, we demonstrated the use of DNNs for simulating diode pair in wave digital filter models. We introduced a training circuit to collect data on 1N34A and 1N4148 diode pair from Kirchhoff domain simulated in LTspice. Our results demonstrate that small DNNs can learn the behavior of nonlinearities and run in real time.

7. ACKNOWLEDGMENT

I express my profound appreciation to Matthew John Yee-King for his invaluable insights on LibTorch and machine learning. Further gratitude is extended to Alberto Bernardini for his valuable contribution with pertinent materials.

References

[1] Alberto Bernardini, Alessio Vergani, and Augusto Sarti. Wave digital modeling of nonlinear 3-terminal devices for virtual analog applications. *Circuits, Systems, and Signal Processing*, 39, 07 2020. doi: 10.1007/s00034-019-01331-7.

[2] Jatin Chowdhury and Christopher Johann Clarke. Emulating diode circuits with differentiable wave digital filters. In *19th Sound and Music Computing Conference*, pages 2–9, 2022. URL: https://zenodo.org/record/6566846.

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016. arXiv:1511.07289.

[4] Eero-Pekka Damskägg, Lauri Juvela, Etienne Thuillier, and Vesa Välimäki. Deep learning for tube amplifier emulation, 2019. arXiv:1811.00334.

[5] Felix Eichas, Stephan Möller, and Udo Zölzer. Block-oriented gray box modeling of guitar amplifiers. 09 2017.

[6] A. Fettweis. Wave digital filters: Theory and practice. *Proceedings of the IEEE*, 74(2):270–327, 1986. doi:10.1109/PROC.1986.13458.

[7] FACEBOOK INC. Reducelronplateausdk, 2023. URL: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html.

[8] Daniele Masti and Alberto Bemporad. Learning nonlinear state-space models using deep autoencoders. pages 3862–3867, 12 2018. doi:10.1109/CDC.2018.8619475.

[9] Klaus Meerkötter. On the passivity of wave digital networks. *IEEE Circuits and Systems Magazine*, 18(4):40–57, 2018. doi:10.1109/MCAS.2018.2872664.

[10] Alberto Bernardini Oliviero Massi, Riccardo Giampiccolo and Augusto Sarti. Explicit vector wave digital filter modeling of circuits with a single bipolar junction transistor. In *International Conference on Digital Audio Effects (DAFx-26)*, 09/2023 2023.

[11] Michael J. Olsen, Kurt James Werner, and Julius O. Smith III. Resolving grouped nonlinearities in wave digital filters using iterative techniques. In *International Conference on Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, 09/2016 2016. . URL: http://dafx16.vutbr.cz/.

[12] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Press, USA, 3rd edition, 2009.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[14] QosmoInc. neutone sdk, 2023. URL: https://github.com/QosmoInc/neutone_sdk.

[15] Tim Schwerdtfeger and Anton Kummert. Newton's method for modularity-preserving multidimensional wave digital filters. In *2015 IEEE 9th International Workshop on Multidimensional (nD) Systems (nDS)*, pages 1–6, 2015. doi:10.1109/NDS.2015.7332656.

[16] W. Shockley. The theory of p-n junctions in semiconductors and p-n junction transistors. *The Bell System Technical Journal*, 28(3):435–489, 1949. doi:10.1002/j.1538-7305.1949.tb03645.x.

[17] Kurt Werner, Vaibhav Nangia, Julius Smith, and Jonathan Abel. Resolving wave digital filters with multiple/multiport nonlinearities. 11 2015.

[18] Kurt James Werner. Virtual analog modeling of audio circuitry using wave digital filters. Dissertation, Stanford University, December 2016.

[19] Kurt James Werner, Vaibhav Nangia, Alberto Bernardini, Julius Smith, and Augusto Sarti. An improved and generalized diode clipper model for wave digital filters. In *139th Audio Engineering Society (AES) Convention*, October 2015.

[20] Alec Wright, Eero-Pekka Damskägg, and Vesa Välimäki. Real-time black-box modelling with recurrent neural networks. 09 2019.

[21] Alec Wright and Vesa Välimäki. Perceptual loss function for neural modelling of audio systems, 2019. arXiv:1911.08922.

[22] Shijie Yang. Neural diode clipper wdfs, 2023. URL: https://github.com/mryangsj/neural-diode-clipper-wdfs.