Neural ODE based modeling of memristive circuits

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

***Abstract*—The development of modern memristive technologies requires accurate dynamic models to enable reliable simulation and prototyping in applications such as resistive RAM, neural networks, and neuromorphic computing. Existing symbolic models often exhibit limitations in accurately capturing the complex dynamic behavior of physical devices across various operating conditions, impacting their reliability in simulations. This work proposes a neural-network-based modeling approach for an SDC memristor using Neural Ordinary Differential Equation (Neural ODE) models. Prior neural-network-based studies were limited by training data derived solely from theoretical models, preventing reliable assessment on real devices. The proposed method leverages a backpropagation technique based on solving the corresponding adjoint ODE backward in time using adjoint sensitivity methods. Experimental data obtained directly from fabricated SDC memristors are used to train and validate the model. Results show that the Neural ODE approach accurately reproduces the device's dynamic characteristics and provides improved flexibility and robustness compared to traditional theoretical models. These findings demonstrate the feasibility of Neural ODE–based frameworks for data-driven modeling of physical memristive devices.**

***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]. While existing symbolic models can accurately simulate memristor behavior under narrowly defined input conditions (e.g., specific signal shape, amplitude, or frequency), their generalization capability and reliability significantly diminish when the parameters of the driving signal are altered. This limitation is particularly challenging for applications like modeling memristors in chaotic oscillators, where current/voltage undergoes continuous and highly dynamic changes. 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.

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

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).

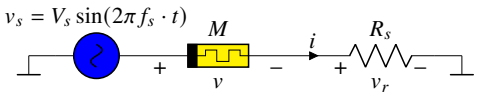

Fig. 1. Schematic diagram of the measurement setup used for the SDC memristor. The device is connected in series with a resistor, and the input voltage is applied via an arbitrary waveform generator. Voltage measurements are acquired using a data acquisition (DAQ) system. TODO: In this figure and in several other figures v_m should be replaced by v and i_m should be replaced by i.—Done

sinusoidal input voltage is supplied by an arbitrary waveform generator.e voltage signals are measured using a data acquisition (DAQ) system. Measurements are carried out for the following combinations of supply voltage amplitudes V_s and frequencies f: $V_s \in \{0.5,\ 1.0,\ 1.5\}$ [V], $f \in \{1,\ 5,\ 10,\ 20,\ 50,\ 100\}$ [Hz].

The analyzed model of a memristor follows the general theoretical framework of memristive devices originally introduced in [8], and is expressed as

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

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

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

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

The first term $\mathcal{L}_{\text{b}}$ 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 two sequences $(z_{k,t})_{t=1}^{T}$ and $(\tilde{z}_{k,t})_{t=1}^{T}$, where $z \in \{v, i, \dot{v}, \dot{i}, \ddot{v}, \ddot{i}\}$ is a selected variable (voltage or current) or its derivative obtained from measurements, $\tilde{z}$ is the predicted value of z, T is the number of samples and $k = 1, 2, \ldots, N_{\text{traj}}$ is the sequence index. The sequences are normalized utilizing the

Algorithm 1 Adaptive Loss Balancing with Clamping

Require: Loss terms $\mathcal{L}_{\text{b}}, \mathcal{L}_{\text{v}}, \mathcal{L}_{\text{c}}$, $\varepsilon > 0$, bounds $[\lambda_{\min}, \lambda_{\max}]$
Ensure: Adaptive weights $\lambda_{\text{base}}, \lambda_{\text{vel}}, \lambda_{\text{curv}}$

1: Compute the geometric mean:
$\overline{\mathcal{L}} \leftarrow \exp\left(\frac{1}{3}\left(\ln(\mathcal{L}_{\text{b}}+\varepsilon) + \ln(\mathcal{L}_{\text{v}}+\varepsilon) + \ln(\mathcal{L}_{\text{c}}+\varepsilon)\right)\right)$
2: $\lambda_{\text{base}} \leftarrow \text{clamp}\left(\frac{\overline{\mathcal{L}}}{\mathcal{L}_{\text{b}}+\varepsilon}, \lambda_{\min}, \lambda_{\max}\right)$
3: $\lambda_{\text{vel}} \leftarrow \text{clamp}\left(\frac{\overline{\mathcal{L}}}{\mathcal{L}_{\text{v}}+\varepsilon}, \lambda_{\min}, \lambda_{\max}\right)$
4: $\lambda_{\text{curv}} \leftarrow \text{clamp}\left(\frac{\overline{\mathcal{L}}}{\mathcal{L}_{\text{c}}+\varepsilon}, \lambda_{\min}, \lambda_{\max}\right)$
5: **return** $(\lambda_{\text{base}}, \lambda_{\text{vel}}, \lambda_{\text{curv}})$

standard deviation to ensure statistical consistency within each dataset. The trajectory-specific standard deviation is calculated using the unbiased estimator

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

where the trajectory temporal mean is defined as $\bar{z}_k = \frac{1}{T}\sum_{t=1}^{T} z_{k,t}$.

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 varying operational conditions, where maintaining the relative temporal structure is paramount for accurate system identification and dynamic analysis. The normalized mean squared error between two sequences $(z_{k,t})_{t=1}^{T}$ and $(\tilde{z}_{k,t})_{t=1}^{T}$ of T data points is defined as

$$\text{NMSE}(z_k, \tilde{z}_k) = \text{MSE}\left(\frac{z_k}{\sigma_{z,k}}, \frac{\tilde{z}_k}{\sigma_{z,k}}\right) = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{z_{k,t}}{\sigma_{z,k}} - \frac{\tilde{z}_{k,t}}{\sigma_{z,k}}\right)^2. \tag{4}$$

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

$$\mathcal{L}_{\text{b}} = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\left(\text{NMSE}(v_k, \tilde{v}_k) + \text{NMSE}(i_k, \tilde{i}_k)\right), \tag{5}$$

where $\tilde{v}_k$ and $\tilde{i}_k$ represent the predicted voltage and current for the kth trajectory, while v_k and i_k denote the corresponding experimental measurements.

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

$$\mathcal{L}_v = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\left(\text{NMSE}(\dot{v}_k, \tilde{\dot{v}}_k) + \text{NMSE}(\dot{i}_k, \tilde{\dot{i}}_k)\right). \tag{6}$$

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

$$\mathcal{L}_c = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\left(\text{NMSE}(\ddot{v}_k, \tilde{\ddot{v}}_k) + \text{NMSE}(\ddot{i}_k, \tilde{\ddot{i}}_k)\right). \tag{7}$$

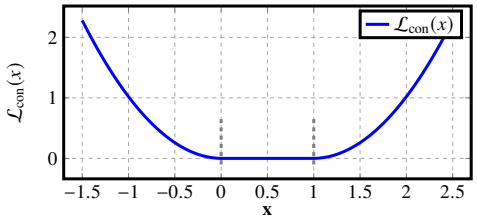

Fig. 2. The constraint function $\mathcal{L}_{\text{con}}(x)$ for $a = 0$, $b = 1$, $f = 0.0025$.

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:

$$\mathcal{L}_{\text{con}}(x) = \mathbb{E}\left[\sigma\left(-\tfrac{x-a}{f}\right)(a-x)^2 + \sigma\left(\tfrac{x-b}{f}\right)(x-b)^2\right], \quad (8)$$

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 $\mathbb{E}[\times]$ 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: $a = 0$, $b = 1$, $f = 0.0025$ 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.

B. Neural ODE based Memristor Modeling

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 **DualNN-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 DualNN-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:

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right), \quad (9)$$

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

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

and for the DualNN-MemODE approach:

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

As the state variables $\mathbf{x}(t)$ in the DualNN-MemODE approach are not restricted to a specific range, no explicit constraints are applied to them.

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

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

$$\mathcal{G}_m(x) = \frac{x}{R_{\text{ON}}} + \frac{1-x}{R_{\text{OFF}}}, \quad (10)$$

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

In the second approach (DualNN-MemODE), the memristor conductance is modeled using the second neural network. This neural network takes the current state $\mathbf{x}$ as the input and produces the corresponding conductance $\mathcal{G}_m(\mathbf{x})$ as the output. In this approach, the memristor state variable $\mathbf{x}$ is not constrained to a single dimension, enabling a more flexible

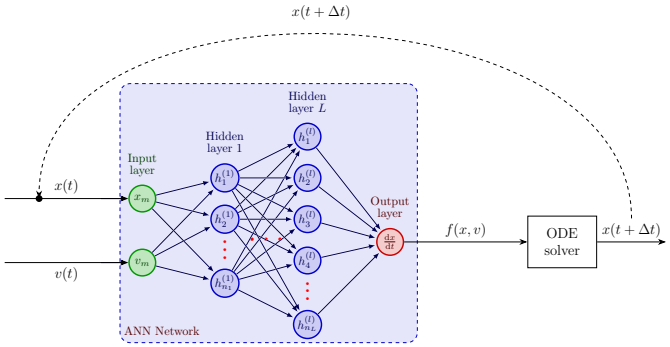

Fig. 3. Conceptual diagram illustrating the use of a neural network to simulate the dynamics of the memristor, for the SingleNN-MemODE, with schematic representation of the Neural ODE framework, where the artificial neural network (ANN) evaluates the function $f(x,v)$, which is then integrated by the 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.

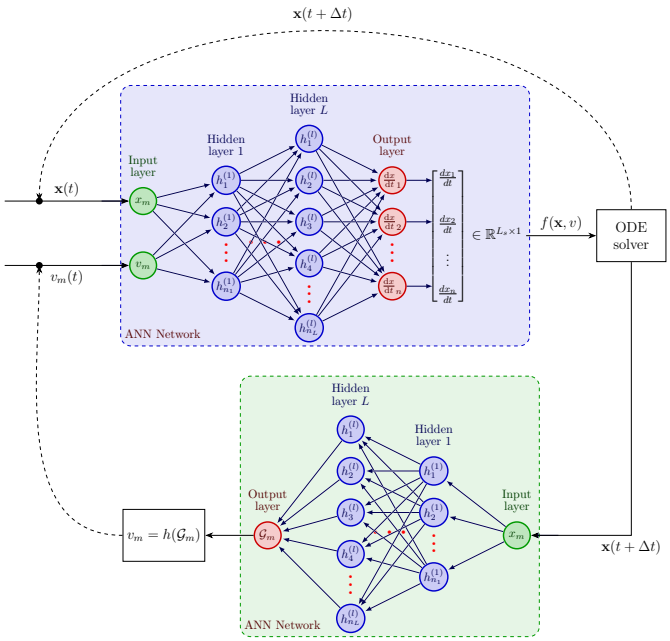

Fig. 4. Conceptual diagram illustrating the use of a neural network to simulate the dynamics of the memristor, for the DualNN-MemODE, with schematic representation of the Neural ODE framework, where the artificial neural network (ANN) evaluates the function $f(\mathbf{x},v)$, which is then integrated by the ODE solver to predict the state at the next time step $\mathbf{x}(t+\Delta t)$. The second ANN models the memristor conductance $G_m(\mathbf{x})$ based on the next state $\mathbf{x}(t+\Delta t)$.

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. (10).

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

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.

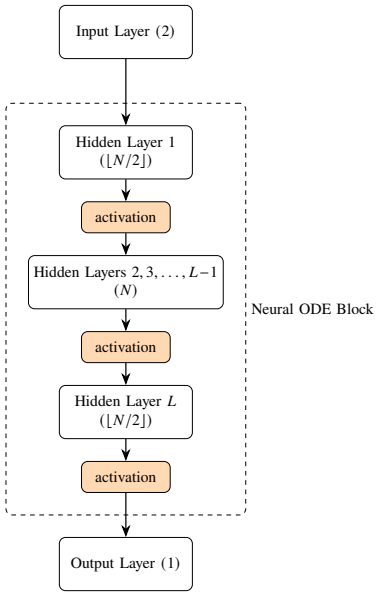

Fig. 5. Architecture of the feedforward neural network used within the Neural ODE framework for memristor dynamics modeling. The network consists of an input layer receiving two inputs, multiple hidden layers with decreasing-increasing-decreasing neuron counts ($\lfloor N/2 \rfloor, N, \ldots, N, \lfloor N/2 \rfloor$), 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 is selected based on minimizing the loss on the validation set, ensuring that the model generalizes well to unseen data. TODO: Hyperparameter values for both methods and for each ANN should be specified: the number of layers, the number of neurons in each layer, the total number of neurons, and any other which are important.—It will be added to the section III

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 describing how the device transitions between TODO: "a two-state system", this is not a two state system—Done a high-resistance state (HRS) and a low-resistance state (LRS) based on the applied voltage. This model describes the memristor dynamics as

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{1}{\tau}\left(f_{\mathrm{ON}}(t)(1-x) - f_{\mathrm{OFF}}(t)x\right), \tag{11}$$

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

$$f_{\mathrm{OFF}}(t) = 1 - \sigma\left(\beta(v(t) - V_{\mathrm{OFF}})\right), \tag{13}$$

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 (10).

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 (11), while the current–voltage relationship is

$$\begin{aligned} i(t) &= \phi i_{\mathrm{md}}(v,x) + (1-\phi) i_{\mathrm{d}}(v), \\ i_d(v) &= \alpha_{\mathrm{f}} e^{\beta_{\mathrm{f}} v} - \alpha_{\mathrm{r}} e^{-\beta_{\mathrm{r}} v}, \\ i_{\mathrm{md}}(v,x) &= \mathcal{G}_{\mathrm{m}}(x) v, \end{aligned} \tag{14}$$

where $\phi \in [0,1]$ is the parameter representing the ratio of the memory dependent current i_{md} and total current i 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.

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

$$\begin{cases} v_{\mathrm{s}}(t) = v_{\mathrm{r}}(t) + v(t), \\ \dfrac{\mathrm{d}x}{\mathrm{d}t} = \dfrac{1}{\tau}\left(f_{\mathrm{ON}}(t)(1-x) - f_{\mathrm{OFF}}(t)x\right), \end{cases} \tag{15}$$

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 $v(t)$ 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, is utilized to efficiently search for candidate solutions. In the second stage, local refinement is 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

TODO: If possible the following information should be provided in this section for each method, including the existing models: the learning time, the evaluation time needed to generate a single trajectory.—DONE

During the hyperparameter tuning, a range of neural network architectures was systematically evaluated in order to identify the most suitable configuration for modeling memristor dynamics. The evaluation considered the number of hidden layers, the number of neurons per layer, and optimization settings (e.g. batch size, learning rate), with the aim of achieving a balance between model accuracy, and generalization capability, as described in the section II-C. 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 may result in slow convergence or trapping in local minima.

In this study, the dataset consists of 18 trajectories obtained from measurements for all combinations of supply voltage amplitudes $V_{\mathrm{s}} \in \{0.5\,\mathrm{V}, 1.0\,\mathrm{V}, 1.5\,\mathrm{V}\}$ and frequencies $f \in \{1\,\mathrm{Hz}, 5\,\mathrm{Hz}, 10\,\mathrm{Hz}, 20\,\mathrm{Hz}, 50\,\mathrm{Hz}, 100\,\mathrm{Hz}\}$, where each corresponding trajectory consist 100 periods of the signal, sampled in that way that each period is represented by 1000 samples. TODO: The numbers of periods and samples per

trajectory should be specified.—DONE This dataset is divided into the training dataset consisting of 14 trajectories and the validation datasets comprising 4 trajectories. The trajectories used for training and validation are selected to cover a diverse range of memristor's operating conditions, thereby enhancing the model's ability to generalize across different dynamic regimes. The validation dataset contains the following pairs of supply voltages and frequencies: $(1.5\,\mathrm{V}, 5\,\mathrm{Hz})$, $(1\,\mathrm{V}, 1\,\mathrm{Hz})$, $(1\,\mathrm{V}, 100\,\mathrm{Hz})$, and $(0.5\,\mathrm{V}, 5\,\mathrm{Hz})$.

During the hyperparameter optimization stage, the optimal configuration of the SingleNN-MemODE architecture was found to comprise four hidden layers with the structure $64 \rightarrow 128 \rightarrow 128 \rightarrow 64$. The model was trained using the `AdamW` optimizer with a learning rate of approximately 1.89×10^{-2} and a batch size of 2. The swish activation function was employed for the hidden layers, while the final layer utilized a linear activation function.

For the DualNN-MemODE architecture, the optimal neural network configuration responsible for computing $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ consisted of four hidden layers with dimensions $120 \rightarrow 240 \rightarrow 240 \rightarrow 120$ with 6 latent states. The auxiliary neural network responsible for computing $\mathcal{G}$ included one hidden layer, containing 80 neurons. Both networks were trained using the `AdaBelief` optimizer with a learning rate of approximately 6.30×10^{-3} and a batch size of 14. The first neural network employed the gelu activation function in the hidden layers and a linear activation in the output layer, whereas the second network used the relu activation function with a sigmoid output activation.

The median training duration for the SingleNN-MemODE architecture is approximately 11 minutes 21 seconds, while the DualNN-MemODE architecture requires approximately 13 minutes 51 seconds when utilizing a single CPU (Intel i9-9900X). For comparative analysis, the median computational time necessary for parameter optimization of the MMS model is approximately 20.4 minutes, whereas the GMMS model requires approximately 8.4 minutes on identical hardware. The neural network architectures were trained employing a standard deep learning framework based on gradient descent optimization algorithms. Conversely, the parameter identification methodology for the analytical models (MMS and GMMS) necessitated a two-stage optimization protocol. Initially, a computationally intensive global search algorithm was executed across an extensive parameter space to obtain an approximate yet robust initial parameter estimate. Subsequently, a local refinement procedure employing gradient-based optimization was applied. This methodology reflects the inherent computational complexity of determining appropriate initial parameters for analytical models and accounts for their extended calibration duration relative to the single-iteration training cost of neural network-based architectures. The computational time for trajectory generation comprising 18 trajectories using the MMS model is approximately 0.0474 seconds, whereas the GMMS model requires approximately 0.0510 seconds. In contrast, the DualNN-MemODE architecture necessitates approximately 3.9664 seconds, while the SingleNN-MemODE architecture requires approximately 5.5620 seconds to generate an equivalent trajectory ensemble. The neural network models were integrated numerically using a relative error tolerance of 10^{-3} and an absolute error tolerance of 10^{-6}.

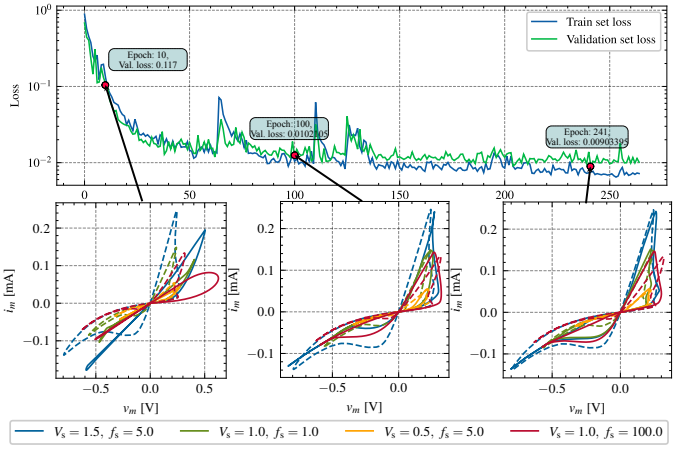

Fig. 6. Learning curve for the SingleNN-MemODE neural network architecture during training over 360 epochs. The insets illustrate the evolution of the $v - i$ hysteresis loops at selected epochs, where the dashed lines correspond to the measured responses and the solid lines denote the model predictions. TODO: replace "Train loss" by "Training set loss" and "Validation loss" by "Validation set loss"—DONE

A representative learning curve corresponding to the SingleNN-MemODE architecture, is presented in Figure 6. The training process is 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. The loss function defined in Eq. (2) was examined during both the training and testing stages. In particular, the effectiveness of optimizing the full loss function (2) was compared against optimizing only the base loss $\mathcal{L}_\mathrm{b}$, defined in Eq. (5). The results indicate that incorporating the additional terms $\mathcal{L}_\mathrm{v}$ and $\mathcal{L}_\mathrm{c}$ significantly improves convergence and enhances the overall predictive performance of the model, as reflected by reduced validation loss values and higher accuracy on unseen data. For instance, when optimizing only the base loss of the SingleNN-MemODE architecture, the validation value is $\mathcal{L}_\mathrm{b} = 2.24\times10^{-2}$, whereas optimizing the full loss function yields a lower value of $\mathcal{L}_\mathrm{b} = 1.85\times10^{-2}$.

In addition to the learning curve, Figure 6 also presents the v–i 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.

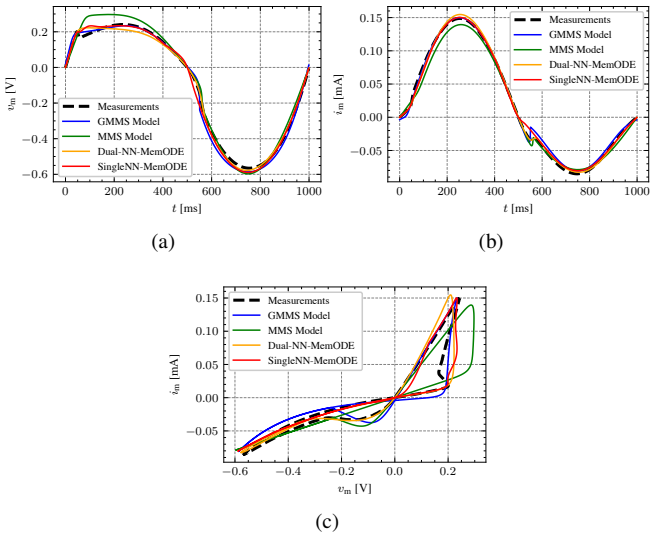

Fig. 7. Simulation test results for tungsten-doped memristor dynamics under sinusoidal voltage excitation (amplitude: 1.0 V, frequency: 1.0 Hz); (a) memristor's voltage waveform, (b) memristor's current response, (c) Voltage-current hysteresis loop ($v_m - i_m$) illustrating the characteristic pinched behavior and memory properties fundamental to memristive operation. TODO: Use black dahsed lines instead of black solid lines in the legends.—DONE

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 ($v - i$) 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 loop, which intersects at the origin, serves as a definitive signature of memristive behavior, while 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 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.

TABLE I
COMPARATIVE ANALYSIS OF THE LOSS FUNCTION $\mathcal{L}_b$.

Loss function values for V_s = 0.5 V				
f_s [Hz]	SingleNN-MemODE	DualNN-MemODE	GMMS Model	MMS Model
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 values for V_s = 1 V				
f_s [Hz]	SingleNN-MemODE	DualNN-MemODE	GMMS Model	MMS Model
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 values for V_s = 1.5 V				
f_s [Hz]	SingleNN-MemODE	DualNN-MemODE	GMMS Model	MMS Model
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}

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 $\mathcal{L}_b$ 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.

Table II reports the average loss function values $\mathcal{L}_b$ 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 DualNN-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

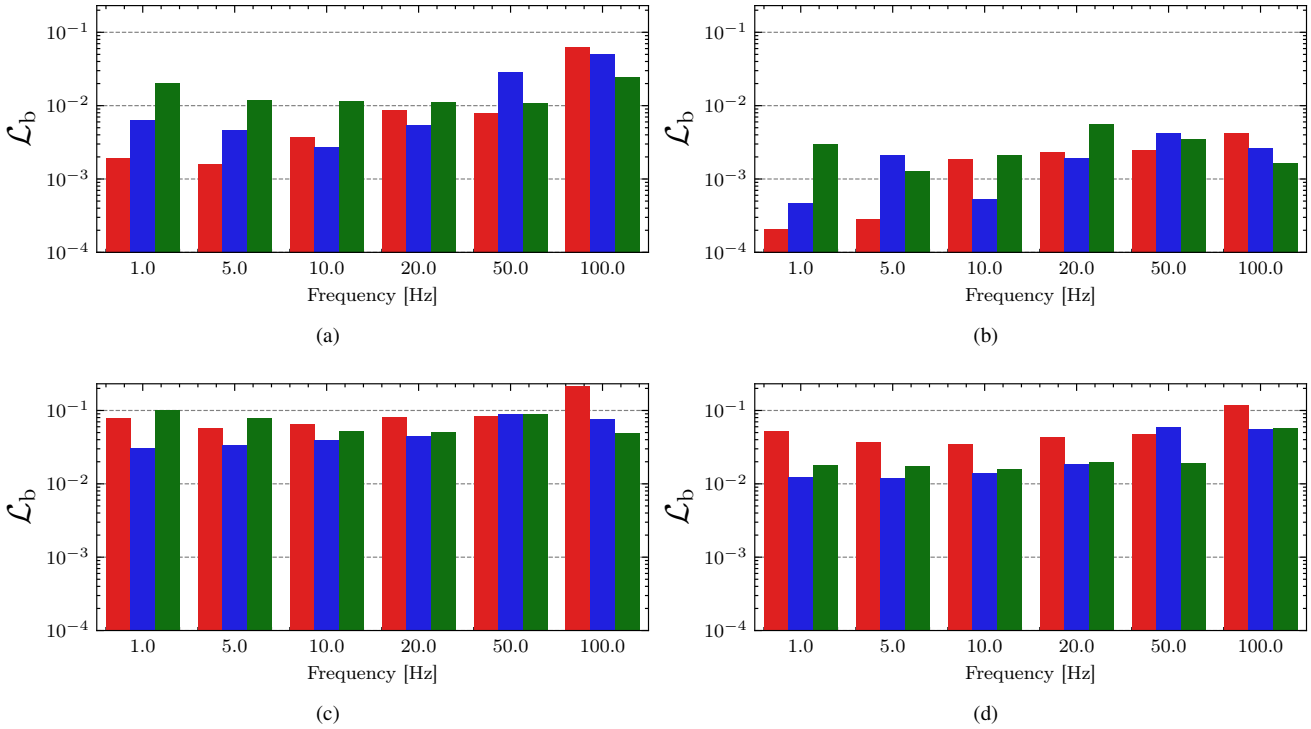

Fig. 8. Comparative analysis of the loss function $\mathcal{L}_b$: (a) the SingleNN-MemODE model, (b) the DualNN-MemODE model, (c) the MMS model, (d) the GMMS model. Results for voltage amplitudes $V_s = 0.5, 1.0, 1.5$ V are plotted in red (■), blue (■) and green (■), respectively. TODO: Replace "$\mathcal{L}_{base}$" by "$\mathcal{L}_b$"—DONE

TABLE II
COMPARATIVE ANALYSIS OF THE LOSS FUNCTION $\mathcal{L}_b$ FOR THE WHOLE DATASET AND VALIDATION DATASET. TODO: RESULTS FOR THE WHOLE DATASET SHOULD BE PROVIDED IN THE FIRST ROW.—DONE

	SingleNN-MemODE	DualNN-MemODE	GMMS Model	MMS Model
Validation dataset				
$\mathcal{L}_b$	1.49×10^{-2}	8.28×10^{-3}	4.19×10^{-2}	5.30×10^{-2}
Whole dataset				
$\mathcal{L}_b$	1.52×10^{-2}	2.23×10^{-3}	3.58×10^{-2}	7.24×10^{-2}

material-dependent switching characteristics inherent in experimental memristor devices.

B. Hyperparameter Sensitivity Analysis

Hyperparameter optimization is 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 is 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 is 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 is performed. An XGBoost gradient boosting model [20] is 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 is 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.

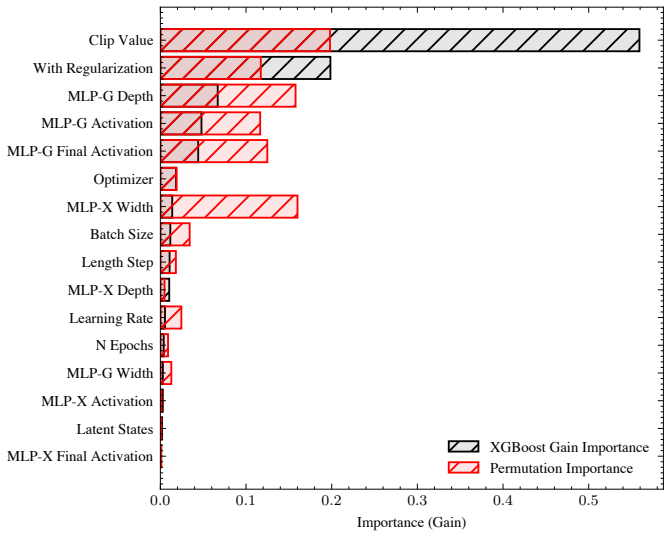

Fig. 9. Hyperparameter importance for the DualNN-MemODE optimization task. Scores are computed using permutation importance applied to an XG Boost surrogate model trained on 200 hyperparameter configurations. Larger values indicate greater relative influence of the corresponding hyperparameter on model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

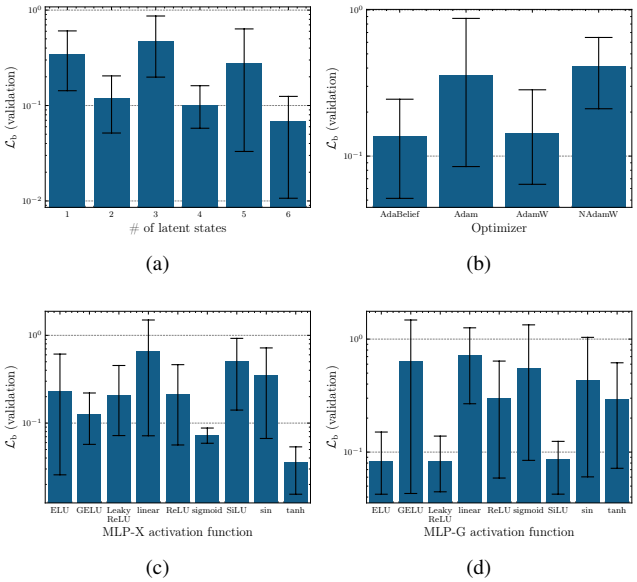

Fig. 10. Validation loss obtained during hyperparameter optimization for the categorical hyperparameters: (a) number of latent states, (b) optimizer type for the first neural network, (c) activation function for the first neural network, used to approximate $\frac{d\mathbf{x}}{dt}$, and (d) activation function for the second neural network, used to approximate $\mathcal{G}(x)$. Mean values are computed by averaging over all other hyperparameters, with error bars indicating the 95% confidence interval.

As a complementary approach, permutation importance [21] is applied to the surrogate model. In this procedure, the values of each hyperparameter are permuted at random, and the resulting degradation in predictive accuracy is 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* (i.e., 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 importance are 0.07 and 0.56, respectively, whereas the permutation-based analysis yields 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.

In Figure 10, the mean values of the validation loss obtained during the hyperparameter optimization procedure for the categorical hyperparameters are presented. For each categorical option, the reported mean is computed by marginalizing over all remaining hyperparameters, while the corresponding 95% confidence intervals are indicated by the error bars. The figure summarizes the influence of (a) the number of latent states 10a, (b) the optimizer type 10b, as well as the activation functions employed in the two neural networks constituting the DualNN-MemODE architecture 10c–10d. This analysis highlights the relative sensitivity of the model performance to discrete architectural and training-strategy choices.

IV. Conclusions

TODO: To do.—Draft created

This paper presents two novel neural ODE-based architectures for modeling the dynamics of memristor devices. The proposed models leverage the expressive power of neural networks to capture complex, nonlinear behaviors inherent in memristive systems, while maintaining the interpretability and physical consistency afforded by the ODE framework. Extensive evaluation against experimental data demonstrates that the neural ODE models significantly outperform traditional phenomenological models, such as the MMS and GMMS models, in terms of accuracy and fidelity. The results highlight the capability of the neural ODE approach to learn intricate device dynamics directly from data, thereby providing a powerful tool for simulating and understanding memristor behavior. Hyperparameter sensitivity analysis further elucidates the critical factors influencing model performance, guiding future optimization efforts. The findings of this study underscore the potential of neural ODEs as a versatile and effective framework for modeling complex physical systems, with implications for the design and analysis of memristor-based circuits and applications. The methods developed herein pave the way for more accurate and efficient simulation tools, facilitating advancements in memristor technology and its integration into emerging computing paradigms, and open avenues for future research in data-driven modeling of nonlinear dynamical systems.

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] 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

[7] 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

[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