Neural ODE based modeling of memristive circuits

Karol Bednarz, Bartłomiej Garda, Zbigniew Galias, *Senior Member, IEEE*

***Abstract*—TODO: To do.**

***Index Terms*—Memristor modeling, artificial neural network, neural ODE.**

I. Introduction

NUMEROUS memristor models have been proposed in the scientific literature following the discovery of memristive behavior at the nanoscale. The original model, introduced in [1], conceptualizes the memristor as a series connection of two variable resistances corresponding to the conductive and insulating regions of the thin film. To more accurately model Self-Directed Channel (SDC) memristors, M. Nugent and T. Molter proposed the generalized *Mean Metastable Switch* (MMS) model, which is a semi-empirical formulation. In this approach, the time derivative of the internal state variable is defined as a function of both the transition probabilities between metastable states and the current value of the state variable [2]. In [3], the authors attempted to model memristive behavior using the *Physics-Informed Neural Networks* (PINNs) framework. Although the reported results indicate the potential of this method, the study is not grounded in experimental measurements of physical memristor devices. Instead, the analysis is limited to comparisons with the outcomes of existing simulation models. Moreover, the training data employed during the neural network learning process were generated based on previously developed theoretical models, thereby limiting the ability to assess the method's accuracy in the context of real-world physical systems.

In [4], the authors introduce the concept of deep neural models in which the dynamics of the hidden state are governed by an *ordinary differential equation* (ODE). The training process is performed in an end-to-end manner, meaning that all parameters are optimized simultaneously within a single training routine. A key innovation of the proposed approach is a novel backpropagation technique, which relies on solving the corresponding adjoint ODE backward in time using *adjoint sensitivity methods*. This formulation enables efficient gradient computation and facilitates the application of neural ODEs in various architectures, including continuous-depth residual networks and generative flow-based models.

In [5], the authors extend the classical Neural ODE framework to enable the modeling of discontinuous events in continuous time—without requiring prior knowledge of the number or timing of such events. The proposed differentiable event functions allow for efficient simulation of hybrid systems, such as systems with collisions, state-switching mechanisms, or point processes. This development opens up new possibilities for modeling and training systems with discrete control inputs and non-smooth dynamics.

Manuscript received Month ??, 2025. This work was supported by the National Science Centre, Poland, under Grant 2024/53/B/ST7/03841.

The authors are with the Department of Electrical Engineering, AGH University of Science and Technology, al. Mickiewicza 30, 30–059, Kraków, Poland, (e-mail: kbednarz@agh.edu.pl).

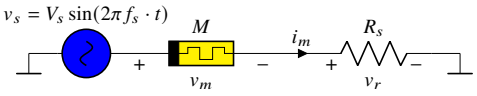

Fig. 1. Schematic diagram of the measurement setup used for the SDC memristor. The device is connected in series with a resistor, and the input voltage is applied via an arbitrary waveform generator. Voltage measurements are acquired using a data acquisition (DAQ) system.

The present study aims to investigate the feasibility of modeling Self-Directed Channel (SDC) memristors using artificial neural networks, and to compare the effectiveness of this approach with that of existing theoretical models. Specifically, we focus on the application of Neural Ordinary Differential Equation (Neural ODE) models to simulate the behavior of SDC memristors, utilizing experimental data obtained from real physical systems.

Our objective is to assess whether neural network-based models can accurately reproduce the dynamic characteristics of SDC memristors, and to determine whether they offer any advantages over traditional theoretical approaches—particularly in terms of modeling flexibility, accuracy, and applicability to real-world, measurement-driven scenarios.

II. Modeling Memristors using Neural Networks

This study utilizes experimental data obtained from Self-Directed Channel (SDC) memristors doped with tungsten. The structure and properties of these devices have been described in detail in the literature, e.g., in [6], [7].

The experimental setup, illustrated in Fig. 1, consists of an SDC memristor connected in series with a resistor. The sinusoidal input voltage is supplied by an arbitrary waveform generator, while the voltage signals are measured using a data acquisition (DAQ) system.

Measurements are carried out for various combinations of supply voltage amplitudes and frequencies. Specifically, the amplitude of the applied voltage was varied as $V_s \in \{0.5,\ 1.0,\ 1.5\}$ [V], and the frequency was selected from the set $f \in \{1,\ 5,\ 10,\ 20,\ 50,\ 100\}$ [Hz].
The analyzed model of a memristor follows the general theoretical framework of memristive devices originally introduced in [8], and is expressed as

$$i(t) = \mathcal{G}\left(\mathbf{x}(t), v(t)\right) v(t), \tag{1a}$$

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = f\left(\mathbf{x}(t), v(t)\right), \tag{1b}$$

Algorithm 1 Adaptive Loss Balancing with Clamping (Compact)

Require: Loss terms $\mathcal{L}_{\text{base}}, \mathcal{L}_{\text{vel}}, \mathcal{L}_{\text{curv}}$, constant $\varepsilon > 0$, bounds $[\lambda_{\min}, \lambda_{\max}]$
Ensure: Adaptive weights $\lambda_{\text{base}}, \lambda_{\text{vel}}, \lambda_{\text{curv}}$
1: Collect losses: $\mathbf{L} \leftarrow [\mathcal{L}_{\text{base}}, \mathcal{L}_{\text{vel}}, \mathcal{L}_{\text{curv}}]$
2: Compute geometric mean: $\overline{\mathcal{L}}_{\text{geo}} \leftarrow \exp\left(\frac{1}{|\mathbf{L}|}\sum_{i=1}^{|\mathbf{L}|} \ln(\mathcal{L}_i + \varepsilon)\right)$
3: **for** each loss $\mathcal{L}_i \in \mathbf{L}$ **do**
4: $\quad \lambda_i \leftarrow \frac{\overline{\mathcal{L}}_{\text{geo}}}{\mathcal{L}_i + \varepsilon}$
5: $\quad \lambda_i \leftarrow \text{clamp}(\lambda_i, \lambda_{\min}, \lambda_{\max})$
6: **end for**
7: **return** $(\lambda_{\text{base}}, \lambda_{\text{vel}}, \lambda_{\text{curv}})$

where $\mathbf{x}$ denotes the vector of internal state variables, whose temporal evolution is governed by the function f, and $\mathcal{G}$ represents the memductance of the device. The fundamental challenge in memristor modeling lies in accurately identifying the functional forms of $\mathcal{G}(\mathbf{x}, v)$ and $f(\mathbf{x}, v)$.

A. Objective Function

The objective function $\mathcal{L}$ to be minimized in the optimization process is based on trajectories generated by the dynamical system. It is designed to simultaneously account for both the signal values and its dynamic structure. To this end, it is defined as the sum of four components, each serving a distinct and complementary role in the optimization process

$$\mathcal{L} = \lambda_{\text{base}}\mathcal{L}_{\text{base}} + \lambda_{\text{vel}}\mathcal{L}_v + \lambda_{\text{curv}}\mathcal{L}_c + \sum_{i=1}^{N_c} \lambda_{\text{con}_i}\mathcal{L}_{\text{con}_i}(x). \quad (2)$$

The first term $\mathcal{L}_{\text{base}}$ denotes the primary error computed as the mean squared error (MSE) between predicted and reference values. $\mathcal{L}_v$ enforces accurate modeling of derivatives of state variables, $\mathcal{L}_c$ promotes consistency of the trajectories in terms of their curvature by computing MSE of second derivatives, and $\mathcal{L}_{\text{con}}(x)$ enforces fulfillment of physical or structural constraints that the model is required to satisfy, including conformity to the underlying physical model. The weighting coefficients λ_{base}, λ_{vel}, and λ_{curv} balance the influence of each term according to the priorities of the modeling task. Their values are calculated during each optimization step using the Geometric Loss Strategy (GLS) [9], [10] using the Algorithm 1.

Let us consider the sequence $(z_{k,t})_{t=1}^{T}$, where $z \in \{v, i, \dot{v}, \dot{i}, \ddot{v}, \ddot{i}\}$ is a selected variable (voltage or current) or its derivative, T is total number of samples and $k = 1, 2, \ldots, N_{\text{traj}}$ is the sequence index. The sequence is normalized utilizing the standard deviation to ensure statistical consistency within each dataset. The trajectory-specific standard deviation is calculated using the unbiased estimator

$$\sigma_{z,k} = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}\left(z_{k,t}^{\text{true}} - \bar{z}_k^{\text{true}}\right)^2}, \quad (3)$$

where the trajectory temporal mean is defined as $\bar{z}_k^{\text{true}} = \frac{1}{T}\sum_{t=1}^{T} z_{k,t}^{\text{true}}$. This standardization approach enhances the numerical stability and convergence properties of optimization algorithms employed in neural network training procedures.

By implementing trajectory-specific normalization rather than global standardization, the methodology preserves the inherent temporal dynamics and statistical properties unique to each circuit configuration while simultaneously ensuring that amplitude disparities do not introduce systematic bias during predictive modeling procedures. This approach is particularly advantageous in scenarios involving heterogeneous circuit topologies or varying operational conditions, where maintaining the relative temporal structure is paramount for accurate system identification and dynamic analysis.

The primary component of the loss function is responsible for model fitting to experimental voltage-current data:

$$\mathcal{L}_{\text{base}} = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\left[\text{MSE}\left(\frac{v_{\text{pred},k}}{\sigma_{v,k}}, \frac{v_{\text{true},k}}{\sigma_{v,k}}\right) + \text{MSE}\left(\frac{i_{\text{pred},k}}{\sigma_{i,k}}, \frac{i_{\text{true},k}}{\sigma_{i,k}}\right)\right], \quad (4)$$

where $v_{\text{pred},k}$ and $i_{\text{pred},k}$ represent the predicted voltage and current for trajectory k, while $v_{\text{true},k}$ and $i_{\text{true},k}$ denote the corresponding experimental measurements. The mean squared error between two sequences $(\hat{y}_i)_{i=1}^{n}$ and $(y_i)_{i=1}^{n}$ of n data points is defined as $\text{MSE}(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$.

The first-order dynamics comparison $\mathcal{L}_v$ enables evaluation of temporal derivative agreement between model predictions and experimental data, particularly critical for capturing rapid memristor state transitions

$$\mathcal{L}_v = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\left[\text{MSE}\left(\frac{\frac{\mathrm{d}v}{\mathrm{d}t}_{\text{pred},k}}{\sigma_{\dot{v},k}}, \frac{\frac{\mathrm{d}v}{\mathrm{d}t}_{\text{true},k}}{\sigma_{\dot{v},k}}\right) + \text{MSE}\left(\frac{\frac{\mathrm{d}i}{\mathrm{d}t}_{\text{pred},k}}{\sigma_{\dot{i},k}}, \frac{\frac{\mathrm{d}i}{\mathrm{d}t}_{\text{true},k}}{\sigma_{\dot{i},k}}\right)\right], \quad (5)$$

The second-order comparison term $\mathcal{L}_c$ addresses trajectory curvature characteristics

$$\mathcal{L}_c = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\left[\text{MSE}\left(\frac{\frac{\mathrm{d}^2v}{\mathrm{d}t^2}_{\text{pred},k}}{\sigma_{\ddot{v},k}}, \frac{\frac{\mathrm{d}^2v}{\mathrm{d}t^2}_{\text{true},k}}{\sigma_{\ddot{v},k}}\right) + \text{MSE}\left(\frac{\frac{\mathrm{d}^2i}{\mathrm{d}t^2}_{\text{pred},k}}{\sigma_{\ddot{i},k}}, \frac{\frac{\mathrm{d}^2i}{\mathrm{d}t^2}_{\text{true},k}}{\sigma_{\ddot{i},k}}\right)\right] \quad (6)$$

This component facilitates superior representation of curvature properties within the phase-space trajectory, particularly important for modeling systems exhibiting complex nonlinear and highly dynamic responses.

To enforce parameter bounds within specified intervals, a sophisticated constraint loss function utilizing smooth sigmoid transitions is implemented. This component prevents parameter drift beyond physically meaningful ranges while maintaining differentiability for gradient-based optimization algorithms. The constraint loss is mathematically expressed as:

$$\mathcal{L}_{\text{con}}(x) = \mathbb{E}\left[\left(\sigma\left(-\tfrac{x-a}{f}\right)(a-x) + \sigma\left(\tfrac{x-b}{f}\right)(x-b)\right)^2\right], \quad (7)$$

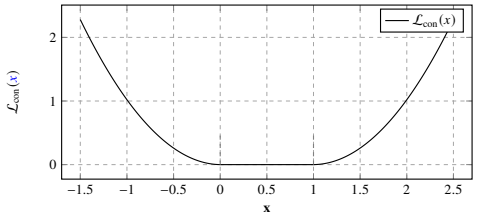

Fig. 2. The constraint function $\mathcal{L}_{\text{con}}(x)$ for $a = 0$, $b = 1$, $f = 0.0025$.

where $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid activation function, a and b denote the lower and upper bounds of the admissible parameter range, respectively, f is a scaling factor controlling the steepness of the transition near the bounds, and $\mathbb{E}[\cdot]$ denotes the expected value (ensemble average over samples or trajectories). The term $\sigma(-(x-a)/f)(a-x)$ penalizes values of x below the lower bound a, while the term $\sigma((x-b)/f)(x-b)$ penalizes values of x above the upper bound b. Squaring ensures that the penalty is non-negative and grows with the violation magnitude.

The operational characteristics of the constraint function $\mathcal{L}_{\text{con}}(x)$ are illustrated in Figure 2 for the parameter configuration: $a = 0$, $b = 1$, $f = 0.0025$ TODO: (verify f) over the domain $x \in (-0.5, 1.5)$. The function $\mathcal{L}_{\text{con}}(x)$ exhibits minimal penalty in the region $x \in (0, 1)$ and progressively increasing penalties as the parameter x exceeds the boundary limits.

The following constraint intervals are applied to memristor model parameters for the SingleNN-MemODE approach:

- $R_{\text{ON}} \in (0, R_{\text{OFF}})$ kΩ,
- $R_{\text{OFF}} \in (R_{\text{ON}}, 200)$ kΩ,
- $x(t) \in (0, 1)$,

and for the Dual-NN-MemODE approach:

- $\mathcal{G} \in (0, 2)$mS.

As the state variable $\mathbf{x}(t)$ in the Dual-NN-MemODE approach is not restricted to a specific range, no explicit constraints are applied to it. TODO: What is $x(t)$ for the Dual-NN-MemODE?—DONE

The upper bound for the OFF-state resistance R_{OFF} and bounds for the conductance parameter G are based on empirical measurements. Remaining constraints are consequences of physically meaningful values of parameters and variables.

TODO: Is the objective function common for all models? If so, perhaps section "C. Objective Function" could be placed before section "A. Neural ODE based Memristor Modeling". —DONE

B. Neural ODE based Memristor Modeling

TODO: I suggest to change the acronym for the first method to "SingleNN-MemODE". What is your opinion?—DONE – I was wondering Det - for deterministic equation, however this might be misleading

In this study, two neural network-based modeling approaches are proposed. The first one, denoted as **SingleNN-MemODE**, employs a deterministic formulation of the memristor conductance $\mathcal{G}$, while the function f defining the dynamics of internal variables is modeled using a neural network. In the second one, referred to as **Dual-NN-MemODE**, both $\mathcal{G}$ and f are modeled using neural networks.

For the SingleNN-MemODE approach, the function f is implemented using an artificial neural network with a single output. For the Dual-NN-MemODE approach the number of outputs which is equal to the number of internal variables may be larger than 1. For both cases, the neural network consists of interconnected nodes (neurons) that process information through weighted connections. Each neuron in the network computes its output according to:

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{8}$$

where w_i are the weights that determine the strength of each input connection x_i, b is the bias term that provides an adjustable threshold for neuron activation, and σ is the activation function (e.g., sigmoid, ReLU, or tanh) that introduces nonlinearity into the model. During the optimization process, the network parameters (weights w_i and biases b) are tuned jointly with the values of R_{ON} and R_{OFF}.

The learning process involves iteratively adjusting the network parameters to minimize the loss function that quantifies the difference between predicted and target outputs. This is achieved through backpropagation algorithm combined with gradient descent optimization, where gradients of the loss function with respect to each parameter are computed and used to update the weights and biases in the direction that reduces the overall error.

In the first approach (SingleNN-MemODE), the memristor conductance is defined on the basis of the physical mechanisms underlying self-directed channel (SDC) devices. In such memristors, the resistance evolves as a result of Ag^+ ion migration and the dynamic formation of conductive filaments. Consequently, the device conductance exhibits a continuous transition between distinct resistance states. This transition can be described as a weighted combination of the limiting resistance values, modulated by the internal variable. The relation between the internal variable and the memristor conductance has the form TODO: Removed definition of $M(x)$, is it needed?—DONE – I think it is not needed.

$$G_m(x) = \frac{x}{R_{\text{ON}}} + \frac{1-x}{R_{\text{OFF}}}, \tag{9}$$

where R_{ON} and R_{OFF} denote the resistances in the low-conductance and high-conductance states, respectively.

In the second approach (Dual-NN-MemODE), the memristor conductance is modeled using the second neural network. This neural network takes the current state $x(t)$ as input and produces the corresponding conductance $G_m(x)$ as output. In this approach, the memristor state variable $\mathbf{x}$ is not constrained to a single dimension, enabling a more flexible representation of the device's internal dynamics. Moreover, it is not restricted to a physically interpretable range, such as the interval $[0, 1]$ in Eq. (9).

Conceptual architectures are illustrated in Fig. 3 for the SingleNN-MemODE and in Fig. 4 for the Dual-NN-MemODE.

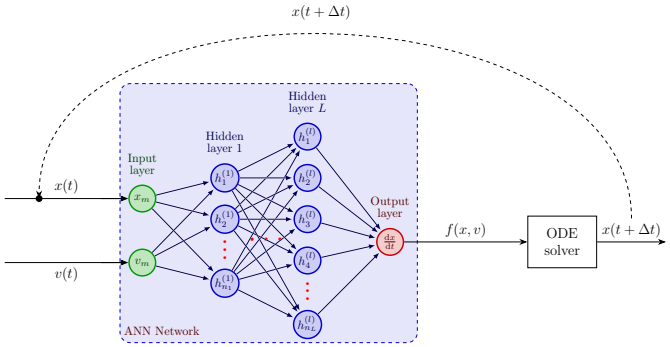

Fig. 3. Conceptual diagram illustrating the use of a neural network to simulate the dynamics of the memristor, for the SingleNN-MemODE, with schematic representation of the Neural ODE framework, where the artificial neural network (ANN) evaluates the function $f(x,v)$, which is then integrated by the ODE solver to predict the state at the next time step $x(t+\Delta t)$. The dashed feedback loop indicates the recurrent nature of the process, where the output state is fed back as input for subsequent predictions. TODO: Replace f_x by f in the figure.—DONE

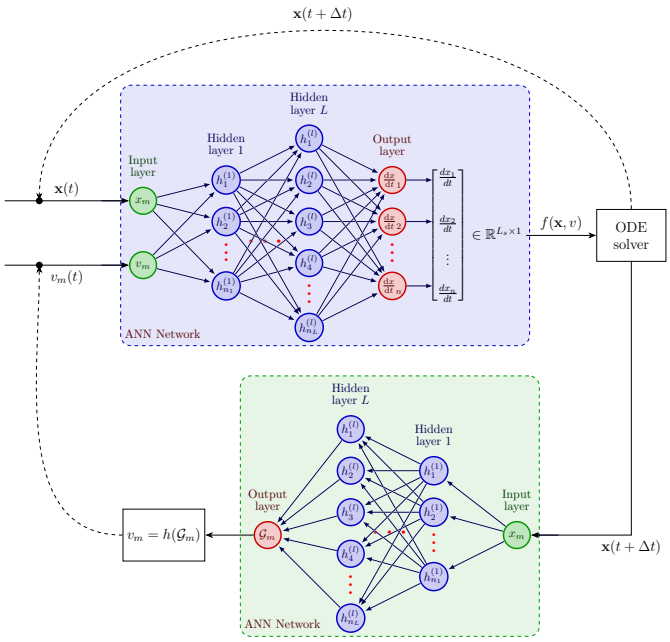

Fig. 4. Conceptual diagram illustrating the use of a neural network to simulate the dynamics of the memristor, for the Dual-NN-MemODE, with schematic representation of the Neural ODE framework, where the artificial neural network (ANN) evaluates the function $f(x,v)$, which is then integrated by the ODE solver to predict the state at the next time step $x(t+\Delta t)$. Additionally, a second ANN models the memristor conductance $G_m(x)$ based on the current state $x(t)$. TODO: Replace f_x by f in the figure.—DONE TODO: Description needed for the second ANN.—DONE

C. Neural Network Architecture

The neural network training is implemented using the `PyTorch` deep learning framework [11], within the `Python` programming environment, utilizing its automatic differentiation capabilities. To enable seamless integration of ordinary differential equations (ODEs) within the neural network training paradigm, the specialized `torchdiffeq` library is employed. This library provides differentiable ODE solvers that facilitate efficient gradient computation with respect to solution trajectories through the adjoint sensitivity method [4], [5], [12]. This approach enables end-to-end training of neural differential equation models by maintaining gradient flow through the numerical integration process, which is essential for learning dynamics of systems governed by ODEs.

Specifically, the `dopri5` (Dormand-Prince 5th order) adaptive Runge-Kutta solver is utilized for its superior balance between computational efficiency and numerical accuracy. This solver employs error estimation for adaptive step-size control, ensuring stable integration of the memristor dynamics while maintaining computational tractability during training.

The neural network architecture, illustrated in Figure 5, employs an expansion–compression (diamond-shaped) topology that incorporates a double reduction in dimensionality in both the initial and final hidden layers. This architectural configuration substantially decreases the overall number of trainable parameters and helps to mitigate vanishing and exploding gradient issues commonly encountered in deep architectures processing temporal sequences.

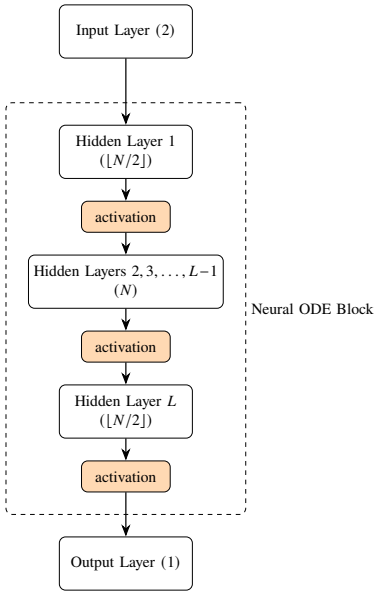

Fig. 5. Architecture of the feedforward neural network used within the Neural ODE framework for memristor dynamics modeling. The network consists of an input layer receiving two inputs, multiple hidden layers with decreasing-increasing-decreasing neuron counts ($\lfloor N/2 \rfloor, N, \ldots, N, \lfloor N/2 \rfloor$), activation functions between layers, and a single output. The Neural ODE block encompasses the core computational layers that approximate the function $f(x(t), v(t))$ of the ODE system.

Several techniques are tested for the optimization of network weights and the `Adam` (Adaptive Moment Estimation) optimizer is selected due to its superior performance. The choice of an optimizer and its parameters is guided by hyperparameter tuning conducted with the `Optuna` framework [13]. The hyperparameter search space comprises the learning rate, weight decay, and the batch size, or architecture parameters like the width or the depth of the neural network. The optimal configuration was selected based on test loss minimization. This approach focuses on minimizing the loss on the test

set, ensuring that the model generalizes well to unseen data. TODO: What is "validation loss minimization"?—DONE

D. Existing Memristor Models for Comparison

To evaluate the effectiveness of the proposed models, their performance is compared with the performance of two existing memristor models: the Mean Metastable Switch (MMS) model [14] and the Generalized Mean Metastable Switch Memristor (GMMS) model [2], [15].

The MMS model is a simplified representation of memristor's behavior, focusing on the average switching characteristics rather than the detailed dynamics. It captures essential features by modeling the device as a two-state system, where the resistance can switch between a high-resistance state (HRS) and a low-resistance state (LRS) based on the applied voltage. This model describes the memristor dynamics as

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{1}{\tau}\left(f_{\mathrm{ON}}(t)(1-x) - f_{\mathrm{OFF}}(t)x\right), \tag{10}$$

$$f_{\mathrm{ON}}(t) = \sigma\left(-\beta(v(t) - V_{\mathrm{ON}})\right), \tag{11}$$

$$f_{\mathrm{OFF}}(t) = 1 - \sigma\left(\beta(v(t) - V_{\mathrm{OFF}})\right), \tag{12}$$

where $\sigma(z) = (1 + e^{-z})^{-1}$, $\beta = \frac{q}{kT}$, k is the Boltzmann constant, T is the absolute temperature, q represents the elementary charge, while V_{ON} and V_{OFF} denote the switching voltages to LRS and HRS states, respectively. The instantaneous conductance of the memristor is defined in (9).

In the GMMS model the memristor is represented as a parallel connection of memory-dependent element and a Schottky diode [2], [15]. This formulation extends the original metastable switch concept by incorporating nonlinear current–voltage characteristics, enabling a more accurate representation of memristor's behavior across a wide range of operating conditions. The evolution of the state variable is governed by (10), while the current–voltage relationship is

$$\begin{aligned} i_{\mathrm{m}}(t) &= \phi i_{md}(v,x) + (1-\phi) i_d(v), \\ i_d(v) &= \alpha_f e^{\beta_f v} - \alpha_r e^{-\beta_r v}, \\ i_{md}(v,x) &= \mathcal{G}_m(x) v, \end{aligned} \tag{13}$$

where $\phi \in [0,1]$ is a parameter representing the ratio of the memory dependent current i_{md} and total current i_{m} flowing through the device. The parameters α_f, β_f, α_r, and β_r are positive constants characterizing the forward and reverse current behavior along the Schottky barrier. TODO: Change of notations. i_{md} - memory dependent current, previously i_{m}, i_{m} - the total current, previously i. Please confirm whether this is ok.—I think this is appropriate, I've using original notation from paper before

The inclusion of additional parameters and nonlinearities in the GMMS model increases model's complexity and poses challenges for parameter optimization.An additional challenge arises due to the Schottky effect, which introduces nonlinearity into the current–voltage characteristic. When the memristor is connected in series with a resistor, this requires solving a set of differential-algebraic equations (DAEs) of the form

$$\begin{cases} v_{\mathrm{s}}(t) = v_{\mathrm{r}}(t) + v_{\mathrm{m}}(t), \\ \dfrac{\mathrm{d}x}{\mathrm{d}t} = \dfrac{1}{\tau}\left(f_{\mathrm{ON}}(t)(1-x) - f_{\mathrm{OFF}}(t)x\right), \end{cases} \tag{14}$$

where $v_{\mathrm{s}}(t)$ denotes the supply voltage, $v_{\mathrm{r}}(t) = R_{\mathrm{s}} i(t)$ is the voltage across the resistor, and $v_{\mathrm{m}}(t)$ is the voltage across the memristor.

Since the existing models considered involve multiple parameters, a hybrid optimization framework is implemented to find the parameter set that minimizes the discrepancy between model predictions and experimental data. The optimization process consists of two stages, combining global exploration with local refinement. In the first stage, a global optimization algorithm is employed to systematically explore the parameter space and identify promising regions. Specifically, the Adaptive Differential Evolution with radius-limited sampling algorithm, available in the `BlackBoxOptim` package [16] for the `Julia` programming language, was utilized to efficiently search for candidate solutions. In the second stage, local refinement was performed using the L-BFGS-B algorithm [17], as implemented in the `Optim.jl` library. This hybrid optimization strategy effectively balances exploration and exploitation, thereby enhancing convergence towards the optimal parameter set while mitigating the risk of premature stagnation in suboptimal regions of the parameter space.

III. Results

During the preliminary experiments, a range of neural network architectures was systematically evaluated in order to identify the most suitable configuration for modeling memristor dynamics. The evaluation considered different numbers of hidden layers, neurons per layer, and optimization settings, with the aim of achieving a balance between model accuracy, generalization capability, and computational efficiency. Among these settings, the batch size determines the number of training samples processed before the model parameters are updated, thereby influencing both the convergence speed and the stability of the learning process. In this study, the batch size corresponds to the number of trajectories processed in parallel before each parameter update. The learning rate, on the other hand, controls the step size of each parameter update during training. An excessively high learning rate may cause the optimization process to diverge, while too low a value can result in slow convergence or trapping in local minima. TODO: The notions of the batch size and learning rates should be explained.—DONE
TODO: Sizes of training and validation datasets should be given.—DONE

In this study, the training datasets consist 14 trajectories obtained from experimental measurements, and the validation datasets comprise 4 trajectories, fixed across all optimization experiments to ensure consistent evaluation of model performance, which was following pairs of supply voltage V_s and frequency f_s: (1.5 V, 5 Hz), (1 V, 1 Hz), (1 V, 100 Hz), and (0.5 V, 5 Hz). The trajectories used for training and validation were selected to cover a diverse range of memristor operating conditions, thereby enhancing the model's ability to generalize across different dynamic regimes. A representative learning curve corresponding to the SingleNN-MemODE architecture, consisting of two hidden layers with 128 neurons each, trained

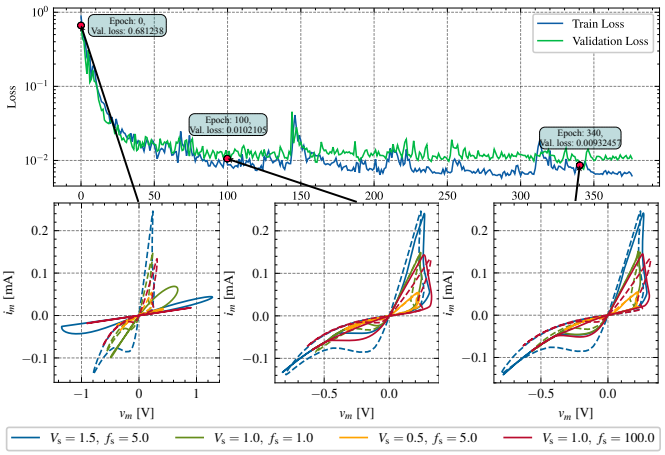

Fig. 6. Learning curve for the SingleNN-MemODE neural network architecture during training over 360 epochs. The insets illustrate the evolution of the $v_m - i_m$ hysteresis loops at selected epochs, where the dashed lines correspond to the measured responses and the solid lines denote the model predictions.

using the `Adam` optimizer with a learning rate of 10^{-2} and a batch size of 2, is presented in Figure 6. The training process was conducted over 360 epochs, with the objective function $\mathcal{L}$ monitored on both the training and validation datasets to provide insights into convergence behavior and potential overfitting. The learning curve demonstrates a consistent decrease in values of $\mathcal{L}$ for both datasets, which indicates effective optimization and a satisfactory model fitting to the experimental data. The close agreement between the training and validation loss further confirms the generalization ability of the selected architecture.

In addition to the learning curve, Figure 6 also presents the v_m–i_m hysteresis characteristics obtained at selected stages of the optimization process. These snapshots highlight how the model progressively refines its internal representation of the device dynamics during training. The gradual improvement of the hysteresis fit across epochs illustrates the capability of the neural network to capture the nonlinear and history-dependent behavior of the memristor, thereby validating the suitability of the chosen architecture and training configuration.

Representative simulation results demonstrating the neural network model's predictive capabilities are presented in Figure 7 through comparative analysis of a tungsten-doped memristor device subjected to sinusoidal voltage excitation with an amplitude of 1.5 V and frequency of 100 Hz. The analysis encompasses the voltage waveform, the current waveform and the voltage-current hysteresis relationships. The voltage profile confirms the fidelity of the input stimulus. The temporal current response reveals the device's instantaneous electrical behavior and switching kinetics. The voltage-current hysteresis loops ($v_m - i_m$) illustrate the fundamental memory properties that define memristive behavior.

The hysteresis loops are particularly diagnostic of memristor performance, as their shape, area, and switching thresholds directly reflect the underlying ionic transport mechanisms and structural modifications responsible for resistive switching. The pinched hysteresis characteristic observed at the origin serves as a definitive signature of memristive behavior, while

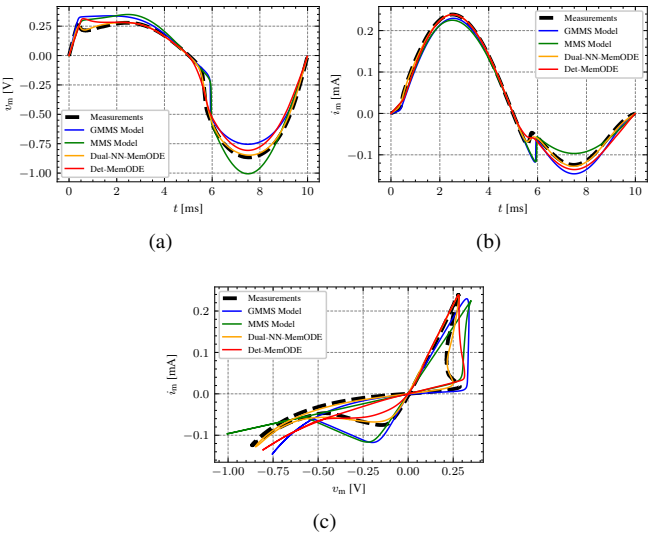

Fig. 7. Simulation results for tungsten-doped memristor dynamics under sinusoidal voltage excitation (amplitude: 1.5 V, frequency: 100 Hz); (a) memristor's voltage waveform, (b) memristor's current response, (c) Voltage-current hysteresis loop ($v_m - i_m$) illustrating the characteristic pinched behavior and memory properties fundamental to memristive operation.

the loop area quantifies the energy dissipation associated with switching events. These features enable comprehensive validation of the neural network model's ability to capture both the static and dynamic aspects of memristor operation.

A. Comparative Analysis with the MMS and GMMS Models

To assess the performance of the neural ODE approach, a comprehensive comparative analysis is conducted between the developed neural network models and the existing models of SDC memristors: the Mean Metastable Switch (MMS) model and the Generalized Mean Metastable Switch (GMMS) model.

This comparison is particularly significant as the MMS and GMMS models have been extensively validated against experimental data and serve as a benchmark for memristor circuit simulation in the scientific literature [18], [15]. The comparative evaluation encompasses temporal voltage and current evolution, hysteretic characteristics, and quantitative loss function metrics to provide comprehensive assessment of modeling fidelity.

Comparison results are plotted in Fig. 8. Quantitative analysis reveals substantial performance improvements achieved by the neural network approach.

Table I presents a comparative analysis of the loss functions $\mathcal{L}_{\text{base}}$ obtained for the proposed and existing models, offering a quantitative assessment of their performance. The results highlight the neural network's enhanced capability to reproduce complex and subtle aspects of memristor dynamics that are insufficiently captured by the phenomenological MMS/GMMS framework. For clarity, the highest loss value in each row is marked in red, whereas the lowest is marked in green. Overall, the neural network model consistently achieves substantially lower loss values, demonstrating superior accuracy and fidelity in representing the device behavior.

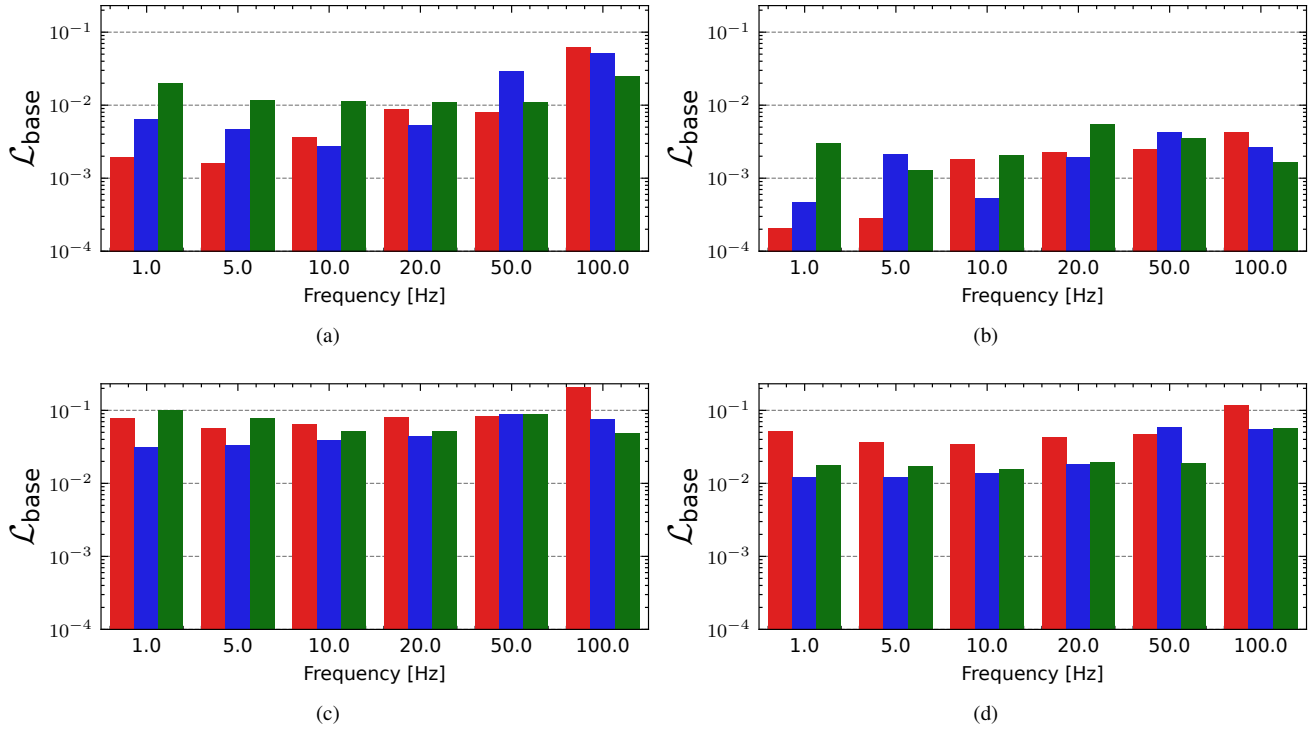

Fig. 8. Comparative analysis of loss function values: (a) the SingleNN-MemODE model, (b) the Dual-NN-MemODE model, (c) the MMS model, (d) the GMMS model. Results for voltage amplitudes $V_s = 0.5, 1.0, 1.5$ V are plotted in red (■), blue (■) and green (■), respectively.

TABLE I
COMPARATIVE $\mathcal{L}_{\text{base}}$ LOSS FUNCTION ANALYSIS.

f_s [Hz]	Det-MemODE	Dual-NN-MemODE	GMMS Model	MMS Model
Loss function comparison for $V_s = 0.5$ V				
1	1.92×10^{-3}	2.07×10^{-4}	5.21×10^{-2}	7.86×10^{-2}
5	1.60×10^{-3}	2.85×10^{-4}	3.70×10^{-2}	5.59×10^{-2}
10	3.64×10^{-3}	1.83×10^{-3}	3.43×10^{-2}	6.44×10^{-2}
20	8.69×10^{-3}	2.27×10^{-3}	4.24×10^{-2}	7.96×10^{-2}
50	7.86×10^{-3}	2.47×10^{-3}	4.71×10^{-2}	8.29×10^{-2}
100	6.24×10^{-2}	4.21×10^{-3}	1.18×10^{-1}	2.10×10^{-1}
Loss function comparison for $V_s = 1$ V				
1	6.40×10^{-3}	4.69×10^{-4}	1.20×10^{-2}	3.08×10^{-2}
5	4.58×10^{-3}	2.11×10^{-3}	1.19×10^{-2}	3.37×10^{-2}
10	2.74×10^{-3}	5.27×10^{-4}	1.36×10^{-2}	3.93×10^{-2}
20	5.32×10^{-3}	1.90×10^{-3}	1.84×10^{-2}	4.43×10^{-2}
50	2.88×10^{-2}	4.26×10^{-3}	5.86×10^{-2}	8.91×10^{-2}
100	5.08×10^{-2}	2.63×10^{-3}	5.45×10^{-2}	7.45×10^{-2}
Loss function comparison for $V_s = 1.5$ V				
1	1.99×10^{-2}	2.95×10^{-3}	1.77×10^{-2}	9.95×10^{-2}
5	1.17×10^{-2}	1.26×10^{-3}	1.73×10^{-2}	7.88×10^{-2}
10	1.14×10^{-2}	2.08×10^{-3}	1.56×10^{-2}	5.24×10^{-2}
20	1.09×10^{-2}	5.51×10^{-3}	1.96×10^{-2}	5.09×10^{-2}
50	1.09×10^{-2}	3.46×10^{-3}	1.87×10^{-2}	8.90×10^{-2}
100	2.44×10^{-2}	1.65×10^{-3}	5.63×10^{-2}	4.87×10^{-2}

TABLE II
COMPARATIVE $\mathcal{L}_{\text{base}}$ LOSS FUNCTION ANALYSIS FOR TEST DATASET.

	Det-MemODE	Dual-NN-MemODE	GMMS Model	MMS Model
$\mathcal{L}_{\text{base}}$	1.82×10^{-2}	8.28×10^{-3}	4.19×10^{-2}	5.30×10^{-2}

Table II reports the average loss function values $\mathcal{L}_{\text{base}}$ calculated on the test dataset for the proposed and existing models, providing a quantitative evaluation of their generalization performance across unseen memristor dynamics. The best result of 8.82×10^{-3} is obtained for the Dual-NN-MemODE model. The above two times reduction of the average values of the loss function for both proposed method when compared to existing methods demonstrates the enhanced modeling capability achieved through the neural ODE framework.

The superior performance of the neural network model can be attributed to its capacity for learning complex nonlinear mappings directly from experimental data, whereas the MMS model relies on predetermined phenomenological relationships that may not fully capture the device-specific dynamics and material-dependent switching characteristics inherent in experimental memristor devices.

B. Hyperparameter Sensitivity Analysis

TODO: It might be of interest to potential readers :-).
Hyperparameter optimization was conducted to systematically explore the parameter space and identify configurations that maximize the predictive performance of the neural network.

The optimization employed the Tree-structured Parzen Estimator (TPE) algorithm, implemented in the `Optuna` framework [19]. The TPE algorithm models the objective function probabilistically by constructing two separate density estimators: one for configurations associated with above-average performance and another for those yielding below-average results. By sampling preferentially from the former, while still maintaining diversity through exploration of the latter, the algorithm achieves an efficient balance between exploitation of promising regions and exploration of the broader search space.

The search space encompassed both architectural and training-related hyperparameters. On the architectural side, the number of hidden layers was varied between one and five, with layer widths ranging from 16 to 512 neurons in increments of 16. Candidate activation functions included widely used nonlinearities such as ReLU, GELU, SiLU, tanh, ELU, leaky ReLU, and sigmoid, as well as identity and sine functions. For the output layer, the choice was restricted to linear, tanh, sigmoid, ReLU. The training-related search space included the learning rate, sampled on a logarithmic scale between 10^{-4} and 10^{-1}, batch size ranging from 1 to 18, optimizer type (Adam, SGD, AdamW, Nadam, AdaBelief), and additional parameters related to regularization and scheduling, including patience, cooldown, reduction factor, tolerance, and weight decay.

To quantify the relative influence of individual hyperparameters on model performance, a surrogate-based importance analysis was performed. An XGBoost gradient boosting model [20] was trained on the explored configurations to predict validation loss. The surrogate achieved an R^2 value of 0.99, indicating high fidelity in capturing the dependence of model performance on hyperparameter selection. Importance was first estimated using the *gain* metric, which reflects the cumulative improvement in predictive accuracy attributable to each hyperparameter across the ensemble. Normalization ensured that scores summed to unity, allowing for direct interpretation as relative contributions to performance variability.

As a complementary approach, permutation importance [21] was applied to the surrogate model. In this procedure, the values of each hyperparameter were permuted at random, and the resulting degradation in predictive accuracy was recorded. Repeated ten times for each feature, the mean error increase provided an alternative measure of importance, which, unlike gain-based metrics, also accounts for nonlinear dependencies and interactions among hyperparameters.

Both sensitivity analysis methods consistently identified the *MLPG depth*—that is, the depth of the neural network used to approximate $\mathcal{G}(x)$—and the *gradient clipping value* as the dominant hyperparameters. According to the gain-based measure, their relative importances were 0.07 and 0.56, respectively, whereas the permutation-based analysis yielded corresponding values of 0.16 and 0.20. Collectively, these two parameters accounted for approximately 50% of the observed performance variance. In contrast, the choice of activation function exhibited only a marginal effect (importance score $\leqslant 0.01$), suggesting that the Neural ODE architecture demonstrates a high degree of robustness with respect to this factor within the examined range.

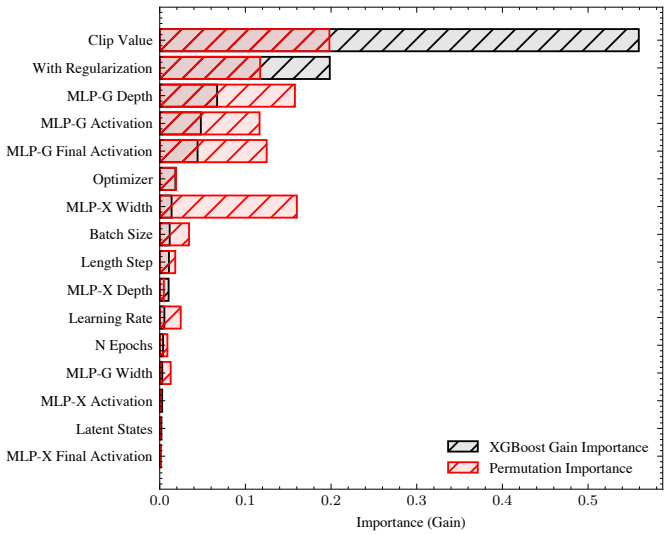

Fig. 9. Hyperparameter importance for the Dual-NN-MemODE optimization task. Scores were computed using permutation importance applied to an XG Boost surrogate model trained on 200 hyperparameter configurations. Larger values indicate greater relative influence of the corresponding hyperparameter on model performance.

TODO: What is the importance of optimizer type (mentioned in the search space).—DONE
TODO: It would be interesting to know what are the obtained values of the loss function when using different values of certain hyperparameters (other parameters should be ones used for computations).—That would be interesting to explore.

References

[1] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008. [Online]. Available: http://dx.doi.org/10.1038/nature06932

[2] T. W. Molter and M. A. Nugent, "The generalized metastable switch memristor model," in *CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications*, 2016, pp. 1–2.

[3] Y. Lee, K. Kim, and J. Lee, "A compact memristor model based on physics-informed neural networks," *Micromachines*, vol. 15, no. 2, 2024. [Online]. Available: https://www.mdpi.com/2072-666X/15/2/253

[4] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, Montréal, Canada, 2018.

[5] R. T. Q. Chen, B. Amos, and M. Nickel, "Learning neural event functions for ordinary differential equations," in *International Conference on Learning Representations (ICLR 2021)*, Vienna, Austria, 2021.

[6] B. Garda and K. Bednarz, "Comprehensive study of SDC memristors for resistive ram applications," *Energies*, vol. 17, no. 2, p. 467, Jan. 2024. [Online]. Available: http://dx.doi.org/10.3390/en17020467

[7] K. A. Campbell, "Self-directed channel memristor for high temperature operation," *Microelectronics Journal*, vol. 59, pp. 10–14, Jan. 2017. [Online]. Available: http://dx.doi.org/10.1016/j.mejo.2016.11.006

[8] L. Chua and S. M. Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976. [Online]. Available: http://dx.doi.org/10.1109/PROC.1976.10092

[9] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018, pp. 7482–7491. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00781

[10] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research,

J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 794–803. [Online]. Available: https://proceedings.mlr.press/v80/chen18a.html

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[12] R. T. Q. Chen, "torchdiffeq," 2018. [Online]. Available: https://github.com/rtqichen/torchdiffeq

[13] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD'19. ACM, Jul. 2019, pp. 2623–2631. [Online]. Available: http://dx.doi.org/10.1145/3292500.3330701

[14] L. Minati, L. Gambuzza, W. Thio, J. Sprott, and M. Frasca, "A chaotic circuit based on a physical memristor," *Chaos, Solitons & Fractals*, vol. 138, p. 109990, Sep. 2020. [Online]. Available: http://dx.doi.org/10.1016/j.chaos.2020.109990

[15] V. Ostrovskii, P. Fedoseev, Y. Bobrova, and D. Butusov, "Structural and parametric identification of knowm memristors," *Nanomaterials*, vol. 12, no. 1, p. 63, Dec. 2021. [Online]. Available: http://dx.doi.org/10.3390/nano12010063

[16] R. Feldt, "Blackboxoptim.jl: Black-box optimization for Julia," 2013–2025. [Online]. Available: https://github.com/robertfeldt/BlackBoxOptim.jl

[17] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1145/279232.279236

[18] K. Bednarz and B. Garda, "Measurement and modeling of self-directed channel (SDC) memristors: An extensive study," *Energies*, vol. 17, no. 21, p. 5400, Oct. 2024. [Online]. Available: http://dx.doi.org/10.3390/en17215400

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.

[20] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. ACM, Aug. 2016, pp. 785–794. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785

[21] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btq134