Neural ODE based modeling of memristive devices

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 of real memristors behaviour 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 Self-Directed Channel (SDC) memristors are used to train and validate the model. Results show that the Neural ODE approach more 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 a broad class 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. The generalized *Mean Metastable Switch* (MMS) model was proposed in [2] for a more accurate modeling of Self-Directed Channel (SDC) memristors. In this approach based on semi-empirical formulation, 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 and voltage of the device 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 of a dynamical system is 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 procedure. 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 memristor devices 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 dynamics 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.

The remaining part of this paper is organized as follows. In Sec. II, the neural network architecture is presented and two neural network-based models (SingleNN-MemODE and DualNN-MemODE) are proposed. The objective function defining the optimization problem to be solved is defined and existing memristors models used for comparison are briefly described. In Sec. III, proposed models are trained using experimental data to obtain accurate models of real SDC

Manuscript received Month ??, 2026. 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 Kraków, al. Mickiewicza 30, 30–059, Kraków, Poland, (e-mail: kbednarz@agh.edu.pl).

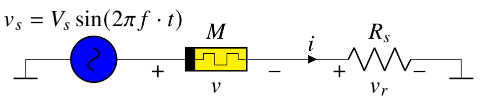

Fig. 1. Schematic diagram of the measurement setup used for the SDC memristor. The device is connected in series with a resistor, and the input voltage is applied via an arbitrary waveform generator. Voltage measurements are acquired using a data acquisition (DAQ) system.

memristors. The results are compared with the results obtained using existing methods. The influence of hyperparameters of the proposed methods on their performance is discussed. The last section concludes the study.

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. Voltage signals are measured using a data acquisition (DAQ) system. Measurements are carried out for the following combinations of supply voltage amplitudes and frequencies: $V_s \in \{0.5,\ 1.0,\ 1.5\}$ [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

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

The first term $\mathcal{L}_{\text{b}}$ denotes the primary error computed as the mean squared error (MSE) between predicted and reference values. $\mathcal{L}_{\text{v}}$ enforces accurate modeling of derivatives of state variables, $\mathcal{L}_{\text{c}}$ 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

Algorithm 1 Adaptive Loss Balancing with Clamping

```
Require: Loss terms $\mathcal{L}_b, \mathcal{L}_v, \mathcal{L}_c$, $\varepsilon > 0$, bounds $[\lambda_{\min}, \lambda_{\max}]$
Ensure: Adaptive weights $\lambda_{\text{base}}, \lambda_{\text{vel}}, \lambda_{\text{curv}}$
1: Compute the geometric mean:
     $\overline{\mathcal{L}} \leftarrow \exp\left(\frac{1}{3}\left(\ln(\mathcal{L}_b+\varepsilon) + \ln(\mathcal{L}_v+\varepsilon) + \ln(\mathcal{L}_c+\varepsilon)\right)\right)$
2: $\lambda_{\text{base}} \leftarrow \text{clamp}(\frac{\overline{\mathcal{L}}}{\mathcal{L}_b+\varepsilon}, \lambda_{\min}, \lambda_{\max})$
3: $\lambda_{\text{vel}} \leftarrow \text{clamp}(\frac{\overline{\mathcal{L}}}{\mathcal{L}_v+\varepsilon}, \lambda_{\min}, \lambda_{\max})$
4: $\lambda_{\text{curv}} \leftarrow \text{clamp}(\frac{\overline{\mathcal{L}}}{\mathcal{L}_c+\varepsilon}, \lambda_{\min}, \lambda_{\max})$
5: return $(\lambda_{\text{base}}, \lambda_{\text{vel}}, \lambda_{\text{curv}})$
```

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 $z_k = (z_{k,n})_{n=1}^{N}$ and $\tilde{z}_k = (\tilde{z}_{k,n})_{N=1}^{N}$, 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, N is the number of samples and $k = 1, 2, \ldots, N_{\text{traj}}$ is the sequence index. The sequences are normalized utilizing the standard deviation to ensure statistical consistency within each dataset. The trajectory-specific standard deviation is calculated using the unbiased estimator

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

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

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 sequences $(z_{k,n})_{n=1}^{N}$ and $(\tilde{z}_{k,n})_{n=1}^{N}$ is defined as

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

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 $\mathcal{L}_{\text{v}}$ enables evaluation of temporal derivative agreement between model predictions

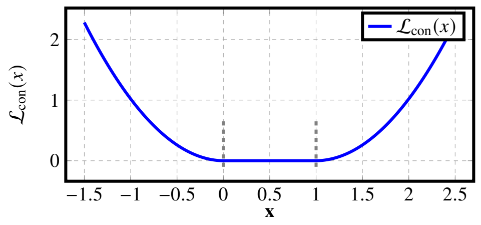

Fig. 2. The constraint function $\mathcal{L}_{\text{con}}(x)$ for $a = 0$, $b = 1$, $\gamma = 0.0025$.

and experimental data, particularly critical for capturing rapid memristor state transitions

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

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

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

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 for the admissibility interval $x \in [a, b]$ is expressed as:

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

where $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid function, a and b denote the lower and upper bounds of the admissible parameter range, γ 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 $\sigma((a-x)/\gamma)(a-x)^2$ penalizes values of x below the lower bound a, while the term $\sigma((x-b)/\gamma)(x-b)^2$ penalizes values of x above the upper bound b. Squaring ensures that the penalty is non-negative and grows quadratically with the violation magnitude.

The operational characteristics of the constraint function $\mathcal{L}_{\text{con}}(x)$ are illustrated in Fig. 2 for the parameter configuration: $a = 0$, $b = 1$, $\gamma = 0.0025$ 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 memductance $\mathcal{G}$, while the function f defining the dynamics of internal variables is modeled using a neural network (see Fig. 3). In the second one, referred to as **DualNN-MemODE**, both $\mathcal{G}$ and f are modeled using neural networks (see Fig. 4).

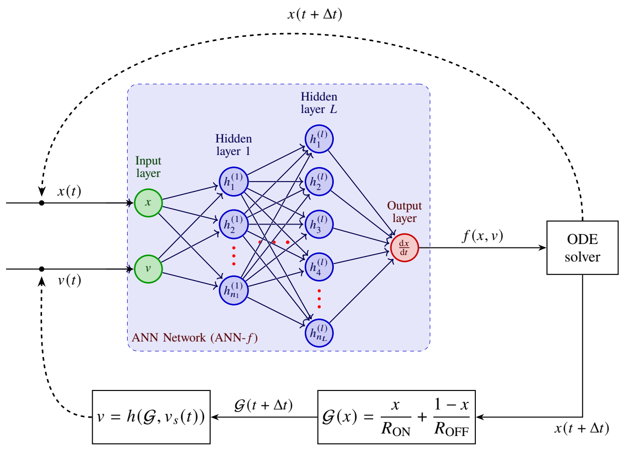

Fig. 3. The SingleNN-MemODE approach. The artificial neural network (ANN) evaluates the function $f(x, v)$, which is then integrated by the ODE solver to predict the state at the next time step $x(t + \Delta t)$. The upper dashed feedback loop indicates the recurrent nature of the process, where the output state is fed back as input for subsequent predictions. The memductance $\mathcal{G}(x)$ is defined using a deterministic function based on the internal variable x. The function $h(\mathcal{G}, v_s(t))$ computes the memristor voltage v given the memductance $\mathcal{G}$ and supply voltage $v_s(t)$.

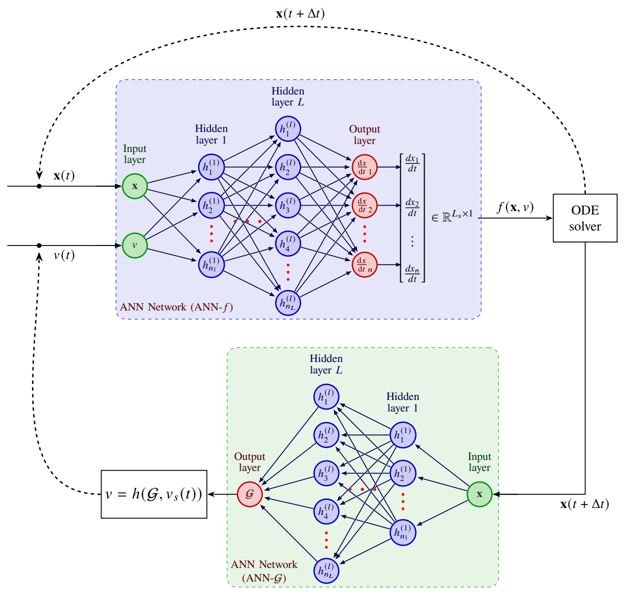

Fig. 4. The DualNN-MemODE approach. The the artificial neural network (ANN) evaluates the function $f(\mathbf{x}, v)$, which is then integrated by the ODE solver to predict the state at the next time step $\mathbf{x}(t+\Delta t)$. The second ANN models the memductance $\mathcal{G}(\mathbf{x})$ based on $\mathbf{x}(t+\Delta t)$. The function $h(\mathcal{G}, v_s(t))$ computes the memristor voltage v given the memductance $\mathcal{G}$ and supply voltage $v_s(t)$.

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 is equal to the number of internal variables and 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:

$$y_k = \sigma\left(\sum_{i=1}^{M_k} w_{k,i} x_i + b_k\right), \tag{9}$$

where k is the neuron's index, M_k is the number of inputs for the kth neuron, $w_{k,i}$ are the weights that determine the strengths of input connections x_i of the kth neuron, b_k 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.

In the SingleNN-MemODE approach, the memductance 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^+ ions migration and the formation of conductive filaments. Consequently, the device memductance exhibits a continuous transition between different resistance states. This transition can be described as a weighted combination of the limiting resistance values, modulated by the internal variable x

$$\mathcal{G}(x) = \frac{x}{R_{\mathrm{ON}}} + \frac{1-x}{R_{\mathrm{OFF}}}, \tag{10}$$

where R_{ON} and R_{OFF} denote the resistances in the low-resistance and high-resistance states, respectively. During the optimization process, the network parameters (weights $w_{k,i}$ and biases b_k) are tuned jointly with the values of R_{ON} and R_{OFF}. For this approach the following constraints are applied:

- $R_{\mathrm{ON}} \in (500\,\Omega, R_{\mathrm{OFF}})$,
- $R_{\mathrm{OFF}} \in (R_{\mathrm{ON}}, 200\,\mathrm{k}\Omega)$,
- $x(t) \in (0,1)$.

The lower bound for R_{ON} and the upper bound for R_{OFF} are based on empirical measurements. Remaining constraints are consequences of physically meaningful values of parameters and variables.

In the DualNN-MemODEapproach, the memductance is modeled using the second neural network. This neural network takes the current state $\mathbf{x}$ as the input and produces the corresponding memductance $\mathcal{G}(\mathbf{x})$ as the output. The memristor state $\mathbf{x}$ is not constrained to be one-dimensional, enabling a more flexible representation of the device's internal dynamics. The state variables are not restricted to any specific range, and hence no explicit constraints are applied to them. For this approach the only constraint is

- $\mathcal{G}(\mathbf{x}) \in (5\,\mu\mathrm{S}, 2\,\mathrm{mS})$.

These bounds are based on measurements.

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 the 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.

C. Neural Network Architecture

The neural network training is implemented using the `JAX`, library for array-oriented numerical computation [11], and `Equinox` [12] 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 `Diffrax` 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], [13], [14], [15]. 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 `tsit5` (Tsitouras 5/4 Runge-Kutta) [16], 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 Fig. 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.

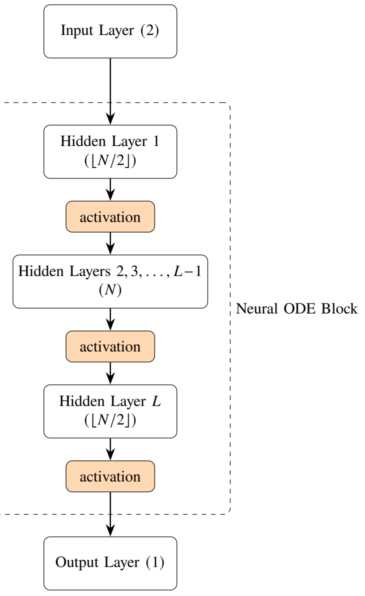

Fig. 5. Architecture of the feedforward neural network used within the Neural ODE framework for memristor dynamics modeling. The network consists of an input layer receiving two inputs, multiple hidden layers with decreasing-increasing-decreasing neuron counts ($\lfloor N/2 \rfloor, N, \ldots, N, \lfloor N/2 \rfloor$), 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 [17]. The hyperparameter search space comprises the learning rate, weight decay, batch size, and architecture parameters like the width and the depth of the neural network. The optimal configuration is selected based on minimizing the objective function on the validation set, ensuring that the model generalizes well to unseen data.

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 [18] and the Generalized Mean Metastable Switch Memristor (GMMS) model [2], [19].

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 of the device by describing transitions between the high-resistance state (HRS) and the low-resistance state (LRS) based on the applied voltage. This model describes memristor's dynamics as

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

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

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

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 memductance is defined in (10).

In the GMMS model memristors are represented as a parallel connection of memory-dependent element and a Schottky diode [2], [19]. 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

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

where $\eta \in [0,1]$ 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 across 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

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

where v_{s} denotes the supply voltage, $v_{\mathrm{r}} = R_{\mathrm{s}} i$ is the voltage across the resistor, and v 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 [20] 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 [21], 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 hyperparameter tuning, a range of neural network architectures is systematically evaluated in order to identify the most suitable configuration for modeling memristor dynamics. The evaluation considers the number of hidden layers, the number of neurons per layer, and optimization settings (e.g. the batch size and the learning rate), with the aim of achieving a balance between model accuracy and generalization capabilities, as described in Sec. 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 a low value may result in slow convergence or being trapped in a local minimum.

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 $f \in \{1\,\mathrm{Hz}, 5\,\mathrm{Hz}, 10\,\mathrm{Hz}, 20\,\mathrm{Hz}, 50\,\mathrm{Hz}, 100\,\mathrm{Hz}\}$. The number of samples for each trajectory is 10^5 (100 periods of the supply voltage per trajectory, 1000 samples per period). This dataset is divided into the training dataset consisting of 14 trajectories and the validation datasets comprising 4 trajectories. 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: $(1.5\,\text{V}, 5\,\text{Hz})$, $(1\,\text{V}, 1\,\text{Hz})$, $(1\,\text{V}, 100\,\text{Hz})$, and $(0.5\,\text{V}, 5\,\text{Hz})$.

During the hyperparameter optimization stage, the optimal configuration of the SingleNN-MemODE architecture found comprises four hidden layers with 64, 128, 128, and 64 neurons in consecutive layers. The model is trained using the `AdamW` optimizer with a learning rate of approximately $1.89{\times}10^{-2}$ and a batch size of 2. The swish activation function $\varphi(x) = x{\cdot}\sigma(x)$ is employed for the hidden layers, while the final layer utilizes the linear activation function $\varphi(x) = x$.

For the DualNN-MemODE architecture, the optimal neural network configuration responsible for computing $f(\mathbf{x}, v)$ consists of four hidden layers with 120, 240, 240, and 120 neurons in consecutive layers and with 6 latent states, corresponding to the dimensionality of the internal state vector $\mathbf{x}$. The auxiliary neural network responsible for computing $\mathcal{G}$ includes one hidden layer containing 80 neurons. Both networks are 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 employs the GELU activation function $\varphi(x) = x \cdot \Phi(x) \approx 0.5x(1 + \tanh[\sqrt{2/\pi}(x + 0.044715x^3)])$ in the hidden layers and the linear activation in the output layer, whereas the second network uses the ReLU activation function $\varphi(x) = \max(0, x)$ with the sigmoid output activation.

A representative learning curve corresponding to the SingleNN-MemODE architecture, is presented in Fig. 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 values of the objective functions for the training and validation datasets further confirms the generalization ability of the selected architecture.

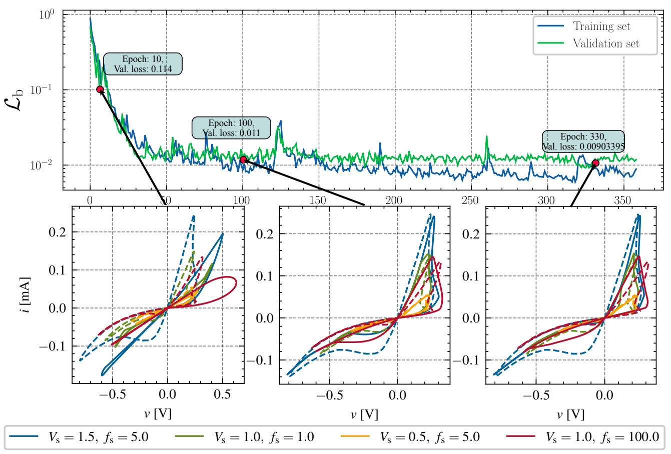

Fig. 6. Learning curve for the SingleNN-MemODE neural network architecture during training over 360 epochs. The insets illustrate v–i hysteresis loops at selected epochs, where the dashed lines correspond to the measured responses and the solid lines denote the model predictions.

Fig. 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.

The effectiveness of optimizing the full loss function $\mathcal{L}$ defined in (2) was compared against optimizing only the base loss function $\mathcal{L}_\text{b}$, defined in (5). For instance, when optimizing only the base loss function for the SingleNN-MemODE architecture, the final value obtained for the validation dataset is $\mathcal{L}_\text{b} = 2.24{\times}10^{-2}$, whereas optimizing the full loss function yields a lower value of $\mathcal{L}_\text{b} = 1.85{\times}10^{-2}$. This result indicates that incorporating additional terms $\mathcal{L}_\text{v}$ and $\mathcal{L}_\text{c}$ improves convergence and enhances the overall predictive performance of the model, as reflected by reduced validation loss values and higher accuracy on unseen data.

Representative simulation results demonstrating the neural network model's predictive capabilities are presented in Fig. 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.

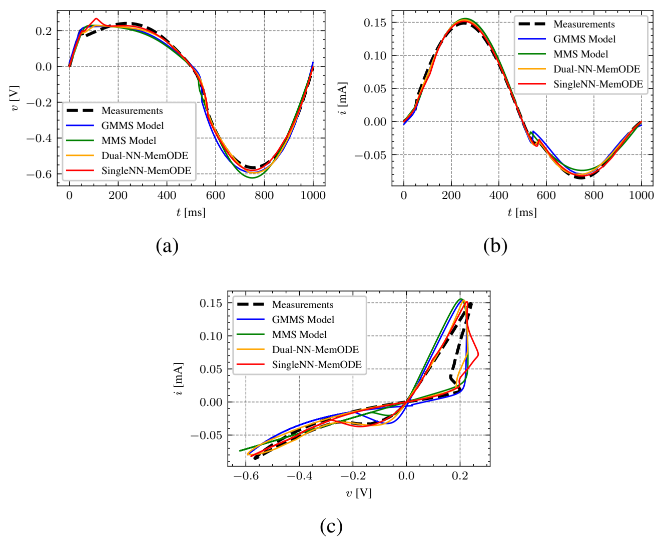

Fig. 7. Simulation test results for tungsten-doped memristor dynamics under sinusoidal voltage excitation (amplitude: 1.0 V, frequency: 1.0 Hz); (a) memristor's voltage waveform, (b) memristor's current response, (c) voltage-current hysteresis loop $(v - i)$ illustrating the characteristic pinched behavior and memory properties fundamental to memristive operation.

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

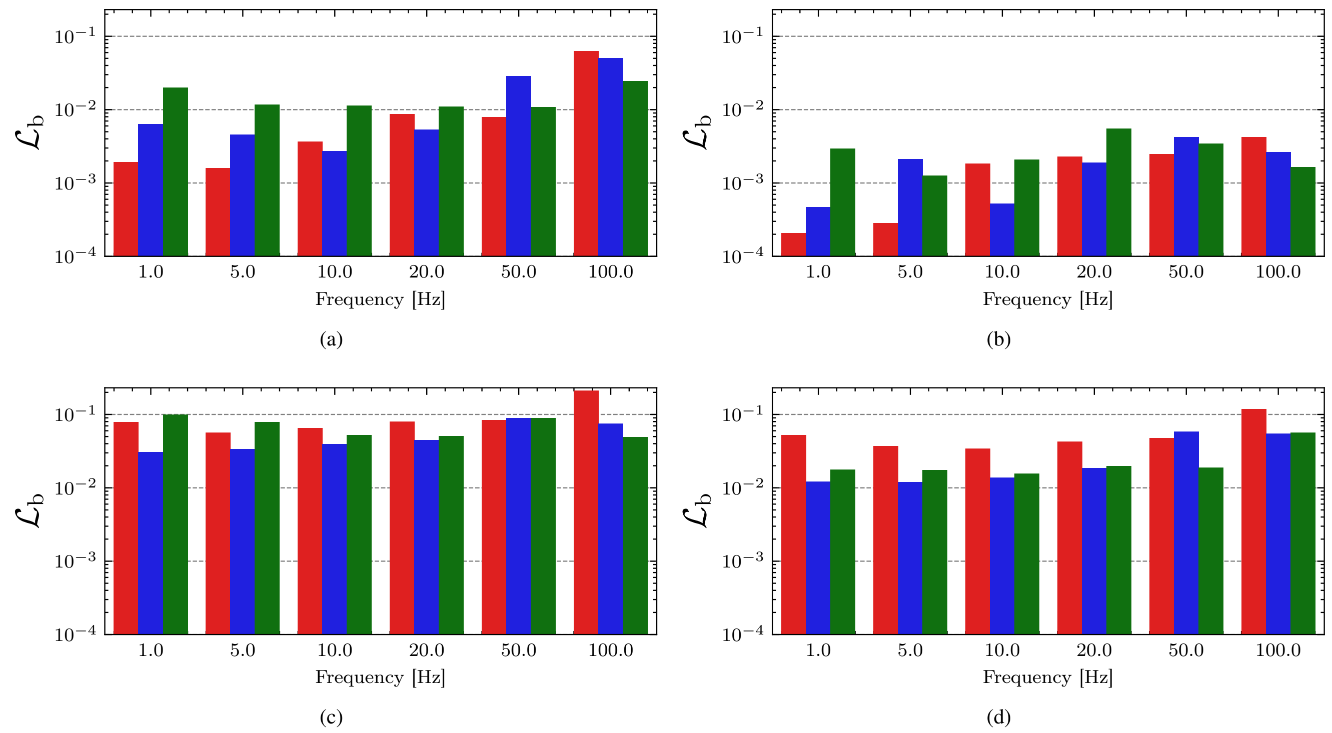

Fig. 8. Comparative analysis of the loss function $\mathcal{L}_\mathrm{b}$: (a) the SingleNN-MemODE model, (b) the DualNN-MemODE model, (c) the MMS model, (d) the GMMS model. Results for voltage amplitudes V_s = 0.5, 1.0, 1.5 V are plotted in red (🟥), blue (🟦) and green (🟩), respectively.

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 benchmarks for memristor circuit simulation [19], [22]. 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 $\mathcal{L}_\mathrm{b}$ 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 $\mathcal{L}_\mathrm{b}$ calculated for the validation, training and the whole datasets for the proposed and existing models. Results obtained for the validation dataset provide a quantitative evaluation of generalization performance of tested models across unseen memristor dynamics. The best result of 5.77×10^{-3} is obtained for the DualNN-MemODE model. The significant reduction of the average values of $\mathcal{L}_\mathrm{b}$ 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 real memristor devices.

Let us now compare computation times needed to find different models and evaluation times of models. Computations are carried out using a 3.5 GHz Intel i9-9900X processor. The median training duration for the SingleNN-MemODE and DualNN-MemODE architectures is approximately 11.3 minutes and 13.8 minutes, respectively. The median computational time necessary for parameter optimization of the MMS model is approximately 20.4 minutes, whereas the GMMS model

TABLE I
COMPARATIVE ANALYSIS OF THE LOSS FUNCTION $\mathcal{L}_b$. VALUES ARE REPORTED FOR DIFFERENT SUPPLY VOLTAGE AMPLITUDES V_s AND FREQUENCIES f_s. ROW-WISE MINIMUM AND MAXIMUM LOSS VALUES ARE COLOR-CODED IN GREEN AND RED, RESPECTIVELY. RECORDS BELONGING TO THE VALIDATION DATASET ARE MARKED BY BOUNDING BOXES.

Loss function values for V_s = 0.5 V

f_s [Hz]	SingleNN-MemODE	DualNN-MemODE	GMMS Model	MMS Model
1	9.31×10^{-3}	1.22×10^{-3}	3.94×10^{-2}	2.43×10^{-2}
5	3.40×10^{-3}	6.43×10^{-3}	2.97×10^{-2}	1.46×10^{-2}
10	3.05×10^{-3}	2.13×10^{-3}	2.85×10^{-2}	2.60×10^{-2}
20	3.79×10^{-3}	1.02×10^{-3}	3.25×10^{-2}	1.14×10^{-1}
50	2.03×10^{-2}	5.60×10^{-3}	3.68×10^{-2}	3.45×10^{-2}
100	2.82×10^{-2}	1.07×10^{-2}	9.91×10^{-2}	1.11×10^{-1}

Loss function values for V_s = 1 V

f_s [Hz]	SingleNN-MemODE	DualNN-MemODE	GMMS Model	MMS Model
1	5.28×10^{-3}	4.47×10^{-3}	1.30×10^{-2}	1.63×10^{-2}
5	8.25×10^{-3}	2.67×10^{-3}	1.46×10^{-2}	1.71×10^{-2}
10	4.72×10^{-3}	8.86×10^{-4}	1.49×10^{-2}	1.58×10^{-2}
20	7.77×10^{-3}	5.53×10^{-3}	1.88×10^{-2}	1.90×10^{-2}
50	3.29×10^{-2}	7.37×10^{-3}	6.29×10^{-2}	6.22×10^{-2}
100	4.48×10^{-2}	8.58×10^{-3}	4.53×10^{-2}	3.25×10^{-2}

Loss function values for V_s = 1.5 V

f_s [Hz]	SingleNN-MemODE	DualNN-MemODE	GMMS Model	MMS Model
1	1.74×10^{-2}	2.48×10^{-3}	1.75×10^{-2}	4.74×10^{-2}
5	1.57×10^{-2}	3.60×10^{-3}	2.15×10^{-2}	3.71×10^{-2}
10	1.55×10^{-2}	3.70×10^{-3}	2.08×10^{-2}	2.31×10^{-2}
20	1.32×10^{-2}	7.93×10^{-3}	2.07×10^{-2}	2.41×10^{-2}
50	1.37×10^{-2}	8.41×10^{-3}	1.69×10^{-2}	4.70×10^{-2}
100	2.64×10^{-2}	1.20×10^{-2}	5.54×10^{-2}	2.41×10^{-2}

TABLE II
COMPARATIVE ANALYSIS OF THE LOSS FUNCTION $\mathcal{L}_b$ FOR THE VALIDATION, TRAINING AND THE WHOLE DATASETS.

	SingleNN-MemODE	DualNN-MemODE	GMMS Model	MMS Model
Validation dataset				
$\mathcal{L}_b$	1.73×10^{-2}	5.77×10^{-3}	2.74×10^{-2}	2.51×10^{-2}
Training dataset				
$\mathcal{L}_b$	1.46×10^{-2}	5.11×10^{-3}	3.42×10^{-2}	4.22×10^{-2}
Whole dataset				
$\mathcal{L}_b$	1.52×10^{-2}	5.26×10^{-3}	3.27×10^{-2}	3.84×10^{-2}

requires approximately 8.4 minutes using the same hardware. The neural network architectures are 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) requires a two-stage optimization procedure. Initially, a computationally intensive global search algorithm is executed over the whole parameter space to obtain an approximate yet robust initial parameter estimate. Subsequently, a local refinement procedure employing gradient-based optimization is 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 computation time for trajectory generation comprising 18 trajectories using the MMS model is approximately 0.047 seconds, whereas the GMMS model requires approximately 0.051 seconds. In contrast, the DualNN-MemODE architecture needs approximately 3.97 seconds, while the SingleNN-MemODE architecture requires approximately 5.56 seconds to generate an equivalent trajectory ensemble. The neural network models are integrated numerically using a relative error tolerance of 10^{-3} and an absolute error tolerance of 10^{-6}.

B. Hyperparameter Sensitivity Analysis

Hyperparameter optimization is conducted to systematically explore the hyperparameter space and identify configurations that maximize the predictive performance of neural network architectures. The optimization employs the Tree-structured Parzen Estimator (TPE) algorithm, implemented in the `Optuna` framework [23]. The TPE algorithm creates a probabilistic model of the objective function 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 encompasses 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 includes widely used nonlinearities such as ReLU, GELU, SiLU, tanh, ELU, leaky ReLU, sigmoid, identity and sine functions. For the output layer, the choice is restricted to linear, tanh, sigmoid, and 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 the model performance, a surrogate-based importance analysis is performed. An XGBoost gradient boosting model [24] is trained on the explored configurations to predict validation loss. The surrogate model achieved an R^2 value of 0.99, indicating high fidelity in capturing the dependence of model performance on hyperparameter selection, where $R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$ represents the coefficient of determination measuring the proportion of explained variance, y_i are the observed values, $\hat{y}_i$ are the model predictions, and $\bar{y}$ is the mean of observed values. Importance is first estimated using the *gain* metric, which reflects the cumulative improvement in predictive accuracy attributed to each hyperparameter across the ensemble. Normalization ensured that scores sum up to unity, allowing for direct interpretation as relative contributions to performance variability.

As a complementary approach, permutation importance [25] 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 provides an alternative measure of importance, which, unlike gain-based metrics, also accounts for nonlinear dependencies and interactions among hyperparameters.

Results of the sensitivity analysis are presented in Fig. 9. According to the XGBoost Gain approach the dominant hyperparameters are the number of epochs, the gradient clipping value (the maximum allowable gradient norm), the ANN-f depth (the depth of the neural network used to approximate $f(\mathbf{x}, v)$), and the number of latent states, with the relative importance scores of 0.47, 0.27, 0.12, and 0.06, respectively.

Collectively, the four most significant parameters account for more than 90% of the total observed performance variance.

The permutation importance approach identifies the following four most dominant hyperparameters influencing model performance: the learning rate, the gradient clipping value, the number of epochs, and the ANN-$\mathcal{G}$ depth (the depth of the neural network used to approximate $\mathcal{G}(\mathbf{x})$). Their relative importance scores are 0.27, 0.24, 0.19, and 0.19, respectively.

Summarizing, the most important hyperparameters are identified as the number of epochs, the gradient clipping value, the learning rate, depths of neural networks and the number of latent states. Widths of neural networks, the batch size and the step length have moderate influence. The following hyperparameters have negligible effects of the model performance: the choice of activation and output functions, and the optimizer type. For example, the choice of activation function exhibits a negligible effect (importance score below 0.01), indicating that the Neural ODE architecture demonstrates a high degree of robustness with respect to this factor across the examined range.

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

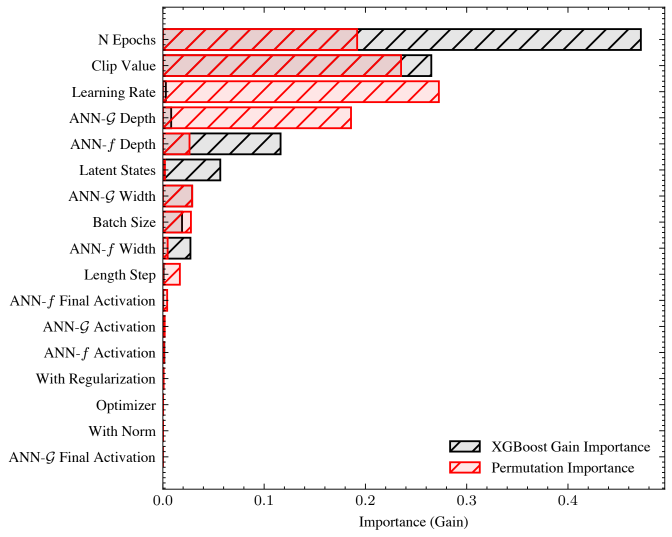

Fig. 9. Hyperparameter importance for the DualNN-MemODE optimization task. Scores are computed using the XG Boost surrogate model and the permutation importance approach. Larger values indicate greater relative influence of the corresponding hyperparameter on the model performance. ANN-$\mathcal{G}$ and ANN-f refer to neural networks used to approximate $\mathcal{G}(\mathbf{x})$ and $f(\mathbf{x}, v)$, respectively.

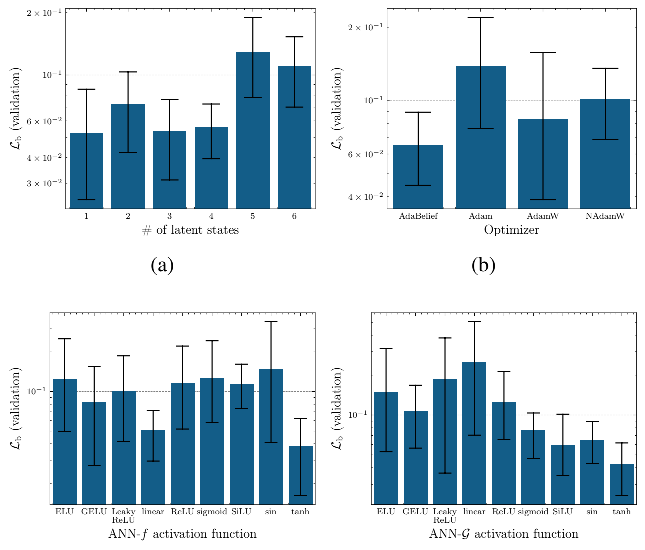

Fig. 10. Validation loss obtained during hyperparameter optimization for selected hyperparameters: (a) number of latent states, (b) optimizer type for the first neural network, (c) activation function for the neural network used to approximate $f(\mathbf{x}, v)$, and (d) activation function for the neural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

IV. Conclusions

This work presented two novel neural ODE-based architectures for modeling the dynamics of memristor devices. The proposed models leverage the 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. The proposed models were applied for modeling of real Self-Directed Channel (SDC) memristors. Extensive evaluation against experimental data demonstrates that the neural ODE approach significantly outperforms 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. Obtained results show that the computation time needed to optimize the proposed models is smaller than that for existing models. On the other hand the evaluation time needed to compute trajectories using the proposed models is significantly larger than for the MMS and GMMS models.

The proposed models may also be used for modeling of memristors produced in other technologies. The findings of this study underscore the potential of neural ODEs as a versatile and effective framework for modeling complex physical systems, with applications in the design and analysis of memristor-based circuits. Developed methods pave the way to 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 RAMapplications," *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] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/jax-ml/jax

[12] P. Kidger and C. Garcia, "Equinox: neural networks in JAX via callable PyTrees and filtered transformations," *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

[13] P. Kidger, "On Neural Differential Equations," Ph.D. dissertation, University of Oxford, 2021.

[14] P. Stumm and A. Walther, "New algorithms for optimal online checkpointing," *SIAM Journal on Scientific Computing*, vol. 32, no. 2, pp. 836–854, 2010.

[15] Q. Wang, P. Moin, and G. Iaccarino, "Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation," *SIAM Journal on Scientific Computing*, vol. 31, no. 4, pp. 2549–2567, 2009.

[16] C. Tsitouras, "Runge–kutta pairs of order 5(4) satisfying only the first column simplifying assumption," *Computers ; Mathematics with Applications*, vol. 62, no. 2, pp. 770–775, Jul. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.camwa.2011.06.002

[17] 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

[18] 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

[19] 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

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

[21] 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

[22] 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

[23] 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.

[24] 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

[25] 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