How to Use the IEEEtran LATEX Templates

IEEE Publication Technology Department

***Abstract*—This document describes the most common article elements and how to use the IEEEtran class with LATEX to produce files that are suitable for submission to the Institute of Electrical and Electronics Engineers (IEEE). IEEEtran can produce conference, journal and technical note (correspondence) papers with a suitable choice of class options.**

***Index Terms*—Class, IEEEtran, LATEX, paper, style, template, typesetting.**

I. Introduction

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

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

Manuscript created October, 2020; This work was developed by the IEEE Publication Technology Department.

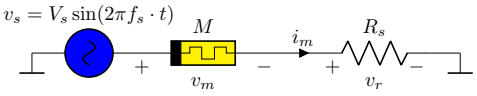

Fig. 1. Schematic diagram of the measurement setup used for the SDC memristor. The device is connected in series with a resistor, and the input voltage is applied via an arbitrary waveform generator. Voltage measurements are acquired using a data acquisition (DAQ) system.

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. Materials and Methods

This study utilizes experimental data obtained from physical 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, consisted of an SDC memristor connected in series with a resistor. The sinusoidal input voltage was supplied by an arbitrary waveform generator, while the voltage signals were measured using a data acquisition (DAQ) system.

Measurements were carried out for various combinations of supply voltage amplitudes and frequencies. Specifically, the amplitude of the applied voltage was varied as $V_s \in \{0.5,\ 1.0,\ 1.5\}$ [V], and the frequency was selected from the set $f \in \{1,\ 5,\ 10,\ 20,\ 50,\ 100\}$ [Hz].

A. Modeling with Neural Networks

The analyzed model is grounded in the general theoretical framework of memristive devices, originally introduced in [8], and is expressed through equations (1).

$$\begin{aligned} v(t) &= \mathcal{M}\big(\mathbf{x}(t), v(t)\big)\, i(t), \\ \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} &= f\big(\mathbf{x}(t), v(t)\big) \end{aligned} \tag{1}$$

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

In this study, two neural network–based modeling approaches are investigated. The first, denoted as **Det-MemODE**, employs a deterministic formulation of the memristor conductance. The second, referred to as **Dual-NN-MemODE**, adopts a data-driven strategy in which the conductance is directly modeled using neural networks. Although both approaches utilize neural networks to approximate the dynamic function $f(\mathbf{x}(t), v(t))$, they differ fundamentally in the representation of the memristor conductance.

In the first approach (Det-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 state variable. Formally, the memristor conductance is expressed as $G_m(x) = \mathcal{M}(x)^{-1}$, where $\mathcal{M}(x)$ denotes the state-dependent resistance.

$$G_m(x) = \frac{x}{R_{\mathrm{ON}}} + \frac{1-x}{R_{\mathrm{OFF}}}, \tag{2}$$

where R_{ON} and R_{OFF} denote the resistances in the low-conductance and high-conductance states, respectively. For all of these approaches also the function $f(\mathbf{x}(t), v(t))$ was implemented using an artificial neural network, whose conceptual architecture is illustrated in Fig. 2 for the Det-MemODE and in Fig. 3 for the Dual-NN-MemODE.

The neural network consists of interconnected nodes (neurons) that process information through weighted connections. Each neuron in the network computes its output according to:

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3}$$

where w_i are the weights that determine the strength and importance 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.

The weights represent the learnable parameters that encode the relationship between inputs and outputs, while biases allow neurons to shift their activation threshold, enabling the network to better fit the training data even when all inputs are zero. During the optimization process, the network parameters (weights and biases) are tuned jointly with the values of R_{ON} and R_{OFF}.

The learning process involves iteratively adjusting the network parameters to minimize a 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

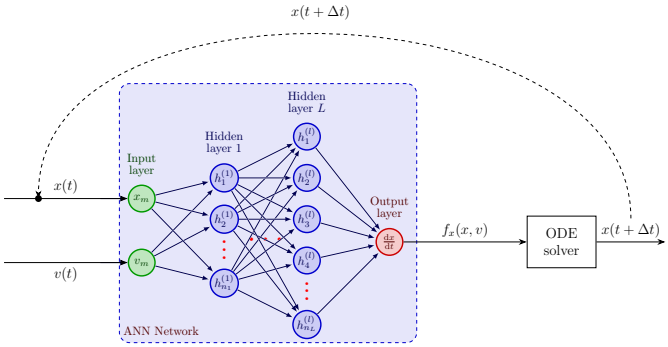

Fig. 2. Conceptual diagram illustrating the use of a neural network to simulate the dynamics of the memristor, for the Det-MemODE, with schematic representation of the Neural ODE framework, where an artificial neural network (ANN) computes the derivative function $f_x(x, v)$, which is then integrated by an 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.

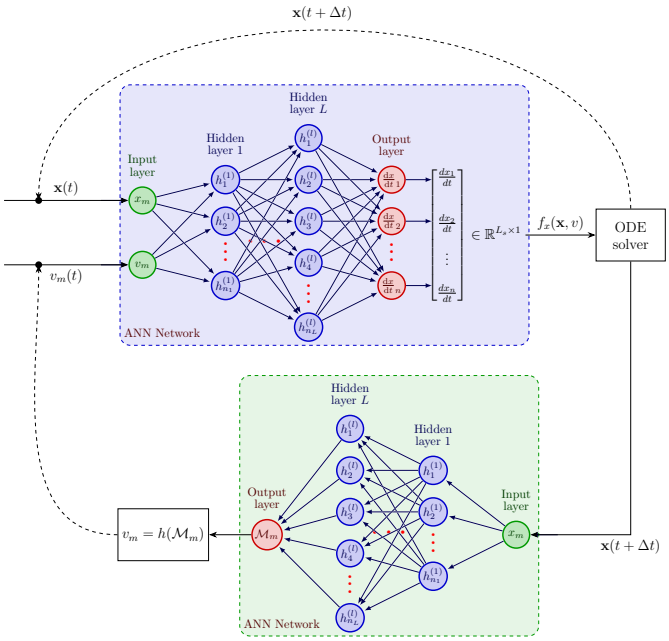

Fig. 3. Conceptual diagram illustrating the use of a neural network to simulate the dynamics of the memristor, for the Dual-NN-MemODE, with schematic representation of the Neural ODE framework, where an artificial neural network (ANN) computes the derivative function $f_x(x, v)$, which is then integrated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

to update the weights and biases in the direction that reduces the overall error.

B. Neural Network Architecture Design

The neural network model development and training pipeline was implemented using the `PyTorch` deep learning framework, described in [9], within the `Python` programming environment, leveraging its automatic differentiation capabilities. To enable seamless integration of ordinary differential equations (ODEs) within the neural network training paradigm, the specialized `torchdiffeq` library was employed. This library provides differentiable ODE solvers that facilitate efficient gradient computation with respect to solution trajectories through the adjoint sensitivity method, well described in [4], [5], [10]. 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 dynamic systems governed by differential equations.

Specifically, the `dopri5` (Dormand-Prince 5th order) adaptive Runge-Kutta solver was utilized for its superior balance between computational efficiency and numerical accuracy. This solver employs embedded 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 4, employs an expansion–compression (diamond-shaped) topology that incorporates a twofold reduction in dimensionality in both the initial and final hidden layers. This architectural configuration effectively addresses several challenges associated with neural ODE training and the modeling of memristive systems, namely:

- Dimensionality reduction at the network's input and output stages substantially decreases the overall number of trainable parameters,
- The reduced parameterization facilitates more stable optimization dynamics by limiting the complexity of the parameter search space,
- The controlled dimensional transitions help mitigate vanishing and exploding gradient issues commonly encountered in deep architectures processing extended temporal sequences,
- The compression phase at the output acts as an implicit regularizer, constraining the learned representations to a compact, low-dimensional subspace consistent with the underlying physical behavior of memristor dynamics.

The network weights were optimized using several optimization techniques, including the `Adam` (Adaptive Moment Estimation) optimizer. The choice of optimizer and its parameters was guided by hyperparameter tuning conducted with the `Optuna` framework, as described in [11]. The hyperparameter search space comprised the learning rate, weight decay, and batch size, while the optimal configuration was selected based on validation loss minimization. To enable adaptive control of the learning rate throughout training, the `ReduceLROnPlateau` scheduling strategy was employed.

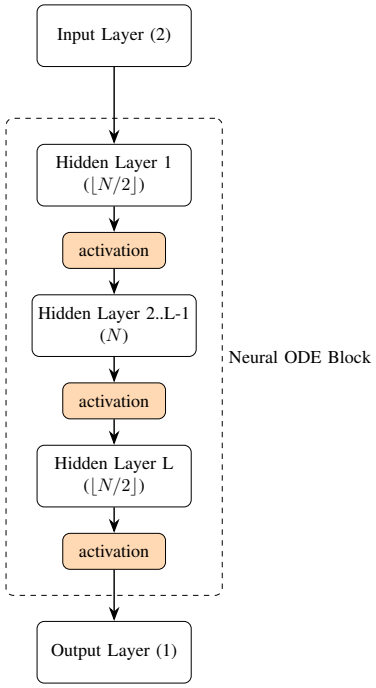

Fig. 4. Architecture of the feedforward neural network used within the Neural ODE framework for memristor dynamics modeling. The network consists of an input layer receiving two inputs, multiple hidden layers with decreasing-increasing-decreasing neuron counts ($\lfloor N/2 \rfloor, N, \lfloor N/2 \rfloor$), activation functions between layers, and a single output. The Neural ODE block encompasses the core computational layers that approximate the derivative function $f(x(t), v(t))$ in the differential equation system.

This performance-based scheduler dynamically adjusts the learning rate in response to plateaus in the monitored validation loss, thereby improving convergence stability. Specifically, the scheduler reduces the learning rate once the monitored loss stagnates, applies a predefined reduction factor, incorporates a patience interval before making adjustments, and enforces lower and upper bounds to prevent instability. This implementation strategy ensures robust and efficient training of neural differential equation models for memristor circuit simulation, integrating state-of-the-art deep learning practices with domain-specific techniques for dynamic system modeling.

C. Deterministic Memristor Model for Comparison

To evaluate the effectiveness of the proposed model, it was compared with the two deterministic models: the Mean Metastable Switch (MMS) model introduced in [12] and the Generalized Mean Metastable Switch Memristor (GMMS) model described in [2], [13].

1) Mean Metastable Switch (MMS) Model: The Mean Metastable Switch (MMS) model is a simplified representation of memristor behavior, focusing on the average switching characteristics rather than the detailed dynamics. It captures the essential features of memristive switching by modeling the device as a two-state system, where the resistance can switch between a high-resistance state (HRS) and a low-resistance state (LRS) based on the applied voltage. This model describes the memristor dynamics using Eq. (4).

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{1}{\tau}\big[f_{\mathrm{on}}(t)(1-x) - f_{\mathrm{off}}(t)x\big] \tag{4}$$

where

$$f_{\mathrm{on}}(t) = \frac{1}{1+e^{-\beta(v(t)-V_{\mathrm{ON}})}} \tag{5}$$

$$f_{\mathrm{off}}(t) = 1 - \frac{1}{1+e^{-\beta(v(t)-V_{\mathrm{OFF}})}} \tag{6}$$

where $\beta = \frac{q}{kT}$ is a temperature-related parameter, V_{ON} denotes the switching voltage to the low-resistance state (LRS), V_{OFF} denotes the switching voltage to the high-resistance state (HRS), k is the Boltzmann constant, T is the absolute temperature, and q represents the elementary charge. The instantaneous conductance of the memristor is defined as in Eq. (2).

2) Generalized Mean Metastable Switch (GMMS) Model: In [2], the authors introduced the Generalized Mean Metastable Switch (GMMS) model, in which the memristor is represented as a parallel connection of two memristors and a Schottky diode. This formulation extends the original metastable switch concept by incorporating nonlinear current–voltage characteristics, thereby enabling a more accurate representation of memristor behavior across a wide range of operating conditions.

Nevertheless, the inclusion of additional parameters and nonlinearities increases the model's complexity and poses challenges for parameter optimization. In this framework, the evolution of the state variable is governed by Eq. (4), while the current–voltage relationship is described by the following set of equations:

$$\begin{aligned} i(t) &= \phi i_m(v,t) + (1-\phi) i_d(v) \\ i_d(v) &= \alpha_f e^{\beta_f v} - \alpha_r e^{-\beta_r v} \\ i_m(v,t) &= \mathcal{G}_m(x) v \end{aligned} \tag{7}$$

where $\phi \in [0,1]$ is a parameter representing the fraction of the total current flowing through the memristor relative to the entire current 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.

In the case of the GMMS model, 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 nonlinear equation of the form Eq. (8). As a result, the system of ordinary differential equations is transformed into a differential-algebraic equation (DAE).

$$\begin{cases} v_s(t) = v_r(t) + v_m(t) \\ \dfrac{\mathrm{d}x}{\mathrm{d}t} = \dfrac{1}{\tau}\big[f_{\mathrm{on}}(t)(1-x) - f_{\mathrm{off}}(t)x\big] \end{cases} \tag{8}$$

where $v_s(t)$ denotes the supply voltage, $v_r(t) = R_s i(t)$ is the voltage across the resistor, and $v_m(t)$ is the voltage across the memristor.

3) Parameter Optimization: Since the deterministic models involve multiple parameters, a hybrid optimization framework was implemented to determine the parameter set that minimizes the discrepancy between model predictions and experimental data. In comparison to neural network–based models, the number of parameters to be optimized is significantly smaller, which facilitates the calibration process. The framework consists of two stages, combining global exploration with local refinement. In the first stage, a global optimization algorithm was 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 for the `Julia` programming language [14], was utilized to efficiently search for candidate solutions. In the second stage, local refinement was performed using the L-BFGS-B algorithm [15], 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.

D. Objective Function

The loss function based on the phase portrait 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. The total loss function is given by:

$$\begin{aligned} \mathcal{L}_{\mathrm{total}} = \lambda_{\mathrm{base}} \cdot \mathcal{L}_{\mathrm{base}} + \lambda_{\mathrm{vel}} \cdot \mathcal{L}_v \\ + \lambda_{\mathrm{curv}} \cdot \mathcal{L}_c + \sum_{i=1}^{N_c} \lambda_{\mathrm{con}_i} \cdot \mathcal{L}_{\mathrm{con}_i}(x) \end{aligned} \tag{9}$$

Here, $\mathcal{L}_{\mathrm{base}}$ denotes the primary error term, measuring the difference between predicted and reference values, in that case the mean squared error (MSE). The term $\mathcal{L}_v$ enforces accurate modeling of the velocity characteristics by considering the derivatives of trajectories in phase space. The component $\mathcal{L}_c$ promotes consistency of the trajectories in terms of their curvature, for example by analyzing changes in trajectory direction or curvature. Finally, $\mathcal{L}_{\mathrm{con}}(x)$ introduces additional physical or structural constraints that the model is required to satisfy, including conditions of stability, nonlinearities, or conformity to the underlying physical model. The weighting coefficients λ_{base}, λ_{vel}, and λ_{curv} balance the influence of each term within the overall loss function according to the priorities of the modeling task, and each of the values were calculated during each optimization step using the Geometric Loss Strategy (GLS), described in [16], [17]. with the following algorithm presented in Algorithm 1.

For circuit trajectory datasets indexed by $k = 1, \ldots, N_{\mathrm{traj}}$, a trajectory-specific standardization procedure is implemented to normalize each electrical variable $z \in \{v, i\}$ (representing voltage and current measurements, respectively). This normalization process utilizes the standard deviation computed

Algorithm 1 Adaptive Loss Balancing with Clamping (Compact)

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

independently for each individual trajectory sequence, ensuring statistical consistency within each temporal dataset. The trajectory-specific standard deviation is calculated using the unbiased sample estimator:

$$\sigma_{z,k} = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}\left(z_{k,t}^{\text{true}} - \bar{z}_k^{\text{true}}\right)^2} \tag{10}$$

where the trajectory-specific temporal mean is defined as:

$$\bar{z}_k^{\text{true}} = \frac{1}{T}\sum_{t=1}^{T} z_{k,t}^{\text{true}} \tag{11}$$

The parameter T represents the temporal resolution of the trajectory, corresponding to the total number of discrete time-series sampling points within each measurement sequence.

This trajectory-wise normalization methodology serves several critical analytical objectives in circuit analysis and machine learning applications. Primarily, it establishes scale invariance across electrical signals exhibiting disparate amplitude characteristics, thereby facilitating quantitative comparison of dynamic waveform morphologies independent of absolute magnitude variations. Furthermore, this standardization approach enhances the numerical stability and convergence properties of optimization algorithms employed in neural network training procedures.

By implementing trajectory-specific normalization rather than global standardization, the methodology preserves the inherent temporal dynamics and statistical properties unique to each circuit configuration while simultaneously ensuring that amplitude disparities do not introduce systematic bias during pattern recognition, feature extraction, or predictive modeling procedures. This approach is particularly advantageous in scenarios involving heterogeneous circuit topologies or varying operational conditions, where maintaining the relative temporal structure of electrical phenomena is paramount for accurate system identification and dynamic analysis. The neural network loss function comprises several specialized components designed to optimize memristor circuit modeling performance across multiple dynamical characteristics. Each component addresses specific aspects of the temporal and nonlinear behavior exhibited by memristive systems.

The fundamental component of the loss function is responsible for model fitting to experimental voltage-current data, incorporating trajectory-specific normalization procedures. This component minimizes the mean squared error (MSE) between predicted and ground-truth voltage and current waveforms, normalized by the standard deviation of each individual trajectory. This normalization ensures scale invariance across datasets with varying amplitude characteristics.The mean squared error for a sequence of n data points is defined in Eq (12):

$$\text{MSE}(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{12}$$

The base loss is formally expressed as it shown in Eq. (13):

$$\begin{aligned}\mathcal{L}_{\text{base}} = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\Bigg[&\text{MSE}\left(\frac{v_{\text{pred},k}}{\sigma_{v,k}}, \frac{v_{\text{true},k}}{\sigma_{v,k}}\right)\\ &+ \text{MSE}\left(\frac{i_{\text{pred},k}}{\sigma_{i,k}}, \frac{i_{\text{true},k}}{\sigma_{i,k}}\right)\Bigg]\end{aligned} \tag{13}$$

where $v_{\text{pred},k}$ and $i_{\text{pred},k}$ represent the predicted voltage and current for trajectory k, while $v_{\text{true},k}$ and $i_{\text{true},k}$ denote the corresponding experimental measurements.

The first-order dynamics comparison enables evaluation of temporal derivative agreement between model predictions and experimental data, particularly critical for capturing rapid memristor state transitions. This component ensures that the neural network learns not only absolute signal values but also their instantaneous rates of change, which is fundamental for accurate modeling of dynamic systems exhibiting switching phenomena. The first-order loss component is defined as:

$$\begin{aligned}\mathcal{L}_v = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\Bigg[&\text{MSE}\left(\frac{\frac{\mathrm{d}v}{\mathrm{d}t}_{\text{pred},k}}{\sigma_{\dot{v},k}}, \frac{\frac{\mathrm{d}v}{\mathrm{d}t}_{\text{true},k}}{\sigma_{\dot{v},k}}\right)\\ &+ \text{MSE}\left(\frac{\frac{\mathrm{d}i}{\mathrm{d}t}_{\text{pred},k}}{\sigma_{\dot{i},k}}, \frac{\frac{\mathrm{d}i}{\mathrm{d}t}_{\text{true},k}}{\sigma_{\dot{i},k}}\right)\Bigg]\end{aligned} \tag{14}$$

This formulation captures the velocity characteristics of electrical variables, ensuring that transient behaviors and switching dynamics are accurately reproduced by the model.

The second-order comparison term addresses trajectory curvature characteristics, which are essential for capturing subtle variations in system dynamics that manifest as nonlinear memristor behaviors. 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. The curvature loss is formulated as:

$$\begin{aligned}\mathcal{L}_c = \frac{1}{N_{\text{traj}}}\sum_{k=1}^{N_{\text{traj}}}\Bigg[&\text{MSE}\left(\frac{\frac{\mathrm{d}^2v}{\mathrm{d}t^2}_{\text{pred},k}}{\sigma_{\ddot{v},k}}, \frac{\frac{\mathrm{d}^2v}{\mathrm{d}t^2}_{\text{true},k}}{\sigma_{\ddot{v},k}}\right)\\ &+ \text{MSE}\left(\frac{\frac{\mathrm{d}^2i}{\mathrm{d}t^2}_{\text{pred},k}}{\sigma_{\ddot{i},k}}, \frac{\frac{\mathrm{d}^2i}{\mathrm{d}t^2}_{\text{true},k}}{\sigma_{\ddot{i},k}}\right)\Bigg]\end{aligned} \tag{15}$$

By incorporating second-order temporal derivatives, this component ensures that acceleration characteristics and higher-order dynamical features are preserved during the learning process.

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

$$\mathcal{L}_{\text{con}}(x) = \mathbb{E}\left[\left(\frac{1}{1+\exp\left(\frac{x-a}{f}\right)}\cdot(a-x) + \frac{1}{1+\exp\left(\frac{-(x-b)}{f}\right)}\cdot(x-b)\right)^2\right] \tag{16}$$

where the constituent parameters are defined as follows, $\sigma(z) = \frac{1}{1+e^{-z}}$ represents the sigmoid activation function (implicitly used), a and b denote the lower and upper bounds of the admissible parameter range, respectively, f is a smoothing (scaling) factor controlling the steepness of the transition near the bounds, the term $\frac{1}{1+\exp((x-a)/f)}(a-x)$ penalizes values of x below the lower bound a, the term $\frac{1}{1+\exp(-(x-b)/f)}(x-b)$ penalizes values of x above the upper bound b, squaring ensures the penalty is non-negative and grows quadratically with the violation magnitude, $\mathbb{E}[\cdot]$ denotes the expectation operator (ensemble average over samples or trajectories).

This constraint formulation provides smooth penalty gradients that guide parameters toward the feasible region while avoiding discontinuities that could disrupt the optimization process. The sigmoid-based transitions ensure that the constraint becomes progressively more restrictive as parameters approach the boundary limits, promoting stable convergence behavior. During neural network training, specific interval constraints were implemented to ensure physically meaningful and numerically stable parameter values that align with experimental memristor characteristics and theoretical boundaries. These constraints prevent parameter drift into unphysical regimes while maintaining the differentiability required for gradient-based optimization algorithms.

The following constraint intervals were applied to critical memristor model parameters for the Det-MemODE and Dual-NN-MemODE approaches:

- $R_{\text{ON}} \in (0, R_{\text{OFF}})\,\text{k}\Omega$ and $R_{\text{OFF}} \in (R_{\text{ON}}, 200)\,\text{k}\Omega$ – These constraints establish a physically meaningful hierarchy where the ON-state resistance is strictly smaller than the OFF-state resistance ($R_{\text{ON}} < R_{\text{OFF}}$), consistent with typical memristor switching behavior. The lower bound for R_{ON} prevents zero resistance values that would cause numerical singularities in current calculations, while the upper bound of 200 kΩ for R_{OFF} reflects experimentally observed resistance ranges in memristive devices and ensures computational stability during optimization.

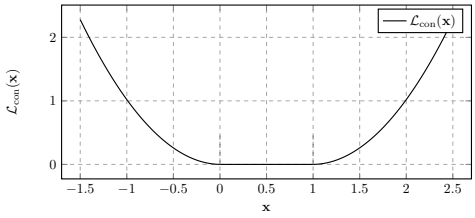

Fig. 5. Operational characteristics of the constraint function $\mathcal{L}_{\text{con}}(\mathbf{x})$ demonstrating smooth penalty transitions for parameter bounds $a = 0$, $b = 1$ with transition margin $\delta = 0.01$ evaluated over the domain $\mathbf{x} \in (-0.5, 1.5)$. The function exhibits minimal penalty within the feasible region $(0, 1)$ and progressively increasing penalties as parameters approach or exceed the boundary limits.

- $x(t) \in (0, 1)$ – This temporal constraint maintains the internal state variable within the normalized domain throughout the entire simulation duration, preserving the physical interpretation of $x(t)$ as a fractional parameter representing the extent of ionic migration or structural modification within the memristive element.
- $\mathcal{G} \in (0, 2)\text{mS}$ – This constraint limits the conductance parameter to a realistic range based on empirical measurements of memristor devices, preventing unphysically high conductance values that could lead to numerical instability or divergence during training, and ensuring passive device nature.

These constraints are particularly crucial for memristor modeling because they preserve the fundamental physical relationships governing resistive switching phenomena while preventing the emergence of artifacts that could arise from parameter values outside the feasible operating regime.

The operational characteristics of the constraint function $\mathcal{L}_{\text{con}}(\mathbf{x})$ are illustrated in Figure 5 for the parameter configuration: $a = 0$, $b = 1$, $\delta = 0.01$ over the evaluation domain $\mathbf{x} \in (-0.5, 1.5)$. This visualization demonstrates the smooth sigmoid-based transition behavior that provides continuous penalty gradients while maintaining differentiability across the parameter space. The constraint function architecture ensures that parameters within the admissible range $(0, 1)$.

III. Results

During the preliminary experiments, a range of neural network architectures was systematically evaluated in order to identify the most suitable configuration for modeling memristor dynamics. The evaluation considered different numbers of hidden layers, neurons per layer, and optimization settings, with the aim of achieving a balance between model accuracy, generalization capability, and computational efficiency. A representative learning curve corresponding to the Det-MemODE architecture, consisting of two hidden layers with 128 neurons each, trained using the `Adam` optimizer with a learning rate of 10^{-2} and a batch size of 2, is presented in Figure 6.

The training process was conducted over 260 epochs, with the loss function value monitored on both the training and validation datasets to provide insights into convergence behavior and potential overfitting. The learning curve demonstrates a consistent and smooth decrease in loss values for both datasets,

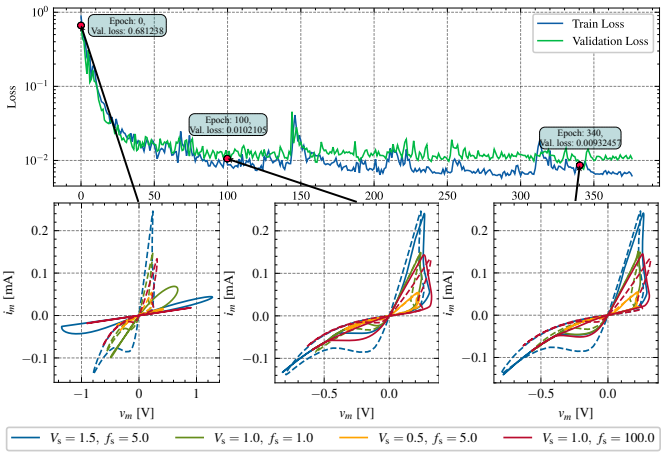

Fig. 6. Learning curve for the Det-MemODE neural network architecture during training over 360 epochs. The figure depicts a gradual decrease in the loss function values for both the training and validation datasets, reflecting effective optimization and adequate convergence of the model toward the experimental memristor data. The insets illustrate the evolution of the $v_m - i_m$ hysteresis loops at selected epochs, where the dashed lines correspond to the measured responses and the solid lines denote the model predictions. These results demonstrate the model's ability to accurately reproduce the nonlinear and history-dependent characteristics of the memristor as training advances.

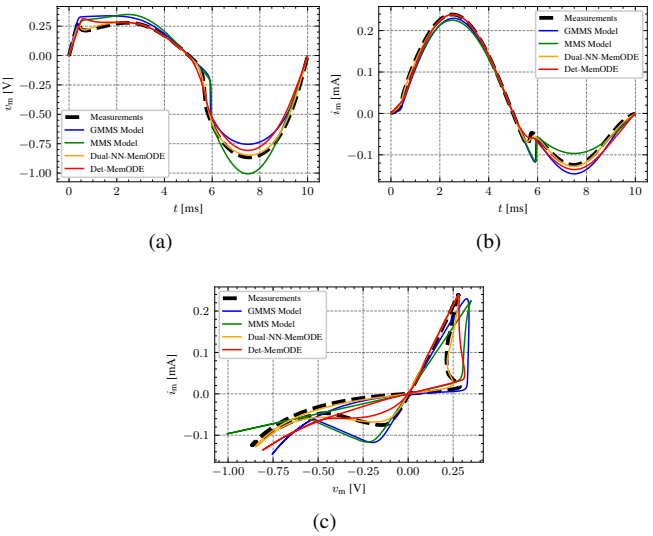

Fig. 7. Representative simulation results for tungsten-doped memristor dynamics under sinusoidal voltage excitation (amplitude: 1.5V, frequency: 100Hz). (a) Applied voltage waveform demonstrating the sinusoidal input stimulus, (b) Predicted memristor current response showing nonlinear temporal evolution and switching characteristics, (c) Voltage-current hysteresis loop ($v_\mathrm{m} - i_\mathrm{m}$) illustrating the characteristic pinched behavior and memory properties fundamental to memristive operation. The neural network model successfully captures the essential features of memristor dynamics including switching thresholds, nonlinear conductance modulation, and hysteretic memory effects.

which indicates effective optimization and satisfactory model fitting to the experimental data. The close agreement between the training and validation loss further confirms the generalization ability of the selected architecture.

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

Representative simulation results demonstrating the neural network model's predictive capabilities are presented in Figure 7 through comparative analysis of a tungsten-doped memristor device subjected to sinusoidal voltage excitation with an amplitude of 1V and frequency of 1Hz. These specific excitation parameters were selected to capture the characteristic switching dynamics while maintaining operation within the device's linear regime, enabling comprehensive evaluation of the model's ability to reproduce fundamental memristive behaviors.

The analysis encompasses three critical dynamic characteristics that collectively characterize memristor behavior: temporal current evolution, applied voltage profiles, and voltage-current hysteresis relationships. The temporal current response reveals the device's instantaneous electrical behavior and switching kinetics, while the applied voltage profile confirms the fidelity of the input stimulus. Most significantly, the voltage-current hysteresis loops ($v_m - i_m$) demonstrate the nonlinear relationship between these electrical quantities and illustrate the fundamental memory properties that define memristive behavior.

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

To establish the performance advantages of the neural differential equation approach, a comprehensive comparative analysis was conducted between the developed neural network model and the established deterministics model especially conducted for the SDC memristor modelling task: Mean Metastable Switch (MMS) with the Eq. (4) and Generalized Mean Metastable Switch (GMMS) model with the Eq. (7).

This comparison is particularly significant as the MMS model has been extensively validated against experimental data and serves as a benchmark for memristor circuit simulation in the scientific literature. The comparative evaluation encompasses temporal voltage and current evolution, hysteretic characteristics, and quantitative loss function metrics to provide comprehensive assessment of modeling fidelity across multiple performance dimensions.

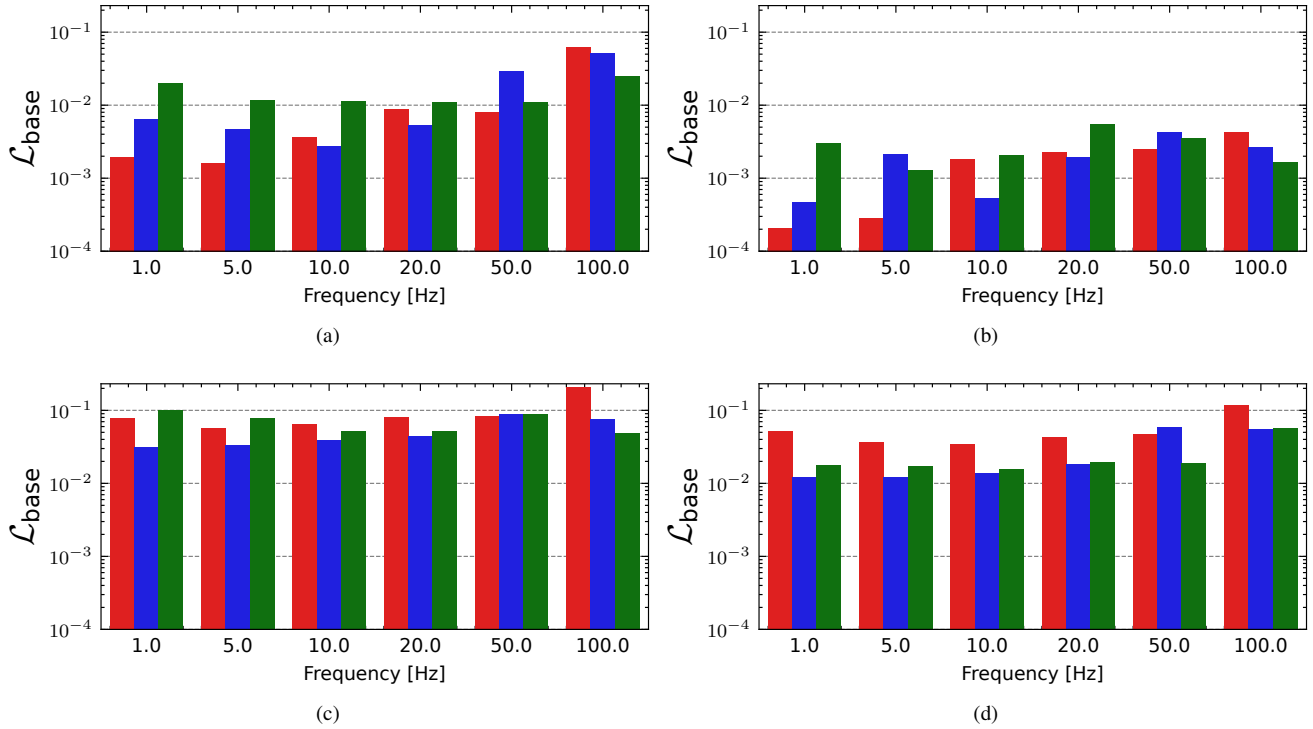

Fig. 8. Comparative analysis of loss function evolution where the following colors distinguish different supply voltage amplitudes: (■) $V_s = 0.5$ V, (■) $V_s = 1.0$ V, and (■) $V_s = 1.5$ V. The models compared are (a) the Det-MemODE model, (b) the Dual-NN-MemODE model, (c) the Mean Mean Metastable Switch (MMS) model, and (d) the Generalized Mean Metastable Switch (GMMS) model. The neural network demonstrates superior convergence characteristics and achieves significantly lower final loss values, indicating enhanced modeling accuracy and better agreement with experimental memristor dynamics.

Quantitative analysis reveals substantial performance improvements achieved by the neural network approach. The average loss function value for the Det-MemODE neural network model was 0.015223, and 0.002226 for the Dual-NN-MemODE model, representing a significant improvement compared to the MMS model's average loss of 0.072356, and GMMS model 0.035835. This near two times reduction in loss indicates markedly superior agreement between neural network predictions and experimental data, demonstrating the enhanced modeling capability achieved through the neural differential equation framework.

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

Table I presents a comparative analysis of the representative loss functions obtained for the neural network and MMS models, offering a quantitative assessment of their performance across key electrical characteristics. 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.

B. Hyperparameter Sensitivity Analysis

Hyperparameter optimization was conducted to systematically explore the parameter space and identify configurations that maximize the predictive performance of the neural network. The optimization employed the Tree-structured Parzen Estimator (TPE) algorithm, implemented in the `Optuna` framework [18]. The TPE algorithm models the objective function probabilistically by constructing two separate density estimators: one for configurations associated with above-average performance and another for those yielding below-average results. By sampling preferentially from the former, while still maintaining diversity through exploration of the latter, the algorithm achieves an efficient balance between exploitation of promising regions and exploration of the broader search space.

The search space encompassed both architectural and training-related hyperparameters. On the architectural side, the number of hidden layers was varied between one and five, with layer widths ranging from 16 to 512 neurons in increments of 16. Candidate activation functions included widely used nonlinearities such as ReLU, GELU, SiLU, $\tanh$, ELU, leaky ReLU, and sigmoid, as well as identity and sine functions. For the output layer, the choice was restricted to linear, $\tanh$, sigmoid, ReLU. The training-related search space included the learning rate, sampled on a logarithmic scale

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

Loss function comparison for $V_s = 0.5$ V				
f_s [Hz]	Det-MemODE	Dual-NN-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 comparison for $V_s = 1$ V				
f_s [Hz]	Det-MemODE	Dual-NN-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 comparison for $V_s = 1.5$ V				
f_s [Hz]	Det-MemODE	Dual-NN-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}

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

	Det-MemODE	Dual-NN-MemODE	GMMS Model	MMS Model
$\mathcal{L}_{\text{base}}$	2.16×10^{-2}	7.08×10^{-3}	4.19×10^{-2}	5.30×10^{-2}
$\mathcal{L}_{\text{v}}$	2.96×10^{-1}	2.22×10^{-1}	1.36	7.28×10^{-1}
$\mathcal{L}_{\text{c}}$	3.67×10^{2}	2.39×10^{2}	2.10	5.02

between 10^{-4} and 10^{-1}, batch size ranging from 1 to 18, optimizer type (Adam, SGD, AdamW, Nadam, AdaBelief), and additional parameters related to regularization and scheduling, including patience, cooldown, reduction factor, tolerance, and weight decay.

To quantify the relative influence of individual hyperparameters on model performance, a surrogate-based importance analysis was performed. An XGBoost gradient boosting model [19] was trained on the explored configurations to predict validation loss. The surrogate achieved an R^2 value of 0.99, indicating high fidelity in capturing the dependence of model performance on hyperparameter selection. Importance was first estimated using the *gain* metric, which reflects the cumulative improvement in predictive accuracy attributable to each hyperparameter across the ensemble. Normalization ensured that scores summed to unity, allowing for direct interpretation as relative contributions to performance variability.

As a complementary approach, permutation importance [20] was applied to the surrogate model. In this procedure, the values of each hyperparameter were permuted at random, and the resulting degradation in predictive accuracy was recorded. Repeated ten times for each feature, the mean error increase provided an alternative measure of importance, which, unlike gain-based metrics, also accounts for nonlinear dependencies and interactions among hyperparameters.

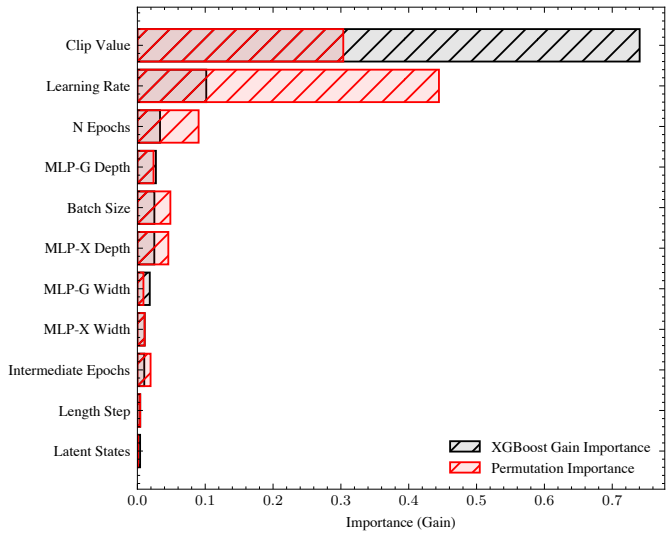

Fig. 9. Hyperparameter importance for the Dual-NN-MemODE optimization task. Scores were computed using permutation importance applied to an XGBoost surrogate model trained on 200 hyperparameter configurations. Larger values indicate greater relative influence of the corresponding hyperparameter on model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

Both approaches consistently identified the learning rate and gradient clipping value as the dominant hyperparameters. According to the gain-based measure, their contributions were 0.10 and 0.74, respectively, while the permutation-based analysis yielded corresponding values of 0.44 and 0.30. Together, these two parameters explained approximately 63% of the observed variance in performance. By contrast, the activation function exhibited only marginal influence (importance score of 0.04), indicating that the Neural ODE architecture is relatively robust to this choice within the tested range.

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, ISSN: 1476-4687. DOI: 10.1038/nature06932. [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, ISSN: 2072-666X. DOI: 10.3390/mi15020253. [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," 2018.

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

[6] B. Garda and K. Bednarz, "Comprehensive study of sdc memristors for resistive ram applications," *Energies*, vol. 17, no. 2, p. 467, Jan. 2024, ISSN: 1996-1073. DOI: 10.3390/en17020467. [Online]. Available: http://dx.doi.org/10.3390/en17020467.

[7] K. A. Campbell, "Self-directed channel memristor for high temperature operation," *Microelectronics Journal*, vol. 59, pp. 10–14, Jan. 2017, ISSN: 1879-2391. DOI: 10.1016/j.mejo.2016.11.006. [Online]. Available: http://dx.doi.org/10.1016/j.mejo.2016.11.006.

[8] L. Chua and S. M. Kang, "Memristive devices and systems," *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976, ISSN: 0018-9219. DOI: 10.1109/proc.1976.10092. [Online]. Available: http://dx.doi.org/10.1109/PROC.1976.10092.

[9] A. Paszke *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, 2019. arXiv: 1912.01703 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1912.01703.

[10] R. T. Q. Chen, *Torchdiffeq*, 2018. [Online]. Available: https://github.com/rtqichen/torchdiffeq.

[11] 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. DOI: 10.1145/3292500.3330701. [Online]. Available: http://dx.doi.org/10.1145/3292500.3330701.

[12] 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. 109 990, Sep. 2020, ISSN: 0960-0779. DOI: 10.1016/j.chaos.2020.109990. [Online]. Available: http://dx.doi.org/10.1016/j.chaos.2020.109990.

[13] 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, ISSN: 2079-4991. DOI: 10.3390/nano12010063. [Online]. Available: http://dx.doi.org/10.3390/nano12010063.

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

[15] 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, ISSN: 1557-7295. DOI: 10.1145/279232.279236. [Online]. Available: http://dx.doi.org/10.1145/279232.279236.

[16] 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. DOI: 10.1109/cvpr.2018.00781. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2018.00781.

[17] 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*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Oct. 2018, pp. 794–803. [Online]. Available: https://proceedings.mlr.press/v80/chen18a.html.

[18] 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.

[19] 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. DOI: 10.1145/2939672.2939785. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785.

[20] 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, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btq134. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btq134.

Jane Doe Biography text here without a photo.

IEEE Publications Technology Team In this paragraph you can place your educational, professional background and research and other interests.